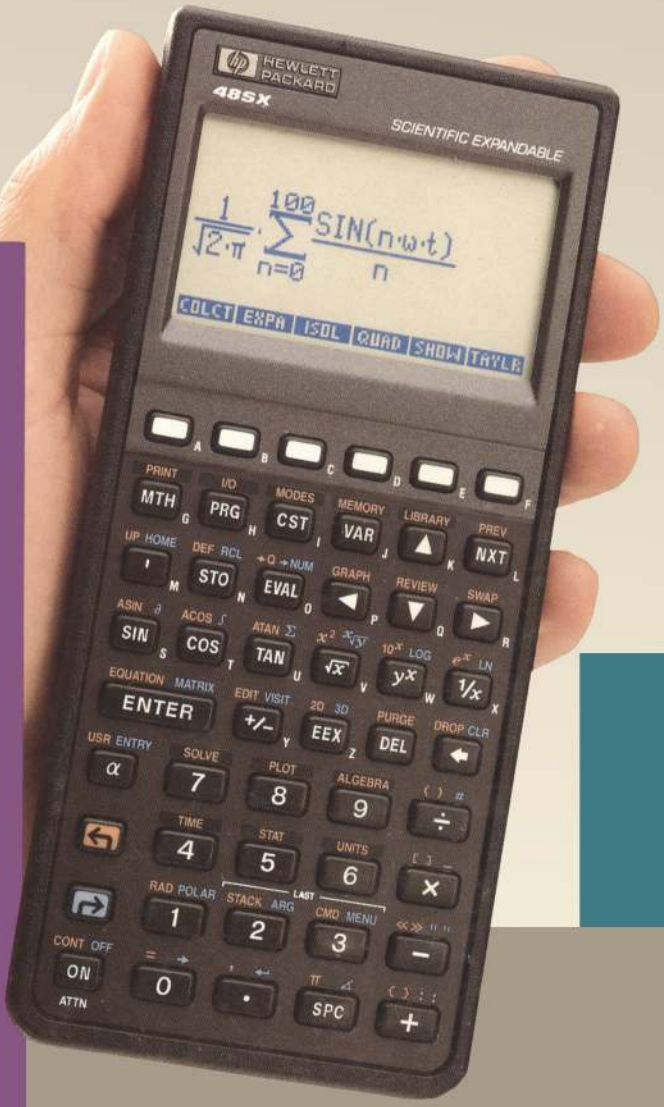
HEWLETT PACKARD

HP 48SX
Scientific Expandable



Owner's Manual
Volume I

Comments on the HP 48SX Owner's Manual

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications.

HP 48SX Owner's Manual

Please circle a response for each of the statements below. You can use the **Comments** space to provide additional opinions.

1 = Strongly Agree
2 = Agree
3 = Neutral
4 = Disagree
5 = Strongly Disagree

- I am satisfied with the product documentation. 1 2 3 4 5
- The manual is well organized. 1 2 3 4 5
- I can find the information I want. 1 2 3 4 5
- The information in the manual is accurate. 1 2 3 4 5
- I can easily understand the instructions. 1 2 3 4 5
- The manual contains enough examples. 1 2 3 4 5
- The examples are appropriate and helpful. 1 2 3 4 5
- The illustrations are clear and helpful. 1 2 3 4 5
- The amount of information is: too much appropriate too little
- The parts I refer to most frequently are:

Part 1: Building Blocks Part 2: Hand Tools Part 3: Power Tools
Part 4: Programming Part 5: Printing, Data Transfer, and Plug-Ins
Appendixes Indexes

Comments: _____

_____

_____

_____

_____

_____

Name: _____

Address: _____

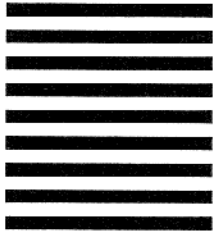
City/State/Zip: _____

Occupation: _____

Phone: (_____) _____



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 38 CORVALLIS, OR

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD COMPANY
DOCUMENTATION DEPARTMENT
CORVALLIS DIVISION
1000 NE CIRCLE BLVD
CORVALLIS OR 97330-9973



HP 48SX Scientific Expandable Calculator

Owner's Manual
Volume I



Edition 4 July 1990
Reorder Number 00048-90003

Notice

For warranty and regulatory information for this calculator, see pages 673 and 676.

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.

© Hewlett-Packard Co. 1990. 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 your calculator 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.

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

Printing History

Edition 1	January 1990	Mfg. No. 00048-90002
Edition 2	April 1990	Mfg. No. 00048-90058
Edition 3	May 1990	Mfg. No. 00048-90061
Edition 4	July 1990	Mfg. No. 00048-90077

Contents

20 How to Use This Manual
20 The Parts in This Manual
21 The Conventions Used in This Manual
21 For More Information

Part 1: Building Blocks

1

24 Getting Started
25 Turning the Calculator On and Off
25 Adjusting the Display Contrast
25 Introducing the Keyboard
27 Discovering the Power of the HP 48
41 Setting the Time and Date
42 Classifying HP 48 Operations
44 Where to Go from Here

2

45 The Keyboard and Display
45 Organization of the Display
46 The Stack
46 The Command Line
47 Keying In Numbers
48 The Status Area, Annunciators, and Messages
50 Organization of the Keyboard
52 The Shift Keys
52 The Alpha Keyboard

53 Generating Accented Characters
54 The Attention Key
55 Keying in Delimiters
55 Menus
56 Displaying Menus
57 Switching to the Last Menu
57 Display Modes

3 60 **The Stack and Command Line**
61 Using the Stack for Calculations
61 An Overview
61 One-Argument Commands
62 Two-Argument Commands
62 Using Previous Results (Chain Calculations)
63 Swapping Levels 1 and 2
64 Clearing the Stack
64 Recovering the Last Arguments
65 Duplicating Level 1
66 Displaying Objects for Viewing or Editing
67 Viewing and Editing an Object
67 Viewing and Editing the Contents of a Variable
68 The EDIT Menu
70 The Interactive Stack
75 Using the Command Line
75 Accumulating Data in the Command Line
76 Entry Modes
77 Recovering Previous Command Lines
78 Other Stack Commands

4 80 **Objects**
81 Real Numbers
81 Complex Numbers
82 Binary Integers
83 Arrays
84 Names
85 Algebraic Objects
85 Programs
86 Strings

86 Lists
87 Graphics Objects
87 Tagged Objects
88 Unit Objects
89 Directory Objects
89 Additional Object Types
90 Commands that Manipulate Objects
97 Object Types
97 Determining Object Types
98 Separating Variable Names by Object Type
98 Evaluating Objects

5 100 **Calculator Memory**
100 Types of Memory
101 Memory Utilization Commands
101 Clearing All Memory
102 Low-Memory Conditions

6 105 **Variables and the VAR Menu**
107 Creating a Variable
107 The STO Command
107 The DEFINE Command
108 Variable Names
109 Using the Contents of a Variable
109 Evaluating a Variable's Name
110 Recalling the Contents of Variables
111 Changing the Contents of Variables
112 Using Quoted Versus Unquoted Variable Names
112 The VAR Menu and REVIEW Catalog
113 Reordering the VAR Menu
114 Purging Variables
114 Purging Individual Variables
114 Purging More Than One Variable
115 Purging All Variables
115 Error Recovery
115 Variable Arithmetic

7 118 **Directories**
118 Directory Concepts
120 Creating Subdirectories
121 Creating and Accessing Variables in Directories
121 Creating Variables
121 Accessing Variables
122 Changing Directories
122 Switching to a Subdirectory
122 Switching to the Parent or HOME Directory
122 Purging Variables and Directories
122 Purging the Contents of a Directory
123 Purging a Directory
123 Using Directory Objects

8 125 **More About Algebraic Objects**
125 Evaluation of Algebraics
127 Symbolic Versus Numerical Results
128 Automatic Simplification
128 The Rules of Algebraic Precedence
129 Expressions and Equations
130 Related Topics

Part 2: Hand Tools

9 132 **Common Math Functions**
133 The MTH (MATH) Menu
134 Arithmetic and General Math Functions
136 Fraction Conversion Functions
137 Exponential, Logarithmic, and Hyperbolic Functions
138 Percent Functions
139 Angle Mode, Trigonometric Functions, and π
139 Selecting the Angle Mode
140 Trigonometric Functions
140 Using π
142 Angle Conversion Functions
144 Symbolic Constants

144 Using Symbolic Constants
144 Using Values for Symbolic Constants
145 Using Flags to Interpret Symbolic Constants
147 Factorial, Probability, and Random Numbers
148 Other Real-Number Functions
149 Using Symbolic Arguments with Common Math Functions

10 150 **User-Defined Functions**
151 Creating a User-Defined Function
152 Executing a User-Defined Function
153 Nesting User-Defined Functions
154 The Structure of a User-Defined Function

11 156 **Complex Numbers**
157 Entering and Displaying Complex Numbers
157 How Complex Numbers Are Displayed
158 Entering Complex Numbers
160 Assembling and Taking Apart Complex Numbers
161 Calculating with Complex Numbers
161 Comparison with Real Number Calculations
163 Complex Results from Real Operations
164 Complex Numbers in Algebraics
164 Algebraics Containing Complex Numbers
165 Using the Symbolic Constant i
165 Additional Commands for Complex Numbers
167 Complex Numbers or Vectors?

12 169 **Vectors**
170 Displaying 2D and 3D Vectors
170 How 2D Vectors Are Displayed
171 How 3D Vectors Are Displayed
172 Entering 2D and 3D Vectors
173 Assembling and Taking Apart 2D and 3D Vectors
176 2D and 3D Vector Calculations
183 Additional Vector Commands
184 Complex Numbers or Vectors?

13 **185** **Unit Management**
187 How the Units application Is Organized
187 Definition of Terms
188 The UNITS Catalog Menu
188 Creating a Unit Object
190 Creating a Unit Object in the Command Line
191 Reviewing Unit Names
191 Unit Objects in Algebraics
192 Unit Prefixes
193 Unit-Object Conversion
193 Unit-Object Conversion in the UNITS Catalog Menu
194 Unit-Object Conversion with CONVERT
195 Unit-Object Conversion in the CST Menu
196 Conversion to SI Base Units
197 Temperature Conversion
198 Dimensionless Units of Angle
199 Unit-Expression Factoring
200 Unit-Object Arithmetic
204 Building Unit Objects with the EquationWriter Application
205 User-Defined Units
206 Additional Commands for Unit Objects

14 **207** **Binary Arithmetic**
207 Setting the Wordsize
208 Selecting the Base
208 Entering Binary Integers
209 Calculations with Binary Integers
210 Additional Binary Integer Commands

15 **212** **Customizing the Calculator**
212 Custom (CST) Menus
213 Creating a Custom Menu
213 Custom Menu Functionality
215 Enhancing Custom Menus
216 Creating a Temporary Menu
216 The User Keyboard
216 User Modes

217 Making User Key Assignments
219 Clearing User Key Assignments
219 Reactivating a Single Standard Key
220 Recalling and Editing User Key Assignments
220 Other Customizing Operations
220 The MODES Menu
222 System Flags

Part 3: Power Tools

16 226 **The EquationWriter Application**
228 How the EquationWriter Application Is Organized
230 Constructing an Equation
237 Turning Off Implicit Parentheses
237 EquationWriter Application Examples
240 Viewing Algebraics and Unit Objects in the EquationWriter Application
241 Editing Equations
241 Backspace Editing
242 Command-Line Editing
246 Inserting an Object from the Stack
247 Replacing a Subexpression with an Algebraic from the Stack
248 A Preview of the Rules Application

17 250 **The HP Solve Application**
253 The Structure of the HP Solve Application
254 Equations, Expressions, and Programs
255 The SOLVE Menu—Specifying the Current Equation
257 Entering a New Current Equation
258 The Equation Catalog—Selecting and Managing Existing Equations
263 The SOLVR Menu—Solving the Current Equation
265 Using EXPR=
266 Choosing Guesses
266 Solving Equations with the Plot Application

267 Using the HP Solve Application with Unit Objects
269 Customizing the SOLVR Menu
272 Linking Two or More Equations
272 Using EQ+ to Link Two or More Equations
274 Saving a List of Two or More Equations
275 Using the HP Solve Application to Find Solutions of Programs
276 How the HP Solve Application Works
277 How the Root-Finder Uses Initial Guesses
277 Halting the Root-Finder
278 Displaying Intermediate Guesses
278 How the Menu of Variables Is Created
279 Interpreting Results
279 When a Solution is Found
282 When No Solution is Found

18 283 **Basic Plotting and Function Analysis**
286 The Structure of the Plot Application
289 Equations, Expressions, and Programs
290 The PLOT Menu—Specifying the Current Equation and Plot Type
291 The PLOTR Menu—Setting Plot Parameters and Drawing the Plot
294 Specifying the Independent Variable
295 Display Ranges and Scaling
295 Drawing the Graph
300 Plotting Two or More Equations
301 The Graphics Environment—Zoom Operations and Function Analysis
304 Zoom Operations
307 Analyzing Functions

19 317 **More About Plotting and Graphics Objects**
318 Refinement Options for Plots
319 Plotting Range Versus Display Range
320 Specifying Axes and Labels
321 Specifying Resolution

322 The Plot Parameter Variable (PPAR)
323 Coordinate Units
325 Changing the Size of PICT
327 Plot Types
328 Function Plots
329 Conic Sections
330 Polar Plots
332 Parametric Plots
333 Truth (Relational) Plots
334 Plotting Programs and User-Defined Functions
335 Plotting with Units
335 Plotting Statistical Data
336 Adding Graphical Elements to PICT
337 Operations In the Graphics Environment for Adding Graphical Elements to PICT
339 Programmable Commands for Adding Graphical Elements to PICT
340 Working with Graphics Objects on the Stack
341 Stack-Related Operations in the Graphics Environment
341 The PICT Command—Working with PICT on the Stack
342 Manipulating Graphics Objects on the Stack

20 345 **Arrays**
346 Entering Arrays
346 The MatrixWriter Application
349 Entering Arrays Using the Command Line
350 How Vectors Are Displayed
350 Editing Arrays
353 Arithmetic Operations with Arrays
353 Doing Arithmetic with Vectors
354 Doing Arithmetic with Matrices
355 Doing Arithmetic with a Matrix and a Vector
356 Solving a System of Linear Equations
357 Complex Arrays
357 Arithmetic with Complex Arrays
357 Additional Complex Array Commands

359 Additional Matrix Commands
361 Advanced Topics Relating to Matrices

21 364 **Statistics**
367 Starting the Statistics Application
369 Designating the Current Statistical Matrix
369 Entering New Statistical Data (Σ+ and NEW)
370 Editing Data
370 Using the STAT Catalog
373 Exiting the Catalog
374 Calculating Single-Variable Statistics
374 Sample Statistics
375 Population Statistics
375 Paired-Sample Statistics
378 Plotting
379 Plotting Bar Charts
381 Plotting Histograms
383 Summation Statistics
383 Test Statistics

22 386 **Algebra**
389 Symbolic Solutions
389 Isolating a Variable
391 Solving Quadratic Equations
393 General and Principal Solutions
394 Showing Hidden Variables
395 Rearranging Terms
395 Collecting Terms
396 Expanding Products and Powers
397 The Rules Transformations
398 The Selection Environment — Specifying a Subexpression
399 Selecting the Rules Transformations
399 Executing a Rules Transformation
400 Exiting a RULES Menu
400 Rules Examples
410 Summary Examples
415 User-Defined Transformations

416 The | (Where) Function

23 418 **Calculus**
419 Differentiation
420 Stepwise Differentiation
422 Complete Differentiation
422 Differentiation of User-Defined Functions
422 Advanced Topic: User-Defined Derivatives
423 Summations
426 Taylor's Polynomial Approximations
428 Integration
428 Symbolic Integration
431 Taylor's Polynomial Approximation of the Integrand
432 Numerical Integration
436 Using the Stack for Integration

24 437 **Time, Alarms, and Date Arithmetic**
439 The Structure of the Time Application
440 Setting the Date and Time
441 Setting the Date
442 Setting the Time
442 Changing the Date and Time Formats
443 Adjusting the Time
443 Setting Alarms
445 Appointment Alarms
445 Repeating an Alarm
446 Acknowledging an Appointment Alarm
447 Unacknowledged Appointment Alarms
448 Setting a Control Alarm
448 Recovery from Short-Interval Repeating Alarms
449 Reviewing and Editing Alarms
453 Using Alarms in Programs
454 Date Arithmetic
456 Time Arithmetic

Part 4: Programming

25 **468** **Programming Fundamentals**
470 Entering and Executing a Program
470 Entering a Program
472 Executing a Program
472 Editing a Program
473 Using Local Variables
479 Programs That Manipulate Data on the Stack
480 Using Subroutines
483 Single-Step Execution of a Program
484 Single-Step Execution from the Start of the Program
486 Single-Step Execution from the Middle of the Program
486 Single-Step Execution of Subroutines
487 Adding Comments to a Program

26 **488** **Tests and Conditional Structures**
490 Program Tests
491 Comparison Functions
493 Logical Functions
494 Testing Object Types
494 Conditional Structures
494 The IF...THEN...END Structure
496 The IF...THEN...ELSE...END Structure
498 The CASE...END Structure
499 Conditional Commands
500 The IFT (If-Then-End) Command
500 The IFTE Function

27 **501** **Loop Structures**
501 Definite Loop Structures
502 The START...NEXT Structure
504 The START...STEP Structure
506 The FOR...NEXT Structure
508 The FOR...STEP Structure

510 Indefinite Loop Structures
510 The DO...UNTIL...END Structure
512 The WHILE...REPEAT...END Structure
513 Loop Counters (INCR and DECR)

28 515 **Flags**
515 Flag Types
516 Setting, Clearing, and Testing Flags
518 Recalling and Storing the Flag States
518 Recalling the Flag States
518 Storing the Flag States

29 519 **Interactive Programs**
520 Suspending Program Execution for Data Input
521 The PROMPT Command
523 The BEEP Command
523 The DISP, HALT and FREEZE Commands
524 The INPUT Command
531 Labeling Program Output
532 Using Tagged Objects as Data Output
533 Using String Commands to Label Data Output
534 Pausing to Display Data Output
534 Using Menus in Programs
534 Displaying a Built-In Menu
535 Custom Menus in Programs
539 Building a Temporary Menu
539 Commands That Return a Key Location
539 The WAIT Command with Argument 0
539 The WAIT Command with Argument −1
540 The KEY Command
540 Turning the HP 48 Off from a Program

30 541 **Error Trapping**
543 The IFERR...THEN...END Structure
544 The IFERR...THEN...ELSE...END Structure
546 User-Defined Errors

31 547 **More Programming Examples**
548 Fibonacci Numbers
548 FIB1 (Fibonacci Numbers, Recursive Version)
550 FIB2 (Fibonacci Numbers, Loop Version)
551 FIBT (Comparing Program-Execution Time)
554 Displaying a Binary Integer
554 PAD (Pad with Leading Spaces)
555 PRESERVE (Save and Restore Previous Status)
557 BDISP (Binary Display)
560 Median of Statistics Data
561 SORT (Sort a List)
563 LMED (Median of a List)
565 MEDIAN (Median of Statistics Data)
568 Expanding and Collecting Completely
569 MULTI (Multiple Execution)
570 EXCO (Expand and Collect Completely)
572 Finding the Minimum or Maximum Element of an Array
573 MNX (Finding the Minimum or Maximum Element of an Array—Technique 1)
576 MNX2 (Finding the Minimum or Maximum Element of an Array—Technique 2)
579 Verification of Program Arguments
580 NAMES (Does the List Contain Exactly Two Names?)
582 VFY (Verify Program Argument)
585 Bessel Functions
588 Animation of Successive Taylor's Polynomials
588 Drawing a Sine Curve and Converting It to a Graphics Object
589 Superposition of Successive Taylor's Polynomials
591 Animation of Taylor's Polynomials
592 Programmatic Use of Statistics and Plotting
597 Animation of a Graphical Image

Part 5: Printing, Data Transfer, and Plug-Ins

32 **602** **Printing**

602 Printing with an HP 82240B Printer
604 Print Formats
605 Basic Printing Commands
606 Printing a Text String
606 Printing a Graphics Object
607 Double Space Printing
607 Setting the Delay
607 The HP 48 Character Set
608 Sending Escape Sequences and Control Codes
608 Accumulating Data in the Printer Buffer
609 Printing with an HP 82240A Infrared Printer
610 Printing to the Serial Port
611 The PRTPAR Variable

33 **612** **Transferring Data to and from the HP 48**

613 Types of Data You Can Transfer
614 The I/O Menu
616 Local and Server Modes
617 Setting the I/O Parameters
617 The SETUP Menu
618 The IOPAR Variable
619 Transferring Data between Two HP 48's
621 Transferring Data between a Computer and the HP 48
621 Cable Connection
622 Transferring Data
624 Backing Up All of HP 48 Memory
626 Character Translations (TRANSIO)
628 More About File Names
629 Errors
629 ASCII and Binary Transmission Modes
631 Sending Commands to a Server (PKT)
632 Serial Commands

34 635 **Using Plug-in Cards and Libraries**
635 Types of Memory
636 Installing and Removing Plug-In Cards
639 RAM Cards
639 Preparing the Card for Installation
642 Uses for RAM Cards
643 Using RAM Cards to Expand User Memory (Merged Memory)
644 Using RAM Cards for Backup (Independent Memory)
645 Backing Up Objects into Independent Memory
646 Accessing Backup Objects
647 Backing Up Objects into User Memory (Port 0)
648 Backing Up All of Memory
649 Freeing Merged Memory
651 Using Application Cards and Libraries
651 Attaching a Library to a Directory
652 Accessing Library Operations (The LIBRARY Menu)
653 Additional Commands That Access Libraries

Appendixes and Indexes

A 656 **Support, Batteries, and Service**
656 Calculator Support
656 Answers to Common Questions
660 Environmental Limits
660 When to Replace Batteries
661 Changing Batteries
661 Battery Types
661 Changing Calculator Batteries
663 Changing a RAM Card Battery
665 Testing Calculator Operation
667 Self-Test
667 Keyboard Test
669 Port RAM Test
670 IR Loop-Back Test
671 Serial Loop-Back Test

673 Limited One-Year Warranty
674 If the Calculator Requires Service
676 Regulatory Information

B 677 **Messages**

C 694 **HP 48 Character Codes**

D 697 **Menu Numbers**

E 699 **Listing of HP 48 System Flags**

707 **Operation Index**

823 **Index**

How to Use This Manual

This two-volume manual is designed to be used with your HP 48SX Scientific Expandable calculator in the following way:

1. Read chapter 1, "Getting Started," to learn about the functionality, power, and ease of use of the HP 48.
2. Read the rest of the chapters in "Part 1: Building Blocks" to gain a fundamental understanding of HP 48 operations.
3. Then, use the table of contents to find other topics in which you're interested.

The Parts in This Manual

There are five parts in this manual:

- *Part 1: Building Blocks* contains information fundamental to the effective operation of the HP 48.
- *Part 2: Hand Tools* contains chapters covering useful HP 48 tools that deal with topics like general math, complex numbers, vectors, and calculator customization.
- *Part 3: Power Tools* contains chapters describing the built-in applications that make the HP 48 so powerful—like the EquationWriter, HP Solve, and Plot applications.
- *Part 4: Programming* covers all aspects of HP 48 programming.
- *Part 5: Printing, Data Transfer, and Plug-Ins* describes the processes of printing data, transferring data to and from a PC or another HP 48, and using RAM cards and application cards.

The Conventions Used in This Manual

As you work through this manual, you'll need to understand certain conventions consistently used throughout both volumes.

- Keys on the keyboard are always shown in a special, boxed typeface — for example, the enter key is shown as [ENTER] and the store key is shown as [STO].
- Menu labels, which are found at the bottom of the display and relate to the keys on the top row of the keyboard, are shown in their own special typeface — for example, PARTS, PROB, and HYP.
- Variables are shown in text in italic typeface — for example, *name*.
- Text that represents something from the display is shown in dot matrix typeface — for example, the display message Memory Clear.
- Programmable commands are shown in all uppercase letters — for example, DUP, EVAL, and SIN.

For More Information

If you ever have a question about an HP 48 operation, you can look it up in the alphabetical "Operation Index" in the back of volume II. There you'll find the name and description of the operation, the keystrokes to execute it, and a page reference for more information.

Also at the back of volume II, you'll find a comprehensive subject index that will direct you to information on any topic covered in this manual.

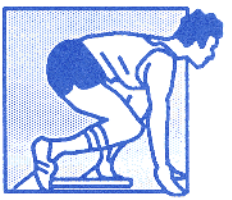
For those of you who plan to do extensive programming with your HP 48, a *Programmer's Reference Manual* (part number 00048-90054) is available from your Hewlett-Packard dealer.

Part 1

Building Blocks

1

Getting Started



Chapter 1 uses an interesting and fun example to get you started with your HP 48 calculator. By working through the example, you'll get a feel for the power and ease of use the HP 48 offers.

The key features introduced in this chapter include:

- *The EquationWriter application*, which enables you to enter and view equations in a form like you would see in textbooks.
- *Symbolic math*, which enables you to obtain a general solution to a problem as well as specific solutions at given points.
- *The Plot application*, which enables you to view graphically math functions and data, and then, while viewing the graph, to find roots, intersections, local extremes, derivatives, slopes, and areas under the curve.
- *The HP Solve application*, which enables you to find numerical solutions to problems without having to solve the equation explicitly.
- *Unit management*, which, when you're doing calculations with compatible units, automatically takes care of unit conversions for you.

Also, throughout the example in this chapter, you'll find short explanations of concepts and terminology specific to the HP 48. These interjections will help you develop a foundation for understanding the calculator.

Then, at the end of the chapter there are instructions for setting the time and date, as well as some additional information about the calculator that will prepare you to use the rest of this manual.

Turning the Calculator On and Off

The HP 48 is shipped with three AAA batteries already installed. When you take your calculator out of the box, it's ready to turn on.

Press [ON] to turn the calculator on. ([ON] is located at the lower-left corner of the keyboard.) To turn the calculator off, press [→] and then [ON]. Since the HP 48 has *continuous memory*, turning it off does not affect the information you've entered. To conserve power, the calculator turns itself off 10 minutes after you stop using it.

While the calculator is on, [ON] acts as the [ATTN] (attention) key; it clears any unprocessed information you've typed in and restores the calculator to normal operation.

Adjusting the Display Contrast

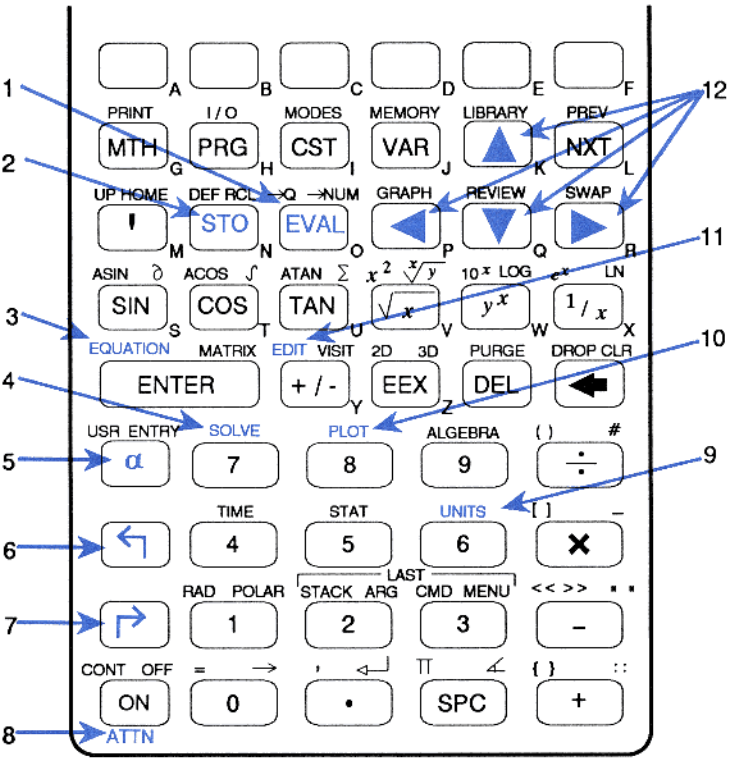
Any time the calculator is on, you can adjust the display contrast by holding down the [ON] key and pressing [+] (darker) or [−] (lighter). Try it now to get the contrast you like best.

Introducing the Keyboard

The HP 48 keyboard has several "levels." Each key has a primary key action shown on the key face — for instance, [ENTER], [+], [7], and [→]. Three of the primary keys, when pressed, redefine the key actions to match other sets of actions. The three keys are:

- [←] (called the *left-shift* key), which activates the orange key definitions above and to the left of each key.
- [→] (called the *right-shift* key), which activates the blue key definitions above and to the right of each key.
- [α] (called the *alpha* key), which activates the white, alphabetic key definitions to the lower right of many of the keys.

The following keyboard illustration highlights many of the keys that you'll be using in the example in this chapter.



Here is a description of the keys called out in the previous illustration:

1. [EVAL] evaluates an object.
2. [STO] stores an object in a variable.
3. [↰][EQUATION] selects the EquationWriter application.
4. [↰][SOLVE] selects the HP Solve application.
5. [α] activates the alpha keyboard.
6. [↰] activates the left-shift keyboard.
7. [↱] activates the right-shift keyboard.
8. [ON] becomes the [ATTN] key when the calculator is on.
9. [↰][UNITS] selects the Units application.
10. [↰][PLOT] selects the Plot application.
11. [↰][EDIT] copies an object into the command line for editing.
12. The four cursor keys ([▲], [▼], [▶], and [◀]) move the cursor in the display.

Discovering the Power of the HP 48

Work through the following example to preview some of what the HP 48 can do. Read the instructions, press the keys, and watch the display.

Example: Black Gold Ltd. The company you work for, Black Gold Ltd., has been experimenting with a new process for turning waste plastics into crude oil. After months of work, the research team gives you the following rate model:

$$R = \frac{G^2(1 + \sin(PT))}{1.4}$$

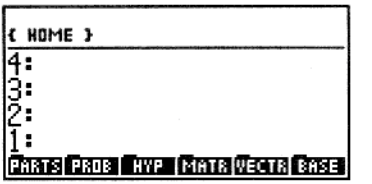
where:

- R is the yield rate in gallons per hour.
- G and P are process variables.
- T is the time in hours.

Analyze the model using your HP 48. As part of the analysis, determine how long it will take you to fill a 1,000-gallon tank with the oil produced, given P of 4.2 and G of 12.

Step 1: Turn on the calculator. (A quick tap that would execute other keys won't work with ON — it requires a solid press.)

ON





Note

This example assumes that no prior data has been entered into your calculator. If one or more of the variables *R*, *G*, *P*, *T*, *RATE*, or *VOLUME* already exist, the example won't work. You can purge those variables (see page 114), or change their names in the example, and then do the example.

If the variables don't exist, yet other data has been entered on the stack, the example will work, but a few of your displays may appear slightly different from those shown in this manual. If you want to clear your display to match exactly the displays shown here, press ↱ CLR (the blue, right-shifted definition of the ◆ key).

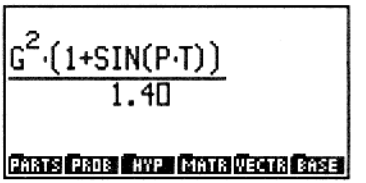
Step 2: Select the EquationWriter application. You'll use this application to enter the rate model in a form that looks just like an equation in a book. Remember, left-shifted (↰) characters are orange on the keyboard and right-shifted (↱) characters are blue. For the following keystroke sequence, pressing ↰ assigns EQUATION to the ENTER key. So to execute ↰ EQUATION, you press the ↰ key followed by the key with ENTER written on it.

↰ EQUATION



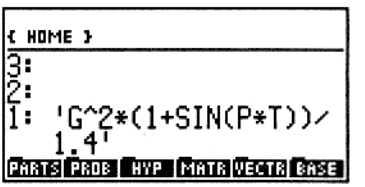
Step 3: Key in the right side of the rate equation. Alphabetic entry requires the use of the [α] key to activate the alpha keyboard. In this manual pressing [α] before a letter (like G) is assumed and not shown as part of the keystrokes. Therefore, to enter G in the keystrokes below, you press [α], followed by the [MTH] key, which has G printed in white just to the right of it. If you make a mistake, use [◆] to backspace over it, or use [ATTN] to exit the EquationWriter application and start over.

G [y^x] 2 [▶] [↰][( )] 1 [+] [SIN] P
[×] T [▶] [▶] [÷] 1.4



Step 4: Enter the expression onto the stack. When you enter a mathematical expression onto the stack from the EquationWriter application, it loses its textbook form and takes on a form more common to computers.

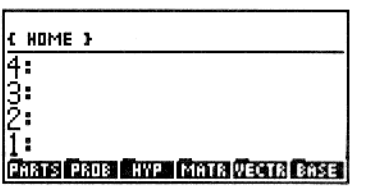
[ENTER]



The **stack** is simply a sequence of numbered (1:, 2:, 3:, ...) storage locations. At most, four levels of the stack are displayed at one time, but the stack can contain many more levels. When you enter an **object**—that is, a piece of data—onto the stack, it goes in level 1 and bumps all the other objects up a level. When objects are deleted or used in a calculation, they are removed from the stack and any other objects drop down to new levels accordingly. The rate expression you just entered went onto level 1 of the stack.

Step 5: Name the algebraic rate expression *RATE*.

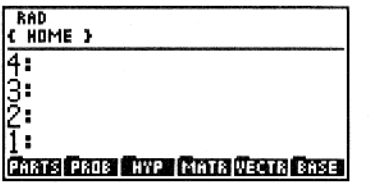
['] RATE [STO]



When you give an object (in this case an algebraic expression) a name, you are storing that object in a **variable**. You can then use the name to represent the object itself.

Step 6: Since the rate model you were given involves a trigonometric function, you need to select an *angle* mode. Set Radians mode. The RAD key is the left-shifted key found above the 1 key.

← RAD

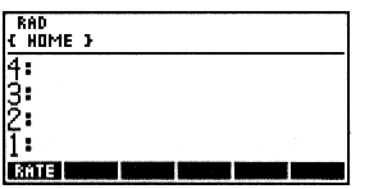


When you pressed ← RAD, the word RAD appeared near the upper-left part of the display. It is one of several **annunciators** that appear at the top of the display when the HP 48 is in certain modes of operation. Another annunciator, α, appears when you press α, telling you the calculator is in Alpha-entry mode.

The RAD annunciator tells you that trigonometric functions will be interpreted and expressed in radians. (Don't do it now, but when you want to leave Radians mode and go back to Degrees mode, you simply press ← RAD again, at which point the annunciator turns off.)

Step 7: Select the VAR menu.

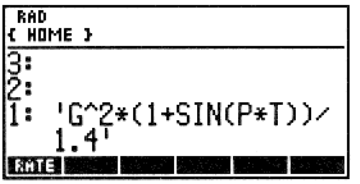
VAR



When you pressed VAR, the VAR menu appeared at the bottom of the display. A **menu** is a temporary definition of the top six, white keys on the keyboard.

Step 8: Recall the variable *RATE* to the stack. RATE is the first menu label in the VAR menu, so to execute ↱ RATE, you press ↱ followed by the first white key on the keyboard (the key with the white letter A next to it).

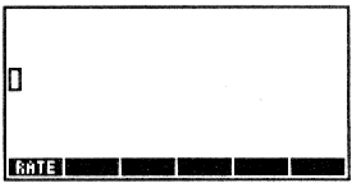
↱ RATE



Now use the EquationWriter application to build the integral.

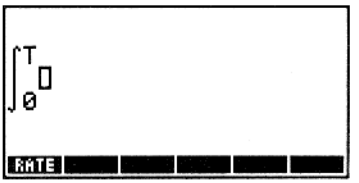
Step 9: Select the EquationWriter application.

↰ EQUATION



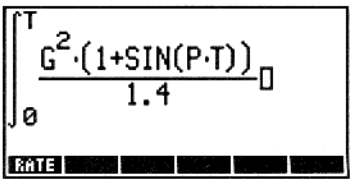
Step 10: Set the limits of integration to be 0 (zero) and *T*. The ∫ key is the blue, right-shifted definition of the COS key.

↱ ∫ 0 ▶ T ▶



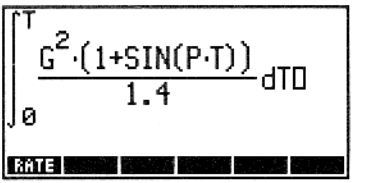
Step 11: Insert the integrand into the integral. ↱ RCL is located over the STO key and, in this context, recalls the object from level 1 into the EquationWriter application. (It takes a few seconds to display the integral.)

↱ RCL



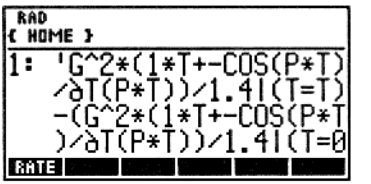
Step 12: Add the variable of integration, *T*.

▶ T



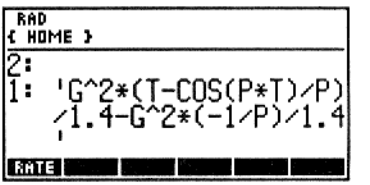
Step 13: Integrate. The EVAL key evaluates an object, which in this case is an integral.

EVAL



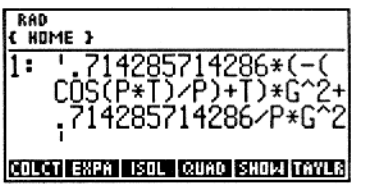
Step 14: Evaluate it further to substitute the limits into the result.

EVAL



Step 15: Select the ALGEBRA menu, and then simplify the equation by collecting like terms. The ALGEBRA key is the left-shifted definition of the 9 key; COLCT is the first menu label in the ALGEBRA menu.

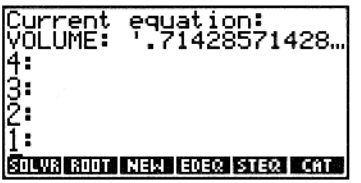
◀ ALGEBRA COLCT



Step 16: Name the expression *VOLUME*. When you press ↰ SOLVE below, a menu with NEW will appear at the bottom of the display. NEW is the third menu label, so to execute it you press the third white key on the keyboard.

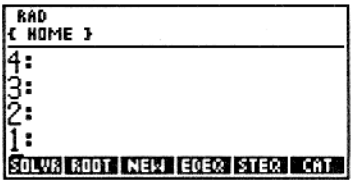
Also, note that the letters V, O, L, U, M, and E in the key sequence below are Alpha-entry characters—however, you don't need to press the α key to enter them because NEW automatically puts the calculator into Alpha-entry mode.

↰ SOLVE NEW VOLUME ENTER



Step 17: Store the given values for G (12) and P (4.2).

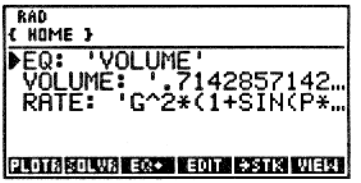
12 ENTER ' G STO
4.2 ENTER ' P STO



When you pressed the first key in the sequence above, did you notice what happened to the stack? The stack moved up in the display and the text you typed appeared on the line below level 1. That line is called the **command line** .

Step 18: Select the Equation Catalog. The **Equation Catalog** contains each of the algebraic objects you've created and named (that is, stored in a variable). When you named the rate and volume expressions, they were automatically added to the Equation Catalog. This catalog lets you choose the equation or equations to plot, solve, or edit.

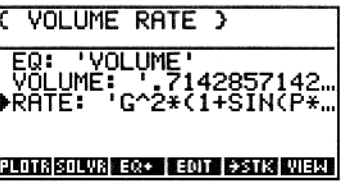
CAT



One of the equations listed in the Equation Catalog is *EQ*. *EQ* is a name the machine automatically gives to the **current equation** — the equation it uses in the Plot or HP Solve applications. Since the calculator uses this name for a special purpose, it is called a **reserved variable name.**

Step 19: Create a list containing the rate and volume equations so that you can plot them both at the same time.

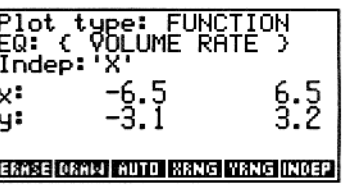
▼ EQ+ ▼ EQ+



At the top of the display is a list you just created using the EQ+ key; it contains the names of the rate and volume expressions. A **list** is another HP 48 object type and is delimited by { }. When you exit the catalog by selecting an application from the menu, this list will become the current equation. (You can also exit a catalog by pressing ATTN.)

Step 20: Select the Plot application.

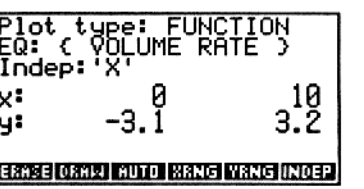
PLOTR



Note the status information regarding the current plot type, the current equation (*EQ*), the independent variable, and the *x*- and *y*-axis ranges to be displayed in the plot. When you select an application, the HP 48 usually displays status information relating to the current values and settings for that application.

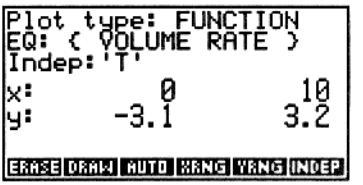
Step 21: Change the *x*-axis display range to 0 to 10 hours.

0 SPC 10 XRNG



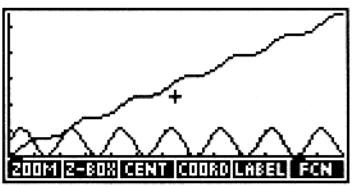
Step 22: Set T as the independent variable.

▯ T INDEP



Step 23: Plot the two equations, autoscaling the y-axis.

AUTO



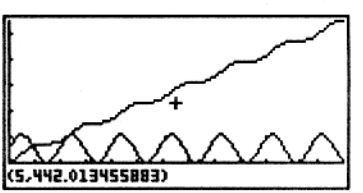
When the graph appears, you are in the **Graphics environment**. The term environment is used in this manual to describe a special state of the calculator in which the keyboard has been redefined and a specialized set of tasks alone can be performed.

Look at the plot for a moment. Notice the regular cycles of the production rate and the cumulative increase in the volume produced.

Before you go on to solve for T at 1,000 gallons, gather some other information from these plots. For instance, find the volume at 6 hours of production.

Step 24: Display the cursor coordinates. The two numbers that appear in parentheses in place of the menu labels represent the x- and y-coordinates, respectively, of the cursor.

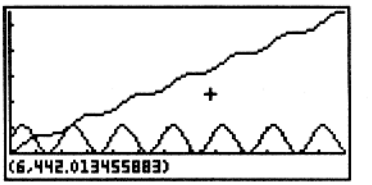
COORD



The two numbers that appear are 5 (the x-coordinate) and 442.013455883 (the y-coordinate).

Step 25: Press the ▶ key repeatedly until the left cursor coordinate increments to exactly 6. Each time you press ▶, the graphics cursor moves and the *x*-coordinate value changes. Eventually you will see exactly 6 for its value. If you move the graphics cursor too far, you can use ◀ to back up.

▶ ▶ . . .



The ▶ key is one of the four **cursor keys** (▶, ▼, ◀, and ▲) on the keyboard.

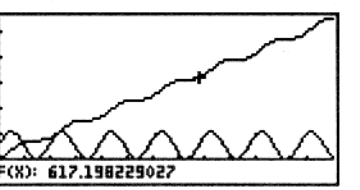
Step 26: Return the menu labels to the display and select the FCN menu. The operations in the GRAPHICS FCN menu let you analyze the mathematical behavior of the plotted function.

+ FCN



Step 27: Calculate the volume at 6 hours of production. Some menus have more than six entries, as in this case. The NXT key takes you to the next page of a menu with more than one page. F(X), which is on page 2 of the FCN menu, finds the value of the function at the *x*-coordinate defined by the cursor location.

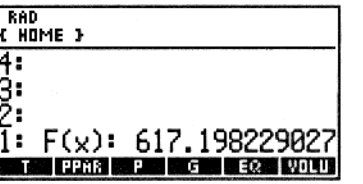
NXT F(X)



At 6 hours, the total amount produced is approximately 617 gallons.

Step 28: Restore the FCN menu, exit the Graphics environment, and select the VAR menu.

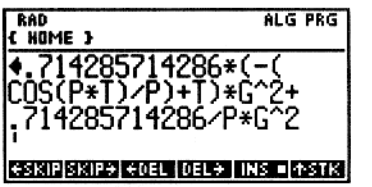
[ATTN] [ATTN] [VAR]



Pressing [ATTN] ([ON]) is the common way of exiting special displays and environments.

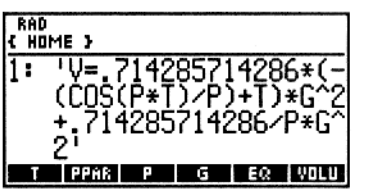
Step 29: Display the volume expression for editing. The [◄] [EDIT] key copies an object into the command line for editing.

VOLU [◄] [EDIT]



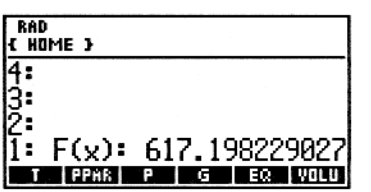
Step 30: Make the expression into an equation by adding the left-side variable *V* for *V*olume (for use later when you solve for *T*, given a *V* of 1,000). [=] is the left-shifted definition of the [0] key.

[▶] V [◄] [=] [ENTER]



Step 31: Store the edited equation back into *VOLUME*.

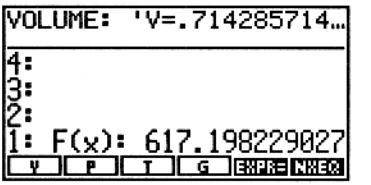
[◄] VOLU



Pressing [◄] before a VAR menu key stores the object in level 1 in that variable, overwriting the previous contents. Pressing [►] before a VAR menu key recalls the contents of that variable to the stack.

Step 32: Select the SOLVR menu in the HP Solve application. The HP Solve application lets you solve for an unknown variable in an equation.

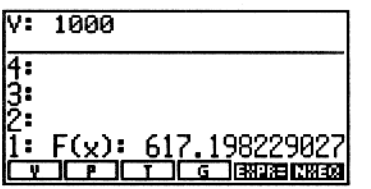
◀ SOLVE SOLVR



Notice at the top of the display that the volume equation is shown. When you enter the HP Solve application, the current equation is displayed for your reference.

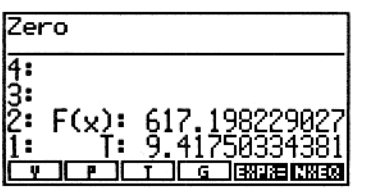
Step 33: Since P and G have values already, you only need to give V a value to solve for T. Enter 1000 as the value for V.

1000 V



Step 34: Now solve for the time T that it takes to produce 1,000 gallons of refined product. In the HP Solve application, pressing ◀ before a menu key solves for that variable, given the values stored in the other variables.

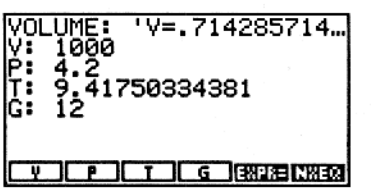
◀ T



With the given values for the process variables, it takes approximately 9.4 hours to produce 1,000 gallons of oil. (The message Zero at the top of the display means an exact root was found.)

Step 35: What were the values of the process variables? Review the equation and variables to find out.

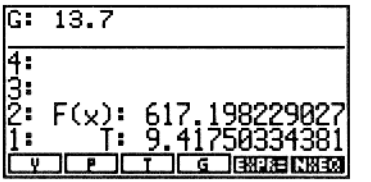
◀ REVIEW



If you improve the process so that the process variable *G* has a value of 13.7, what is the new time required to produce 1,000 gallons of product?

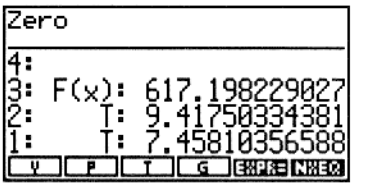
Step 36: Enter the new value for *G*.

13.7 G



Step 37: Solve for the new time.

◀ T

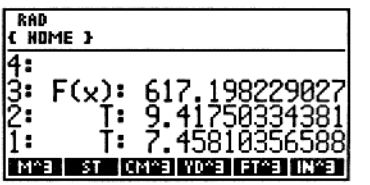


If you improve the process as described, it would take about 7.46 hours to produce 1,000 gallons — almost a 2-hour savings.

As the last part of this example, assume the 1,000 gallon tank (US gallons) is full and you want to add that volume to another, larger tank that already contains 2,500 liters of oil. What is the total volume of oil in US gallons when you combine the volumes from the two tanks?

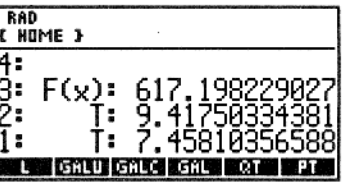
Step 38: Select the UNITS VOL menu. The Units application contains submenus (like the VOL menu) that themselves contain different types of units. These unit menus can be used to append units to numbers (creating **unit objects**) and to do math with differing units quickly and easily.

◀ UNITS VOL



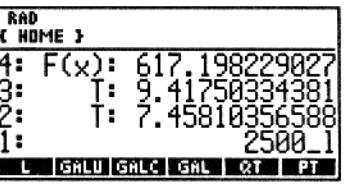
Step 39: Go to page 2 of the menu where labels for liters (L) and gallons (GAL) are found.

NXT



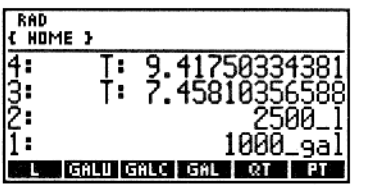
Step 40: Enter the volume with the units that need to be converted.

2500 L



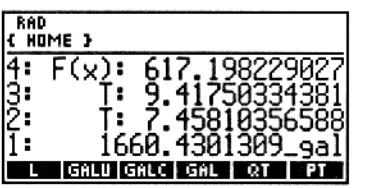
Step 41: Enter the volume with the units you want in the final answer.

1000 GAL



Step 42: Add them. Notice the unit conversion takes place automatically.

+



The total volume of the combined tanks of oil is approximately 1,660.43 US gallons.

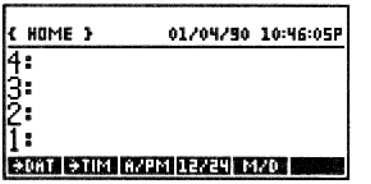
At this point, you don't need to have the HP 48 in Radians mode any longer, so press ↰ RAD to turn off the RAD annunciator and return the calculator to Degrees mode (its default state). Also, you can clear the stack by pressing ↱ CLR.

Setting the Time and Date

Before going any further, set your system time and date:

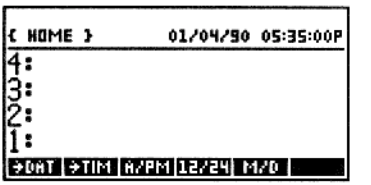
Select the TIME SET menu.

◀ TIME SET



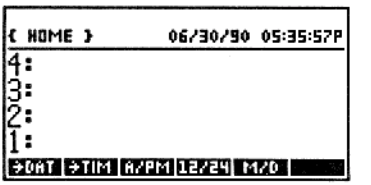
Enter the current time in the form of hours/minutes/seconds (*HH.MMSS*) using 24-hour format. (This example assumes the time is 5:35 pm. Substitute the actual time.)

17.3500 →TIM



Enter the current date using month/day/year format (*MM.DDYYYY*). (This example assumes the date is June 30, 1990. Substitute the actual date.)

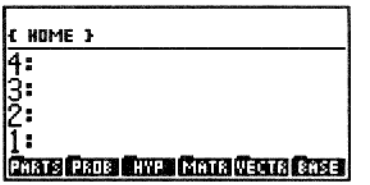
6.301990 →DAT



The date and time you just set are displayed in the upper-right corner of the display.

Now that the time and date are set, switch to the MTH menu. (Even though the ticking clock disappears, the current date and time are still being kept by the calculator.)

MTH



More information on system time, date, and alarms is contained in chapter 24, "Time, Alarms, and Date Arithmetic."

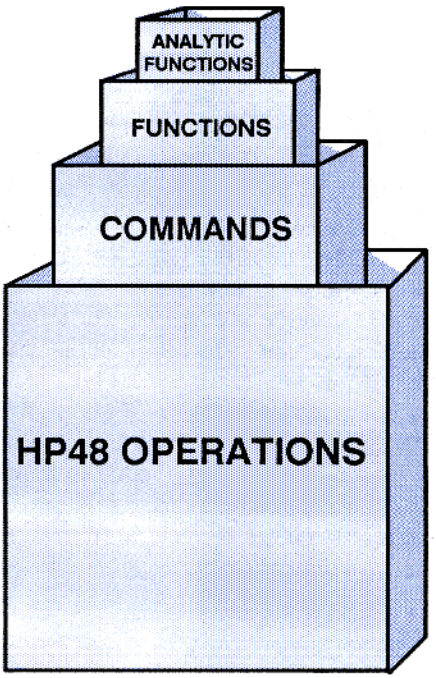
Congratulations! You've completed a brief tour of some of the features the HP 48 offers and you've set your system time and date. There's a lot more power to discover as you work through the rest of this manual, but you're off to a good start. The next material in this chapter will help you understand other chapters better. Read the rest of this chapter before going on to chapter 2.

Classifying HP 48 Operations

An HP 48 *operation* is any action the calculator can perform. In the example in this chapter you executed numerous operations. (Every time you pressed a key, you executed an operation.) Later on, it will be helpful to know if an operation can be included in a program, if it can be included in an algebraic object, and if it has an inverse or derivative. Therefore, operations are classified by category throughout this manual:

- **Operation:** any action built into the calculator 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 just a command—it can be included in a program, but it cannot go in an algebraic and has no derivative or inverse.

For handy reference, the operation index in the back of volume II tells you how each operation in the machine is classified. Also, throughout the manual, HP 48 activities are referred to as operations, commands, functions, or analytic functions where appropriate.

Where to Go from Here

Now that you've worked through chapter 1, you're ready for a more detailed look at the HP 48. For your convenience, this two-volume manual is organized into five parts. You should read the rest of part 1: Building Blocks to get a thorough understanding of the HP 48 basics. Then, depending on the types of problems you want to use the HP 48 to solve, refer to the appropriate chapters in part 2: Hand Tools, part 3: Power Tools, part 4: Programming, or part 5: Printing, Data Transfer, and Plug-Ins. The table of contents at the beginning of this manual contains a listing of all the chapters with their main topics.

If, at any time, you desire specific information about an operation in the calculator, you can refer to the operation index at the back of volume II for a description of the operation and a page reference for more information. There is also a large subject index at the back of volume II to help you find specific subjects of interest.

2

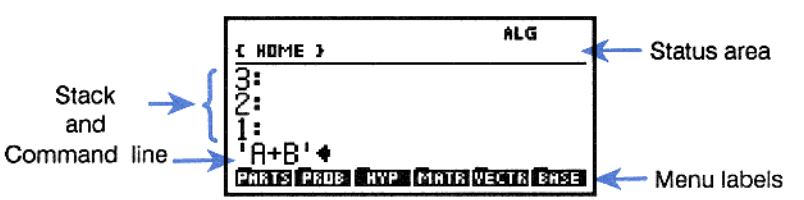
The Keyboard and Display



Chapter 2 describes the keyboard and display in detail. Understanding the information in this chapter is part of the foundation you need to effectively use your HP 48.

Organization of the Display

For most operations, the display is divided into three sections. This configuration is called the *stack display*.



- The top section displays messages and status information.
- The middle section displays the *stack* and *command line*.
- The bottom section contains six *menu labels* that describe the current function of the six menu keys at the top of the keyboard.

The Stack

The stack is a sequence of storage locations for seeing and manipulating data. It is arranged in levels — level 1, 2, 3, etc. In general, you enter data onto the stack and then execute commands to manipulate the data.

The stack display shows level 1 and as many additional levels as possible. As you enter data, the stack grows to accommodate more information. The number of levels is limited only by the amount of memory available.

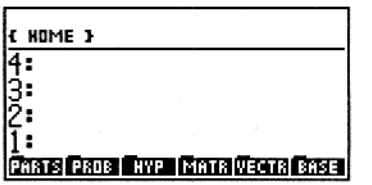
The Command Line

The command line is used to key in and edit text. If you type in more than 21 characters, information scrolls off the left side of the display, and an ellipsis (. . .) appears to tell you there is more information "to the left."

Example: Using the Stack and Command Line. The following keystrokes show how the stack and command line are used to calculate $\sqrt{15+23}$.

Clear the stack.

[→] [CLR]



Type 15 into the command line.

15



Enter the contents of the command line into level 1.

[ENTER]



Enter 23 onto the stack. Entering a second number "pushes" the 15 to level 2.

23 [ENTER]



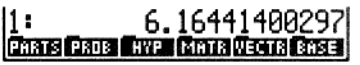
Add the two numbers. The two values are removed from the stack and their sum is returned to level 1.

[+]



Now calculate the square root of the value in level 1.

[√x]



Keying In Numbers

Numbers are keyed into the command line. If you make a typing mistake, use the backspace key ([←]) to erase it, and then retype it.

To key in a negative number, type the digits of the number and then press [+/−].

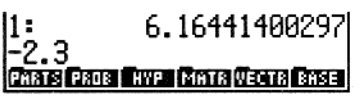
To key in a number as a mantissa and an exponent:

1. Key in the mantissa. If it is negative, change its sign by pressing [+/−].
2. Press [EEX].
3. Key in the exponent. If it is negative, press [+/−].

Calculate $10^{-2.3}$.

Enter −2.3.

2.3 [+/−]



Complete the calculation.

[↶] [10^x]



Calculate $\frac{-8.09\times10^{12}}{4.81\times10^{-6}}$.

Key in the two numbers.

8.09 [+/−] [EEX] 12 [ENTER]
4.81 [EEX] 6 [+/−] [ENTER]



Divide the two numbers.

÷

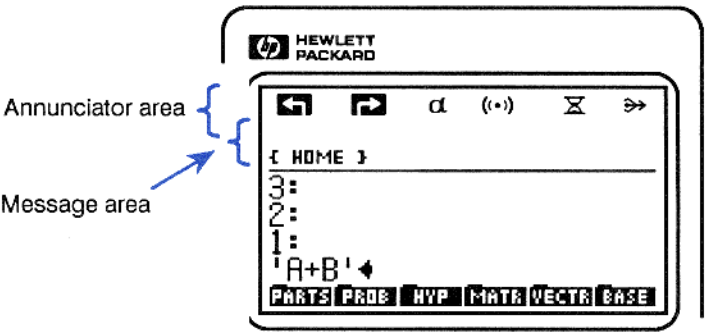
```
1:    -1.68191268191E18
PARTS PROB HYP MATR VECTR BASE
```

The Status Area, Annunciators, and Messages

The status area displays:

- Annunciators that indicate the current 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 similar to files in a file cabinet; they are covered in chapter 7.
- Messages that inform you when an error has occurred, or that provide other information to help you use the calculator more effectively.

The first six annunciators in the following table appear at the very top of the display. The other annunciators and the directory path share their "territory" with messages. When you clear a message, the directory path and any active annunciators reappear.



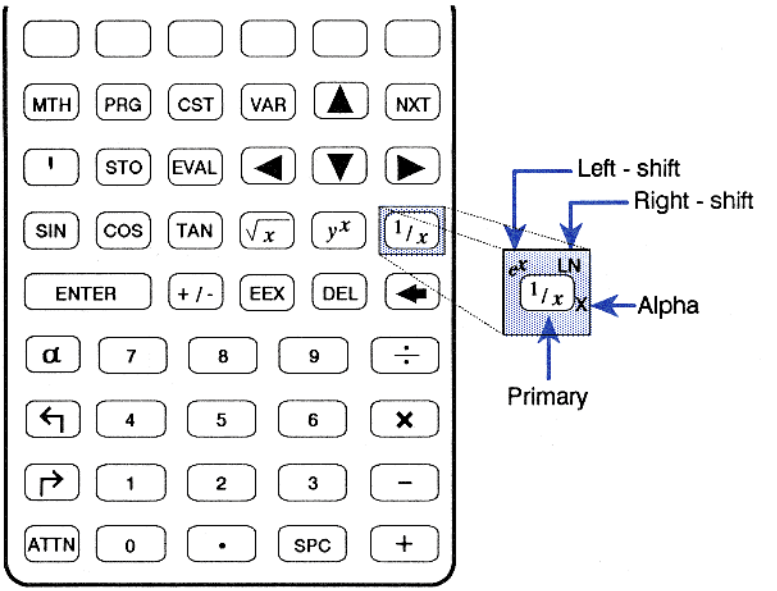
Annunciators

Symbol	Meaning
◧	Left-shift is active (you pressed [↰]).
◨	Right-shift is active (you pressed [↱]).
α	The alpha keyboard is active (the keys have their alpha assignments).
((•))	Alert; an appointment has come due, or a low battery condition has been detected. See the message in the status area for information. (If there is no message displayed, turn the calculator off and back on. A message describing the cause of the alert should appear.)
⧖	Busy, not ready to process new input. The calculator can take up to 15 keystrokes in queue while busy and then process them when free.
⇛	Transmitting data to an external device.
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 [↰][USR].
ALG	Algebraic-entry mode is active.
PRG	Program-entry mode is active.

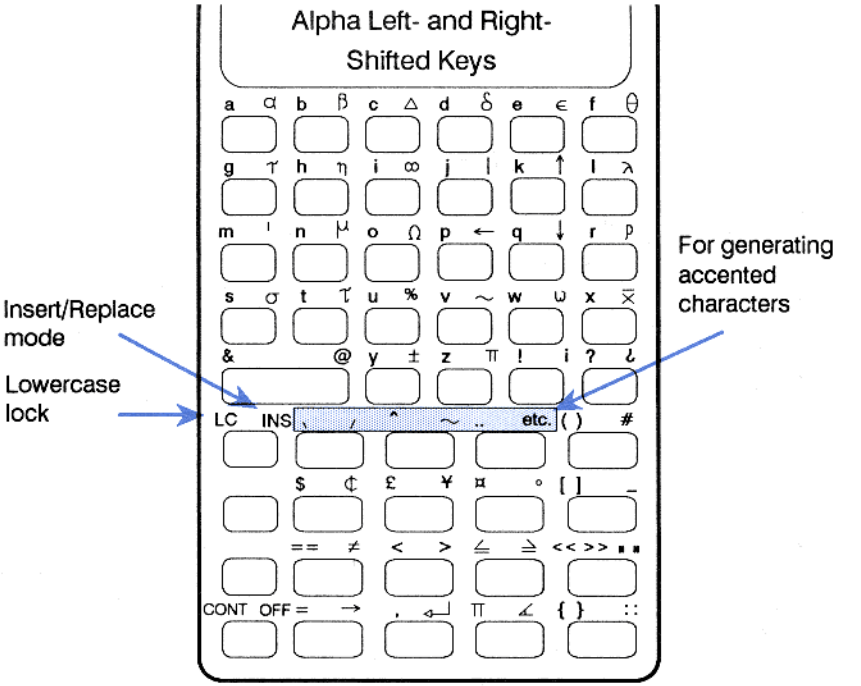
Organization of the Keyboard

The HP 48 keyboard has six levels, each containing a different set of keys:

- *The primary keyboard*, which is represented by the labels on the key faces. For example, [+], [7], [ENTER], [TAN], and [▲] are all keys on the primary keyboard.
- *The left-shift keyboard*, which is activated by pressing the [↰] key on the primary keyboard. A left-shift key is orange and located above and to the left of its associated primary key. To execute ASIN, for example, you press the [↰] key followed by the associated [SIN] key.
- *The right-shift keyboard*, which is activated by pressing the [↱] key on the primary keyboard. A right-shift key is blue and located above and to the right of its associated primary key. To execute →NUM, for example, you press the [↱] key followed by the associated [EVAL] key.
- *The alpha keyboard*, which is activated by pressing the [α] key on the primary keyboard. An alpha key is white and located to the right of its associated primary key. Alpha keys are all capital letters. To generate N, for example, you press [α] followed by the associated [STO] key. When the alpha keyboard is active, the number pad retains its numerical operations.
- *The alpha left-shift keyboard*, which is activated by pressing [α] and then [↰] on the primary keyboard. Alpha left-shift characters are primarily lowercase letters, along with some special characters. To type n, for example, you press [α], then [↰], and then [STO]. (Alpha left-shift characters are not shown on the keyboard.)
- *The alpha right-shift keyboard*, which is activated by pressing [α] and then [↱] on the primary keyboard. Alpha right-shift characters are Greek letters and other special characters. To generate λ, for example, you press [α], then [↱], and then [NXT]. (Alpha right-shift characters are not shown on the keyboard.)



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 the alpha left-shift and alpha right-shift keys.



The Shift Keys

When you press [←] (left-shift) or [→] (right-shift) to access the shifted operations printed above the primary keys, the ↰ or ↱ annunciator turns on to indicate that left-shift or right-shift is active.

To cancel left-shift or right-shift, press the same shift key again. Pressing the other shift key overrides the first shift with the second one.

The Alpha Keyboard

To activate the *alpha keyboard* and turn on the α annunciator, press [α]. This activates Alpha-entry mode for one character. For example, pressing [α] then [SIN] types S.

You can press and hold down [α] while you type several letters in a row.

The primary (unshifted) alpha assignments are printed on the keyboard to the lower right of each key. In addition, many keys have left- and right-shifted alpha assignments. (All uppercase letters are unshifted, while their lowercase counterparts are left-shifted.)

Alpha Lock. Press [α] twice in a row to lock Alpha-entry mode. Alpha-entry mode remains active until you press [α] again or [ENTER].

Lowercase Alpha Lock. You can lock lowercase Alpha-entry mode by pressing [α][←][α]. Then, when you press [α] (or [α][α]), you get lowercase alpha characters and must use [←] to get uppercase. Try it now to get a feel for how it works. ([←][DROP] will clear any extraneous characters you generate from your display.)



Note

Keystroke examples in this manual show the alpha characters without the [α] key. For example, the keystrokes for entering "HELLO" onto the stack are shown as [→]["] HELLO [ENTER]. Even though it's not shown in the keystrokes, you still must activate Alpha-entry mode in one of the ways described above before entering the alpha characters.

Generating Accented Characters

To type accented characters, use one of the five accent marks (`, ´, ˜, ^, and ¨) or the [etc] key in conjunction with a letter. These six keys are the alpha left- and right-shift keys associated with the primary keys [7], [8], and [9] (see the keyboard illustration on page 52).

To generate an accented character during text entry, type the letter and then the accent. During editing, position the cursor to the right of the letter and then type the accent. In each case, the letter to the immediate left is changed.

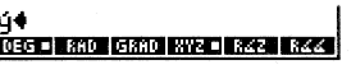
Example: Typing Accented Characters. Key in the accented character ý.

Key in the lowercase letter y. (It's the alpha left-shifted key associated with [+/−].)

y



Key in the ´ accent character. (It's the alpha right-shifted key associated with [7].)



Notice how the accent character didn't actually show up alone. Instead, it modified the original letter to form a special accented character. Press [ATTN] to clear the command line.

The [etc] key and the accent characters can also be used to generate certain special characters. The following table shows the special characters that can be generated using these keys:

Use [α] [→] [etc]		Use Any Accent Mark	
To Change:	**To:**	**To Change:**	**To:**
A	Å	C	Ç
a	å	c	ç
E	Æ	D	Ð
e	æ	d	ð
O	Ø	P	Þ
o	ø	p	þ

The Attention Key

When the HP 48 is on, [ON] becomes the [ATTN] (attention) key. Generally, [ATTN] halts the current activity:

- If the command line is present, [ATTN] cancels it.
- If a special environment is active, [ATTN] cancels it and restores the stack display.
- If a program is running, [ATTN] aborts the program.

Keying in Delimiters

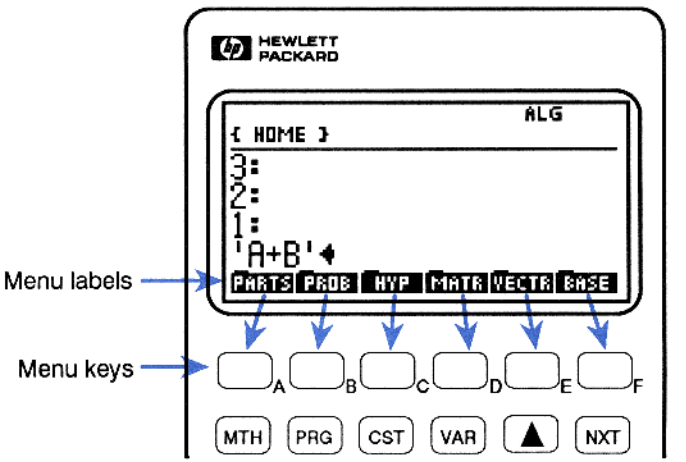
The keyboard includes certain "punctuation" characters, called *delimiters*, that identify different types of information you can enter. A piece of information is called an *object*. For example, the (real) number `46.3`, the unit value `14.7_psi` (delimited by _), the algebraic expression `'A+B+C'` (delimited by two ' marks), and the vector `[ 2,3,4 ]` (delimited by `[ ]`) are different types of objects.

When an object has both opening and closing delimiters (for example, the square brackets around vectors), the delimiter key types both delimiters and then positions the cursor so that you can type the object between them.

Objects and delimiters are covered in chapter 4.

Menus

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



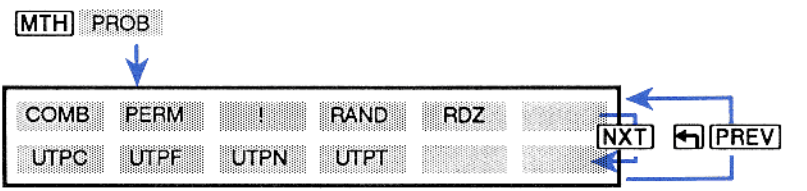
Some menus have multiple sets of labels or *pages*. You can cycle through multiple menu pages using the NXT key.

If a menu label has a bar over the left side, it selects another menu.

Displaying Menus

Many of the keys on the HP 48 keyboard display menus. Some of these menus contain more than six entries. Press NXT to see the next "page" of labels and ◀ PREV to return to the previous page. Pressing NXT repeatedly eventually redisplays the first page. To go directly to the first page from whatever page you're on, press ▶ PREV.

In the following illustration of the MTH PROB menu, notice how NXT and ◀ PREV work:



Example: A Menu of Menus. The MTH menu is a menu that contains other menus.

Press MTH and notice that each menu key in the MTH menu has a bar over its left side and, therefore, calls another menu.

MTH

PARTS PROB HYP MATR VECTR BASE

Press one of the MTH menu keys and practice using NXT and ◀ PREV.

With a few exceptions, 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. (The special menus that act a little differently are explained in later chapters.)

Example: Using a Menu. Calculate 7!. The factorial function is located in the MTH PROB (math probability) menu.

Key in the 7 and execute the function.

7 [MTH] PROB !



Switching to the Last Menu

There may be times when you are working primarily with a particular menu, but need to use commands in another menu. For example, you may need to leave briefly the third page of the STAT 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 [→] [LAST MENU] (found over the [3] key) returns you to that menu. Menus of menus (such as the MTH menu) aren't stored as the last menu.

Display Modes

The *display mode* controls the format the HP 48 uses to display numbers. (Regardless of the current display mode, a number is always stored as a signed, 12-digit mantissa with a signed, three-digit exponent.) The keys for setting the display mode are located in the MODES menu ([←] [MODES]). A box in the menu label indicates that mode is active—for instance, STD■ means that Standard mode is active.

Display Modes

Keys	Programmable Command	Description
↩ MODES (pages 1 and 4):		
STD	STD	*Standard mode*; displays numbers using full precision. All significant digits to the right of the decimal point are shown, up to 12 digits.
FIX	FIX	*Fix mode*; displays numbers rounded to a specified number of decimal places. Real numbers on the stack are displayed with digit separators—commas (for period fraction mark) or periods (for comma fraction mark).
SCI	SCI	*Scientific mode*; displays a number as a mantissa (with one digit to the left of the decimal point) and an exponent.
ENG	ENG	*Engineering mode*; displays a number as a mantissa followed by an exponent that is a multiple of 3.
FM,		Switches fraction mark (the character that separates the integer and fractional part of the number) between period and comma. A box in the label indicates a comma fraction mark.

FIX, SCI, and ENG require a numerical argument:

- Fix mode requires the number of decimal places.
- Scientific mode requires the number of decimal places in the mantissa.
- Engineering mode requires the number of mantissa digits to be displayed after the first significant digit.

Example: Changing Display Modes.

Display 12345.6789 rounded to two decimal places.

12345.6789 ENTER
◀ MODES 2 FIX



Switch to scientific notation with a 5-decimal place mantissa.

5 SCI



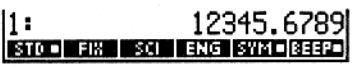
Switch to engineering notation with a 4-digit mantissa (3 digits after the first digit).

3 ENG



Return to Standard display mode.

STD



3

The Stack and Command Line



The stack is a series of levels, each acting as a storage location for data. As new data is entered, old data on the stack is "bumped" to higher levels. The display lets you see at most four levels of the stack at one time, but there can be many more levels in memory. The number of levels on the stack is limited only by the amount of available memory.

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.

This chapter covers:

- Using the stack for calculations.
- Viewing and editing the contents of the stack.
- Using the command line.

Using the Stack for Calculations

An Overview

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

- Commands that require *arguments* (objects the command acts upon) take their arguments from the stack. (Therefore, the arguments must be present *before* you execute the command.)
- The arguments of a command are removed from the stack when the command is executed.
- Results are returned to the stack so that you can use them in other operations.

One-Argument Commands

One-argument commands act on the argument (object) in level 1 and return the result to level 1.

Example. Use the one-argument commands LN ([→] [LN]) and INV ([1/x]) to calculate $\frac{1}{\ln 3.7}$.

First, calculate ln 3.7. You do not need to press [ENTER] before executing the command.

3.7 [→] [LN]



Calculate the inverse of the result.

[1/x]



Two-Argument Commands

Two-argument commands act on the the arguments (objects) in levels 1 and 2, and return 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 all examples of two-argument commands.

Examples. The following keystrokes show examples of two-argument commands.

Calculate 85 − 31.

85 ENTER
31 −



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

45 $\sqrt{x}$
12 ×



Calculate $\frac{5+9i}{4}$. (Parentheses are the delimiters for complex numbers.)

← ( ) 5 SPC 9 ENTER
4 ÷



Calculate $4.7^{2.1}$

4.7 ENTER
2.1 y^x



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

2401 ENTER
4 → $\sqrt[y]{}$



Using Previous Results (Chain Calculations)

Chain calculations involve more than one operation. The stack is especially useful for chain calculations because it retains intermediate results.

Examples. The following examples demonstrate using the stack for chain calculations.

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

12 ENTER 3 +
7 ENTER 9 +



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

×



Now, calculate $23^2 - (13 \times 9) + {}^5/_7$.

First, calculate 23^2 and the product of 13×9.

23 ◀ x^2
13 ENTER 9 ×



Subtract the two intermediate results and calculate ${}^5/_7$.

−
5 ENTER 7 ÷



Add the two results.

+



Swapping Levels 1 and 2

The SWAP command (◀ SWAP) exchanges the contents of levels 1 and 2. (▶ performs a SWAP when no command line is present.) SWAP is useful with commands where the order is important, such as − (−), / (÷), and ^ (y^x).

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

Calculate $\sqrt{13 + 8}$:

13 ENTER 8 + $\sqrt{x}$



Enter 9 and swap levels 1 and 2.

9 ↶ SWAP



Divide the two values.

÷



Clearing the Stack

The DROP command (↶ DROP) removes the object in level 1. (◆ performs DROP when no command line is present.) The remaining items drop down one level.

The CLEAR command (↷ CLR) clears the entire stack.

Recovering the Last Arguments

Executing LASTARG (↷ LAST ARG) places the arguments of the most recently executed command on the stack so that you can use them again.

Example. Use ↷ LAST ARG to calculate ln 2.3031 + 2.3031.

First, calculate ln 2.3031.

2.3031 ↷ LN



Retrieve the argument of LN. (LAST ARG is the blue, right-shifted assignment over the 2 key.)

↷ LAST ARG



Add the two numbers.

+



Example. LASTARG is particularly useful for more complicated arguments. The following keystrokes calculate:

$$\frac{[1\ 1\ 1]\times[1\ 2\ 1]}{[1\ 1\ 1]\cdot[1\ 2\ 1]}$$

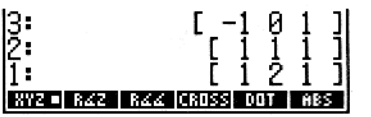
Enter the two vectors and calculate the cross product. The CROSS command is located in the VECTR menu (within the MTH menu).

[↰][ ] 1 [SPC] 1 [SPC] 1 [ENTER]
[↰][ ] 1 [SPC] 2 [SPC] 1
[MTH] VECTR CROSS



Retrieve the two vectors.

[↱] LAST ARG



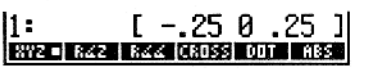
Calculate the dot product.

DOT



Divide the cross product by the dot product.

[÷]



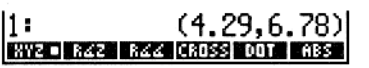
Duplicating Level 1

The DUP command ([PRG] STK [NXT] DUP) duplicates the contents of level 1 and pushes the other stack contents up one level. ([ENTER] performs DUP when no command line is present.)

Example. Calculate $(4.29,6.78i) + (4.29,6.78i)^4$.

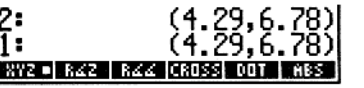
Enter the first complex number.

[↰][( )] 4.29 [SPC] 6.78 [ENTER]



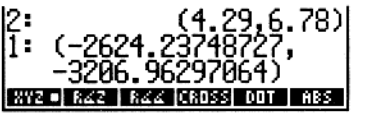
Duplicate the number.

ENTER



Raise the complex number to the 4th power.

4 y^x



Add the result to the original complex number.

+



Displaying Objects for Viewing or Editing

As introduced in chapter 1, objects are the basic items of information that the HP 48 uses and manipulates. You cannot always see all of the objects on the stack—you can only see the beginning of large objects, and you cannot see stack contents that have changed levels and scrolled off the display.

There are three ways to view or edit objects:

- By using the ▼ key, which lets you view and edit the level 1 object *in the most appropriate environment*. (The → ▼ key lets you view and edit the object stored in a variable.)
- By using the ← EDIT key, which lets you edit an object on the stack *in the command line*. (The → VISIT key lets you edit the object stored in a variable.)
- By using the Interactive Stack, which lets you view and edit all levels of the stack.

Whenever an object is copied into the command line for viewing or editing, these things happen:

- The EDIT menu is displayed, which provides operations that make it easier to edit large objects.
- 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.

Viewing and Editing an Object

- Pressing ◀ EDIT copies the object in level 1 into the command line, where you can see the entire object and edit it, if necessary.
- Pressing ▼ does the same thing as ◀ EDIT, except not all objects are copied into the command line—matrices are copied into the MatrixWriter environment, and algebraic objects and units are copied into the EquationWriter environment.
- Pressing ▶ VISIT with a stack-level number as an argument copies the object in that level into the command line for editing. For example, 3 ▶ VISIT copies the object in level 3 into the command line.
- Pressing ▶ ▼ with a stack-level number as an argument does the same thing as ▶ VISIT, except not all objects are copied into the command line—matrices are copied into the MatrixWriter environment, and algebraic objects and units are copied into the EquationWriter environment.

Press ENTER to end the edit session and return the edited object to its original location. Press ATTN to end the session without change.

Viewing and Editing the Contents of a Variable

- Pressing ▶ VISIT with a variable name for an argument lets you edit the *contents* of that variable in the command line. For example, 'EX1' ▶ VISIT copies the contents of the variable *EX1* into the command line.
- Pressing ▶ ▼ with a variable name argument does the same thing, except not all objects are copied into the command line—matrices are copied into the MatrixWriter environment, and algebraic objects and units are copied into the EquationWriter environment.

Press ENTER to end the edit session and return the edited object to its original location. Press ATTN to end the session without change.

The EDIT Menu

If a command line is present, the EDIT menu is displayed when you press ◀ EDIT. Also, the EDIT menu is 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*.

EDIT Menu Operations

←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 box in the menu label indicates Insert mode is active.
↑STK	Activates the Interactive Stack (see page 70).

Example: Viewing and Editing Objects. Enter several objects on the stack, and then see and edit them as described.

Store the vector [1 2 3] in a variable named *A*, and then recall *A* to the stack.

◀ [ ] 1 SPC 2 SPC 3 ENTER
' A STO VAR ' A ENTER



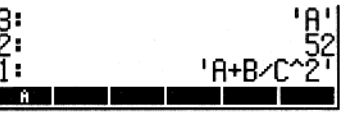
Enter a real number on the stack.

52 ENTER



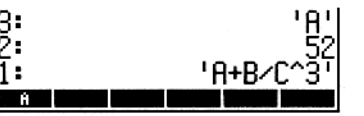
Enter an algebraic object on the stack.

← EQUATION A + B ÷
C y^x 2 ENTER



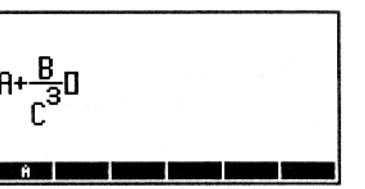
Edit the algebraic object to make C^2 become C^3.

← EDIT SKIP→ ◀ ◀
DEL 3 ENTER



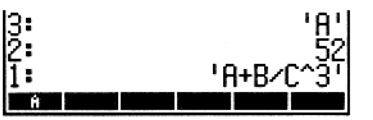
See the edited equation in the EquationWriter environment.

▼



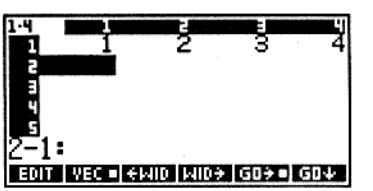
Return to the stack.

ATTN



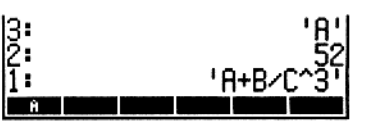
Edit the contents of the variable *A* using the MatrixWriter application.

' A ↱ ▼
▶ ▶ ▶ 4 ENTER



Save the edited vector and return to the stack.

ENTER



Now use the stack from this example to practice more editing on your own. Try to exercise all the viewing and editing features previously described in this section.

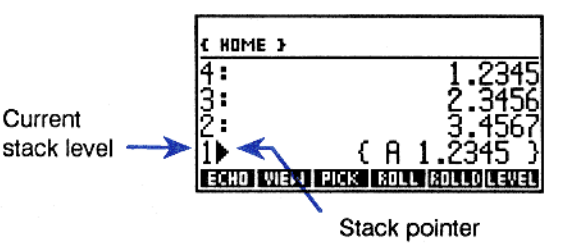
The Interactive Stack

The normal stack display is a "window" that shows level 1 and as many higher levels as will fit. The *Interactive Stack* lets you:

- 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.

The Interactive Stack is a special environment in the HP 48 where the keyboard is redefined for a specific set of stack-manipulation operations only. You must exit the Interactive Stack before you can execute any other calculator operations (see "Exiting the Interactive Stack" on page 74).

To activate the Interactive Stack, press ▲ or ↑STK (in the EDIT menu). This turns on the *stack pointer*, which points to the *current stack level*, and displays the Interactive-Stack menu. Use ▲ and ▼ to move the stack pointer up and down the stack.



The Interactive-Stack Operations. When you select the Interactive Stack, the HP 48 keyboard is redefined and the Interactive-Stack menu appears. If a command line is present when you select the Interactive Stack, just the ECHO key appears in the menu.

Most of the operations in the menu have equivalent programmable commands, which are described at the end of this chapter.

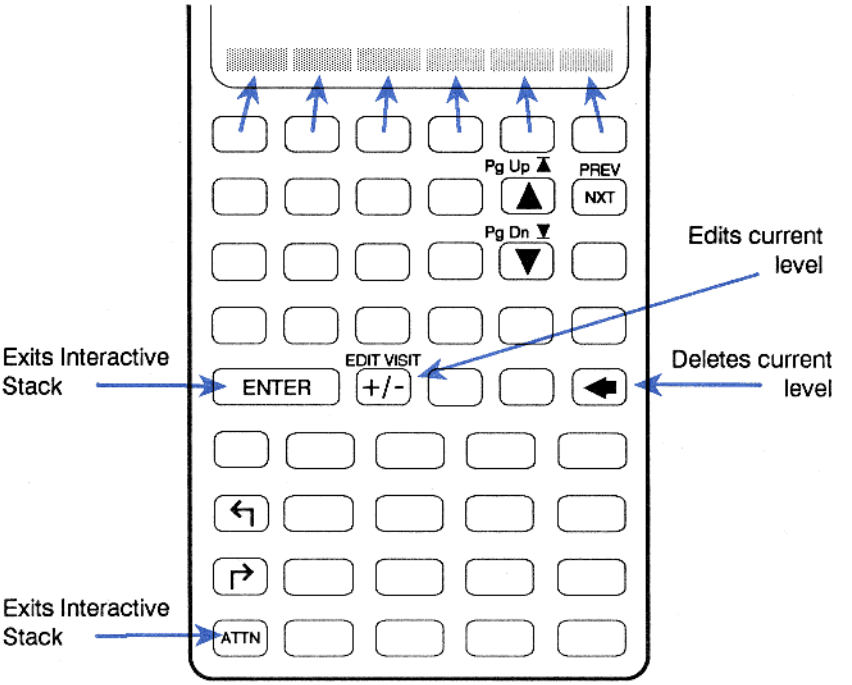
Interactive-Stack Operations

ECHO	Copies the contents of the current stack level into the command line at the cursor position.
VIEW	Used for viewing the object in the current level (see "Viewing Objects," following this table).
↱ VIEW	Used for viewing the object specified by name or stack-level number (see "Viewing Objects," below).
PICK	Copies the contents of the current level to level 1 (equivalent to *n* PICK).
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 stack level (equivalent to *n* →LIST).
DUPN	Duplicates levels 1 through the current stack 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	Drops levels 1 through the current stack level (equivalent to *n* DROPN).

Interactive-Stack Operations (continued)

KEEP	Clears all the stack levels above the current level.
LEVEL	Enters the current stack level number into level 1.
▲	Moves the stack pointer up one level. When prefixed with ↰, moves the stack pointer up four levels (↰ PgUp in the following keyboard illustration); when prefixed with ↱, moves the stack pointer to the top of the stack (↱ ▲ in the following keyboard illustration).
▼	Moves the stack pointer down one level. When prefixed with ↰, moves the stack pointer down four levels (↰ PgDn in the following keyboard illustration); when prefixed with ↱, 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 ATTN to abort).
↱ VISIT	If the object in the current level is a real number, copies the object on the level specified by the integer part of that number into the command line for editing. If the object in the current level is a variable, copies the contents of that variable into the command line. Press ENTER when finished editing (or ATTN to abort).
◆	Deletes the object in the current level.
NXT	Selects the next page of Interactive-Stack operations.
ENTER	Exits the Interactive Stack.
ATTN	Exits the Interactive Stack.

The redefined keyboard looks like this:



Viewing Objects. Within the Interactive Stack, the VIEW key lets you look at an object in the most appropriate environment. For object types other than matrices, algebraics, and units, pressing VIEW is equivalent to pressing ◀ EDIT; it copies the object in the current stack level into the command line and activates the EDIT menu. Press ENTER to return the edited object to its original stack level, or ATTN to end the session without change.

When the object to be viewed is an algebraic object or unit, pressing VIEW automatically starts up the EquationWriter application and copies the object into it; when the object is a matrix, pressing VIEW starts up the MatrixWriter application and copies the object into it.

▶ VIEW also enables you to view objects in the most appropriate environment. However, like ▶ ▼ in the normal environment, ▶ VIEW takes an argument of either a stack-level number or a name. If the current level of the Interactive Stack contains a number, pressing ▶ VIEW puts the object on that level in the most appropriate

environment for viewing; if the current level contains a name, pressing [→] VIEW puts the contents of the associated variable into the best environment for viewing.

Exiting the Interactive Stack. When you're done, press [ENTER] or [ATTN] to exit the Interactive Stack and display the changed stack.

After exiting, you can cancel any changes made in the Interactive Stack by pressing [←][LAST STACK].

Example: Using the Interactive Stack. Use the Interactive Stack to edit the contents of the command line by following these general steps:

1. Position the cursor in the command line where you want the text placed.
2. Press [←][EDIT] ↑STK. (If the command line has only one line, you can press [▲] instead.) The command line and EDIT menu temporarily disappear. In their place the display shows the stack, with the *stack pointer* (▶) positioned at level 1, and the ECHO key.
3. Use [▲] and [▼] to move the stack pointer to the desired level and press ECHO to copy (echo) the object into the command line.

Put these objects on the stack.

1.2345 [ENTER]
2.3456 [ENTER]
3.4567 [ENTER]



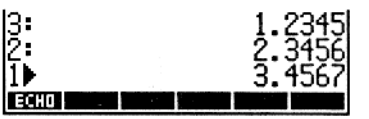
Now, create the list { A 1.2345 }. First, start the list.

[←]{ } A



Select the Interactive Stack.

[▲]



Move the pointer to level 3 and echo the object.

▲ ▲ ECHO ENTER



Enter the list.

ENTER



Using the Command Line

The command line appears when you enter or edit text (except when you are using the EquationWriter or MatrixWriter applications).

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. Use spaces, newlines (↱ ↵) or delimiters to separate text intended for distinct objects. For example, you can key in 12 SPC 34 and then press ENTER to enter them onto the stack, or + to enter them and do the addition.

When the command line is present:

- Characters are normally *inserted* at the current cursor position.
- The cursor keys ◀, ▶, ▲, and ▼ are active. (Sometimes the command line occupies more than one line on the display.) The right-shifted cursor keys (↱ ◀, ↱ ▶, etc.) move the cursor far left, far right, etc.
- ⌫ erases the character to the left of the cursor.
- DEL deletes the character at the current cursor position.

- [←][EDIT] displays the EDIT menu, which contains additional editing operations. (If the command line has only one line, [▼] also displays the EDIT menu.)
- If the command line has only one line, [▲] selects the Interactive Stack.
- [ENTER] processes the text in the command line, moving data to the stack and executing commands.

Commands can be typed into the command line for subsequent execution, as shown below.

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

Key in the following command line:

12 [SPC] 100 [→][ENTRY] [→][LOG]



Complete the calculation.

[ENTER] [−]



Entry Modes

Different entry modes make it easier for you to key in various object types. There are four entry modes.

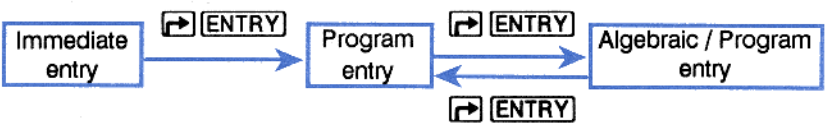
Immediate-Entry Mode. Immediate-entry mode is the default mode. In Immediate-entry mode, the contents of the command line are entered immediately when you press a function or command key (such as [+], [SIN], or [STO].)

Algebraic-Entry Mode. (Indicated by the ALG annunciator.) Algebraic-entry mode is used primarily for keying in names and algebraic expressions for immediate use. It is activated when you press ['][']. In Algebraic-entry mode, command keys act as typing aids (for example, [SIN] types SIN( ). Other commands *are* executed immediately—for example, [STO] or [←][PURGE]).

Program-Entry Mode. (Indicated by the PRG annunciator.) Program-entry mode is used primarily for entering programs and lists, and is activated when you press ◀ « » or ◀ { }. Program-entry mode is also used for command-line editing (◀ EDIT and ▶ VISIT). 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. (Indicated by the ALG *and* PRG annunciators.) Algebraic/Program-entry mode is used for keying algebraic objects into programs. It is automatically activated when you press ' while in Program-entry mode. In Algebraic/Program-entry mode, function and command keys behave as they do in Algebraic-entry mode (for example, SIN types SIN( )). Pressing a command key (for example, STO) restores Program-entry mode.

Changing Entry Modes Manually. 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 example, you can manually invoke Program-entry mode to enter 4 5 + √ into the command line, and then press ENTER to calculate $\sqrt{9}$. This key also makes it easier to edit algebraic objects in programs.

Recovering Previous Command Lines

The HP 48 automatically saves a copy of the four most recently executed command lines. To retrieve the most recent command line, press ◀ LAST CMD (found over the 3 key). Press ◀ LAST CMD repeatedly to retrieve the other saved command lines.

Other Stack Commands

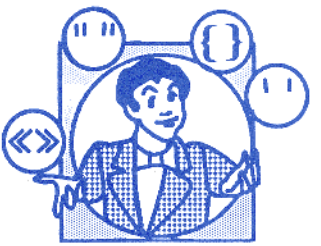
The following table describes additional commands from the PRG STK menu that are programmable and that manipulate the stack.

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).	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

Objects



The basic items of information the HP 48 uses are called *objects*. The HP 48 can store and manipulate various *types* of objects. For example, you enter a real number, matrix, or program onto the stack as a single object. Here are the HP 48 object types:

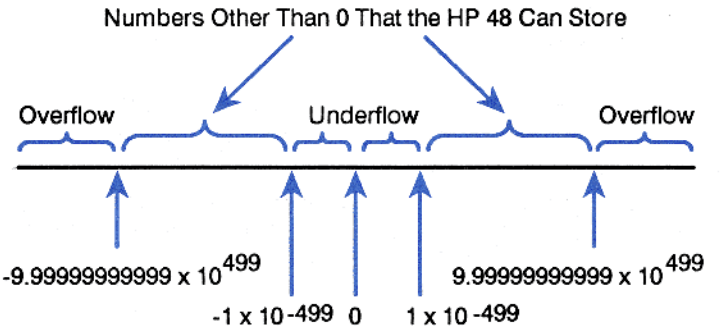
Real Numbers	Programs	Directory Objects
Complex Numbers	Strings	Backup Objects
Binary Integers	Lists	Library Objects
Arrays	Graphics Objects	XLIB Names
Names	Tagged Objects	Built-In Functions
Algebraic Objects	Unit Objects	Built-In Commands

Many HP 48 operations are the same for all object types; for instance, you use the same procedure to store a real number, matrix, or program. Some operations apply only to particular object types—for example, you cannot take the square root of a program.

This chapter introduces the HP 48 object types, shows some examples of how they are used, and covers some commands that manipulate objects. Each object type is covered in more detail in other chapters.

Real Numbers

The numbers `12`, `-3.6`, and `4.7E10` are examples of real numbers. The following illustration shows the range of real numbers the HP 48 can store.



Complex Numbers

A complex number is represented by a pair of real numbers delimited by parentheses. Complex numbers can be entered or displayed in rectangular or polar form:

- Rectangular form: $x + iy$, displayed as (x,y).
- Polar form: $(re^{i\theta})$, displayed as $(r,\angle\theta)$.

Complex numbers are also used to represent the coordinates of a point in two dimensions.

Example. Add the complex numbers $14 + 9i$ and $8 - 12i$.

If the `R∠Z` or `R∠∠` annunciator is on, press [→] [POLAR] to set rectangular coordinates mode.

Enter the complex numbers into levels 1 and 2. Use a space to separate the real and imaginary parts.

[←] [( )] 14 [SPC] 9 [ENTER]
[←] [( )] 8 [SPC] 12 [+/–] [ENTER]



Add the two values.

⊞



Complex numbers are covered in chapter 11.

Binary Integers

HP 48 binary integers are unsigned integers that represent a sequence of bits. They are delimited by a # character preceding the number and by an optional lowercase letter (h, d, o, or b) that identifies the current base. You can enter binary integers in hexadecimal, decimal, octal, or binary base. The binary base mode, set in the BASE menu (displayed by pressing [MTH] BASE), determines which base is active.

Example. Calculate $B17_{16} + 47_8$. Display the result in hexadecimal base.

Select hexadecimal base and enter the two values. Append the lowercase letter o ([α] [↰]O) to the octal value to specify its base.

[MTH] BASE HEX
[↱][#] B17 [ENTER]
[↱][#] 47o [ENTER]



Add the two values.

⊞



Press DEC to return to decimal base.

Binary integers are covered in chapter 14.

Arrays

Arrays can be one-dimensional (*vectors*) or two-dimensional (*matrices*). The delimiters for arrays are square brackets ([ ]). The HP 48 MatrixWriter application helps you enter and edit matrices.

Example. Multiply the following matrix and vector.

$$\begin{pmatrix} 1 & -2 & 0 \\ 4 & 5 & -3 \end{pmatrix} \times \begin{bmatrix} 2 & 1 & 2 \end{bmatrix}$$

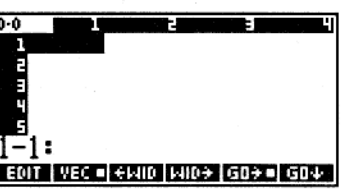
Enter the vector [2 1 2]. Use spaces to separate the components.

◀ [ ] 2 SPC 1 SPC 2 ENTER



Now use the MatrixWriter application to enter the matrix. First, select the MatrixWriter application.

▶ MATRIX



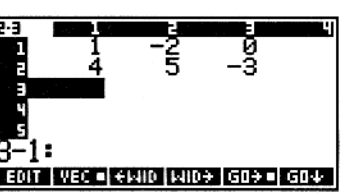
Enter the first row.

1 ENTER
2 +/– ENTER
0 ENTER



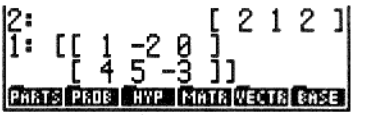
Start a new row and enter the three values. You can enter them one at a time, or you can enter them all at once by separating them with spaces.

▼
4 SPC 5 SPC 3 +/– ENTER



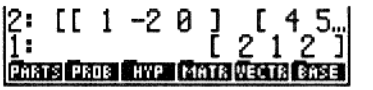
Now enter the matrix into level 1.

ENTER



To do the multiplication, the matrix must be in level 2 and vector must be in level 1, so swap the levels. (Pressing ▶ when no command line is present is the same as pressing ◀ SWAP.)

▶



Multiply them.

×



Arrays are covered in chapter 20.

Names

Names are used to identify variables. If you want to put a name on the stack without evaluating it, enclose it between ' (tick) characters.

Example. Enter the names *A1* and *B1* and multiply them.

Enter the names on the stack. The ALG annunciator comes on when you press '.

' A1 ENTER
' B1 ENTER



Multiply the two names.

×



The result is an algebraic expression containing the two names.

Variable names are covered in chapter 6.

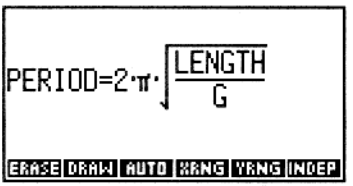
Algebraic Objects

Algebraic objects, like names, are delimited by two ' marks. Algebraics represent mathematical expressions and, on the stack, have a conventional "computer form" like these two examples:

```
'PERIOD=2*π*√(LENGTH/G)'

'∂X(2*X^3+COS(X))'
```

The EquationWriter application helps you enter and manipulate algebraic objects by displaying them as they might appear printed in a book. For example, this is how the PERIOD equation above would look in the EquationWriter environment:



Algebraic objects, often referred to as *algebraics* in this manual, are covered in chapter 8; the EquationWriter application is covered in chapter 16.

Programs

Programs are sequences of commands and other objects enclosed by the delimiters « and ». For example, given a real number argument in level 1 representing the radius, this program computes the area of a circle:

```
« SQ π →NUM * »
```

The delimiters prevent the commands from being executed as you enter them. Instead, they are executed later when you evaluate the program object.

Programming is covered in part 4 (chapters 25 through 31).

Strings

Strings are sequences of characters, usually used to represent text in programs. They are delimited by quotation marks. For example, you can enter the string `"Minor of a Matrix"` onto the stack and then print it.

A *counted string* is an alternate string form in which the number of characters is specified. Counted strings are prefaced with `C$` *n*, where *n* is a real integer. `C$` designates that the string is a counted string, and *n* specifies the number of characters to be gathered into the string. For example, entering `C$ 7 ABC DEF GHI` from the command line results in the creation of the string `"ABC DEF"`. The leftover `GHI` is entered as a name, just like it would be entered if it were on the stack by itself.

Another form of the counted string is to preface the string with `C$ $`. This form specifies that all characters remaining on the command line are put into the string.

Lists

Lists are sequences of objects grouped together, delimited by braces — for example, `{ X 0 1 }`. Lists allow you to combine objects so they can be manipulated as one object.

Graphics Objects

Graphics objects encode the data for HP 48 "pictures," including plots of mathematical data, custom graphical images, and representations of the stack display, itself. They are created by certain plotting commands, and are viewed in the Graphics environment. They can also be put on the stack and stored into variables. On the stack, a graphics object is displayed as:

GRAPHIC $n \times m$

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

Creating and manipulating graphics objects is covered in chapters 18 and 19.

Tagged Objects

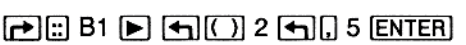
A tagged object consists of an object combined with a tag that labels that object. Tagged objects are keyed in as:

:*tag*: *object*

The colons delimit the tag. When a tagged object is displayed on the stack, the leading colon is dropped for readability.

Example. Enter the complex numbers (2,5) and (4,9) with tags B1 and B2, and then calculate the product.

Enter the tagged objects.





Calculate the complex product.





The tags were ignored by ⊠.

Tagged objects are particularly useful for labeling the contents of variables and program output (see chapter 29).

Unit Objects

A unit object consists of a real number combined with a unit or unit expression. The underscore character (_) separates the unit from the number—for example, `2_m` and `26.7_kg*m^2/s^2`.

Example. Calculate the following:

$$\frac{50.8\ \frac{\text{ft}}{\text{s}}}{2.5\ \text{s}}$$

Enter the unit objects 50.8 ft/s and 2.5 s.

50.8 ◀ UNITS LENG FT
◀ UNITS TIME ▶ S
2.5 S



Divide the two values.

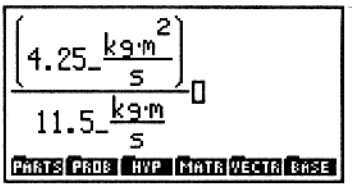
÷



Algebraic expressions can contain unit objects. Here is an example:

```
'(4.25_kg*m^2/s)/(11.5_kg*m/s)'
```

When units are included in algebraic objects, the EquationWriter application helps you enter, edit, and manipulate them. Here is the same expression as it is displayed by the EquationWriter application :



Unit objects are covered in chapter 13.

Directory Objects

The HP 48 uses directory objects to set up hierarchical directory structures for data you store. Directory objects are covered in chapter 7.

Additional Object Types

Three object types involve operations with plug-in cards (covered in chapter 34):

- Backup objects. They are created when you store objects in a plug-in memory card.
- Library objects. A library is a directory of commands and operations that are *not* built into the calculator. Libraries can be provided by a plug-in application card, or they can exist in built-in or plug-in RAM.
- XLIB names. These are objects provided by plug-in application cards.

Two object types describe the HP 48 built-in command set. You can think of them as built-in program objects.

- Built-in functions—for example, SIN and LN.
- Built-in commands—for example, DUP and DRAW.

Commands that Manipulate Objects

The HP 48 contains commands for assembling, disassembling, and modifying portions of objects. These commands (except +) are located in the PRG OBJ (program object) menu (PRG OBJ).

Command/Description	Example	
	Input	Output
+ (+) Combines two strings or lists, or adds an object to a string or list.	2: 'A' 1: { 2 3 } 2: "ABC" 1: "DE"	2: 1: { A 2 3 } 2: 1: "ABCDE"
→ARRY (→ARR) Stack to array; combines real or complex numbers into an *n*-element rectangular vector or a matrix of dimensions *n* by *m*. (*n* or {*n m*} is in level 1.)	3: 8 2: 9 1: 2 7: 1 6: 2 5: 3 4: 4 3: 5 2: 6 1: { 3 2 }	3: 2: 1: [8 9] 5: 4: 3: 2: 1: [[1 2] [3 4] [5 6]]
CHR Character; returns the character corresponding to the character code. (See the character set in appendix C.)	1: 140	1: "α"

Command/Description	Example	
	Input	Output
C→R Complex to real; separates a complex number (or complex array) into two real numbers (or real arrays) representing the real part and the imaginary part.	2: 1: (2,3)	2: 2 1: 3
DTAG Delete tag; removes the tag from a tagged object.	1: A:123	1: 123
EQ→ Equation to stack; splits an equation into its left and right sides.	2: 1: 'A=B+C' 2: 1: 'B+C'	2: 'A' 1: 'B+C' 2: 'B+C' 1: 0
GET Gets the *n*th (in level 1) element of a vector, matrix, or list, or the ⟨*n m*⟩ element of a matrix.	2: [4 5 6] 1: 2 2: [[4 5 6] [7 8 9]] 1: 4 2: [[4 5 6] [7 8 9]] 1: ⟨ 2 1 ⟩ 2: ⟨ A B C ⟩ 1: 2	2: 1: 5 2: 1: 7 2: 1: 7 2: 1: 'B'

Command/Description	Example	
	Input	Output
GETI Same as GET, except also returns the vector, matrix, or list to level 3 and *n* + 1 to level 2, or the array to level 3 and the next array element to the right to level 2.	3: 2: [4 5 6] 1: 2 3: 2: [[4 5 6] [7 8 9]] 1: 4 3: 2: [[4 5 6] [7 8 9]] 1: { 2 2 }	3: [4 5 6] 2: 3 1: 5 3: [[4 5 6] [7 8 9]] 2: 5 1: 7 3: [[4 5 6] [7 8 9]] 2: { 2 3 } 1: 8
→LIST Stack to list; creates a list containing *n* (in level 1) objects.	3: 'X+Z' 2: 'X' 1: 2	3: 2: 1: {'X+Z' X}
NUM Returns the character code corresponding to the character.	1: "A"	1: 65

Command/Description	Example	
	Input	Output
OBJ→ Object to stack; separates a complex number, array, or list into its elements (same as C→R, ARRY→, and LIST→); for arrays and lists, also returns the number of elements or dimensions to level 1. For strings, removes the string delimiters and executes the contents as a command line (same as STR→). For algebraics, separates the outermost function and its arguments. For units, separates the number and the unit expression. For tagged objects, separates the tag and the object.	2: 1: (4,5) 3: 2: 1: [8 9] 4: 3: 2: 1: [[1 2] [5 6]] 4: 3: 2: 1: 〈1 2 Y〉 1:"5 SQ 2 *" 4: 3: 2: 1: 'A + B'	2: 4 1: 5 3: 8 2: 9 1: 〈 2 〉 5: 1 4: 2 3: 5 2: 6 1: 〈 2 2 〉 4: 1 3: 2 2: 'Y' 1: 3 1: 50 4: 'A' 3: 'B' 2: 2 1: +

Command/Description	Example	
	Input	Output
POS Position of an object in a list, or position of one string within another.	2: { A 3 C } 1: 'C' 2: "ABCDEFG" 1: "DE"	2: 1: 3 2: 1: 4
PUT Replaces the *n*th element of a vector, matrix, or list, or the ⟨*n m*⟩ element of a matrix, with the contents of level 1. (*n* or ⟨*n m*⟩ is in level 2.)	3: [4 5 6] 2: 2 1: 7 3: [[4 5 6] [7 8 9]] 2: 4 1: 2 3: { 7 8 9 } 2: 2 1: 'A'	3: 2: 1: [4 7 6] 3: 2: 1: [[4 5 6] [2 8 9]] 3: 2: 1: { 7 A 9 }
PUTI Same as PUT, except also returns the list to level 2 and *n* + 1 to level 1, or the array to level 2 and ⟨*n m*⟩ + 1 to level 1.	3: [4 5 6] 2: 2 1: 7 3: [[4 5 6] [7 8 9]] 2: 4 1: 2 3: { 7 8 9 } 2: 2 1: 'A'	3: 2: [4 7 6] 1: 3 3: 2: [[4 5 6] [2 8 9]] 1: 5 3: 2: ⟨7 A 9⟩ 1: 3

Command/Description	Example	
	Input	Output
REPL Replace; replaces a portion of a list or string in level 3. Takes the replacement object from level 1 and the position in the list or string to start the replacement from level 2. (How REPL works with graphics objects is described in chapter 19.)	3: ⟨A B C D⟩ 2: 2 1: ⟨F G⟩ 3: ⟨A B C⟩ 2: 3 1: ⟨F G⟩ 3: ⟨A B⟩ 2: 10 1: ⟨F G⟩	3: 2: 1: ⟨A F G D⟩ 3: 2: 1: ⟨A B F G⟩ 3: 2: 1: ⟨A B F G⟩
R→C Real to complex; combines two real numbers or arrays into a complex number or complex array.	2: -7 1: -2	2: 1: (-7,-2)
SIZE Number of elements in a list; number of characters in a string; dimension of an array; and size of a graphics object.	1: ⟨UX 2 Y⟩ 1: "ABCDEFG" 1: [[4 5 6] [7 8 9]] GRAPHIC 6x12	1: 3 1: 7 1: (2 3) 2: # 6d 1: #12d
→STR Object to string; converts an object to a string.	1: 'A+B'	1: "'A+B'"

Command/Description	Example	
	Input	Output
SUB Subset of a list or string. The positions of the beginning and ending elements are in levels 2 and 1.	3: { A B C } 2: 2 1: 3 3: "ABCDEFG" 2: 3 1: 5	3: 2: 1: { B C } 3: 2: 1: "CDE"
→TAG Stack to tag; combines two objects to form a tagged object.	2: 123 1: 'Value'	2: 1:Value: 123
→UNIT Stack to unit; assembles a scalar from level 2 and a unit expression from level 1 to form a unit object.	2: 85 1: 17_m	2: 1: 85_m

GET, GETI, PUT, and PUTI allow name arguments in place of the array argument. For example, evaluating `'A1' 2 GET` returns the second element of *A1*; evaluating `'A2' 2 "ABC" PUT` replaces the second element in *A2* with "ABC".

Object Types

Determining Object Types

There are 20 types of objects used in the HP 48. Each object type is represented by an integer.

Object Type Numbers

Object	TYPE Number	Object	TYPE Number
Real number	0	Binary integer	10
Complex number	1	Graphics object	11
String	2	Tagged object	12
Real array	3	Unit object	13
Complex array	4	XLIB name	14
List	5	Directory	15
Global name	6	Library	16
Local name	7	Backup object	17
Program	8	Built-in function	18
Algebraic object	9	Built-in command	19

The TYPE command ([PRG] OBJ [NXT] TYPE) returns a number representing the type of object in level 1.

The VTYPE command ([PRG] OBJ [NXT] VTYPE) returns a number representing the type of object stored in a variable. The command takes the variable name as its argument. VTYPE returns −1 if the variable doesn't exist.

Separating Variable Names by Object Type

The TVARS command ([←] [MEMORY] [NXT] TVARS) accepts an object type number as an argument, and returns a list containing the names of variables in the current directory containing that object type. For example, pressing TVARS with 8 in level 1 returns a list of all the names of variables containing programs. If no variables contain that object type, TVARS returns an empty list to the stack.

Evaluating Objects

Evaluation is the fundamental calculator operation for prodding an object into action. Evaluation is often implicit in calculator operations—it happens when commands are executed, programs are run, etc. Objects on the stack can be explicitly evaluated by executing the EVAL command (represented on the keyboard by [EVAL]).

The result of evaluating an object can be a sequence of subsequent actions, which can include further evaluations. The following table describes the effect of evaluating different objects.

Obj. Type	Effect of Evaluation
Local Name	Recalls the contents of the variable. If appropriate, the contents can then be explicitly evaluated using the EVAL command.
Global Name	Calls the contents of the variable: ■ A name is evaluated. ■ A program is evaluated. ■ A directory becomes the current directory. ■ Other objects are put on the stack. If no variable exists for a given name, evaluating the name returns the quoted name to the stack.

Obj. Type	Effect of Evaluation
Program	Enters each object in the program: ■ Names are evaluated, unless delimited by tick marks ('). ■ Commands are executed. ■ Other objects are put on the stack.
List	Enters each object in the list: ■ Names are evaluated. ■ Programs are evaluated. ■ Commands are executed. ■ Other objects are put on the stack.
Algebraic	Enters each object in the algebraic: ■ Names are evaluated. ■ Commands are executed. ■ Other objects are put on the stack.
Other Objects	Puts the object on the stack.

Suppose you created two global variables, *TWOPI* and *CIRCUM*:

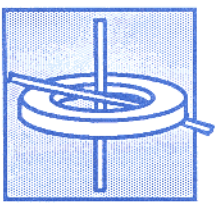
- *TWOPI* contains the real number 6.28318530718.
- *CIRCUM* contains the program « TWOPI * ».

The label CIRCU in the VAR menu represents *CIRCUM*. When you press CIRCU, here's what happens:

1. The name *CIRCUM* is evaluated.
2. The program stored in the variable *CIRCUM* is evaluated.
3. The name *TWOPI* (the first object in the program) is evaluated.
4. The real number stored in the variable *TWOPI* is returned to the stack.
5. The command * (multiply) is executed.

5

Calculator Memory



Every operation you perform with your HP 48 requires memory. This chapter covers the following memory topics:

- Types of memory.
- Commands for memory utilization.
- Low memory conditions.

Types of Memory

The HP 48 has two types of memory:

- *Read-only memory*, or ROM, is memory that is dedicated to specific operations and cannot be altered. The HP 48 has 256K bytes of built-in ROM, which contains its command set. You can expand the amount of ROM by installing plug-in application cards, which are described in chapter 34, "Using Plug-in Cards and Libraries."
- *Random-access memory*, or RAM, is memory that you can change. You can store data into RAM, modify its contents, and purge data. The HP 48 contains 32K bytes of built-in RAM. You can increase the amount of RAM by adding memory cards, which are also described in chapter 34.

RAM is also called *user memory* because it is 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, create an equation or matrix, run a program, etc.

The next two chapters, "Variables" and "Directories," cover the organization and management of user memory.

Memory Utilization Commands

Available Memory. The MEM command (◀ MEMORY MEM) returns the number of bytes of unused user memory.

Memory Requirements and Checksums of Objects. The BYTES command (◀ MEMORY BYTES) takes an object as its argument and returns:

- To level 2: the checksum of the object. The checksum is a binary integer specific to the object. You can use checksums to ensure that you've keyed in a large object (for example, a program or matrix) properly by comparing the checksum of the listing with the checksum you get after you've keyed it in. (The programs in part 4 of this manual each have a checksum at the end of the listing to help you verify that you've keyed in the program correctly.)
- To level 1: The amount of memory in bytes the object takes up. If the object is a variable name, the memory used by the variable's name *and* contents is returned. If it's a built-in object, 2.5 bytes is returned.

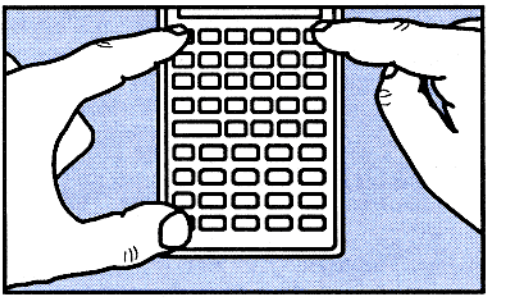
Additional memory commands are covered in chapter 6, "Variables and the VAR Menu," and in chapter 7, "Directories."

Clearing All Memory

Clearing memory *erases all information you've stored* and resets all modes to their default settings. Therefore, you probably won't do this very often, or at least without careful forethought.

To clear all memory:

1. Press and hold down these three keys simultaneously: [ON], the leftmost menu key, and the rightmost menu key.



2. Release the two menu keys, then release [ON]. The calculator beeps and displays the message `Try To Recover Memory?`.
3. Press NO. The HP 48 beeps and displays `Memory Clear`.

If necessary, you can cancel the clearing operation before releasing [ON] by *continuing to hold down* [ON] as you press the second menu key from the left. You can also answer YES to the `Try To Recover Memory?` prompt—however, the calculator may not be able to recover all memory at that point. You probably would lose at least your stack, alarms, and user-key assignments.

Low-Memory Conditions

HP 48 operations share memory with the objects you create. Normal calculator operation becomes slow or fails if user memory is sufficiently full. When a low-memory condition occurs, the HP 48 returns one of a series of low memory warnings. These messages are listed below in order of increasing severity:

No Room for Last Stack. If there is not enough memory to save a copy of the current stack, the message `No Room for Last Stack` is displayed when ENTER is executed. Also, the LAST STACK operation ([←][LAST STACK]) is disabled.

This condition indicates that user memory is getting full. You should make more room by clearing unnecessary objects from memory.

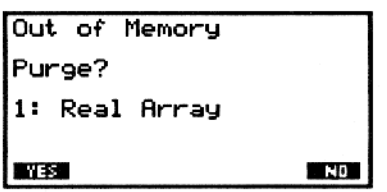
Insufficient Memory. If there isn't enough memory to complete execution of an operation, `Insufficient Memory` is displayed. If the LAST ARG operation ([→][LAST ARG]) is enabled (flag −55 is clear), the original arguments are restored to the stack. If LAST ARG is disabled (flag −55 is set), the arguments are lost.

To alleviate this condition, clear unnecessary objects from memory.

No Room to Show Stack. The HP 48 may complete all pending operations and then not have enough free memory to display the stack. In this case the calculator displays `No Room To Show Stack` in the top line of the display. Those lines of the display that would normally display stack objects, now show those objects only by type, for example, `Real Number`, `Algebraic`, and so on.

The amount of memory required to display a stack object varies with the object type. If there is no room to show the stack, clear one or more objects from memory, or store a stack object in a variable so that it does not have to be displayed.

Out of Memory. In the extreme case of low memory, there is insufficient memory for the calculator to do anything—display the stack, show menu labels, execute a command, etc. In this situation you *must* clear some memory before continuing. A special `Out of Memory` procedure is activated, which starts with the following display:



This display prompts you to press `YES` or `NO` to purge or not purge the object (described by object type) in level 1, which, in the above display, is a real array. If you press `YES`, the object is dropped and the choice is repeated for the new object in level 1. The succession of objects continues until the stack is empty or you press `NO`. Then, the prompt to clear level 1 is replaced by a prompt to discard the contents of LAST CMD ([←][LAST CMD]). That prompt is then followed by other prompts. Here is the sequence of items you are prompted to clear:

1. Stack level 1.
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.



Note

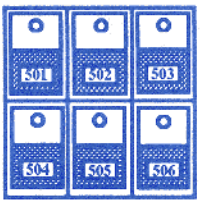
It is possible for the purge sequence to begin with the command line and *then* cycle through the stack, the contents of LAST CMD, etc. If you answer NO to the purge prompt for the command line, you will be returned to the command line when you terminate the Out of Memory procedure.

The prompt for variables (prompt 9 above) starts with the newest object in the *HOME* directory and then proceeds with successively older objects. If the variable to be purged is an empty directory, YES purges it. If the directory is not empty, YES causes the variable-purge sequence to cycle 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 ATTN. 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 with the sequence of choices to purge.

6

Variables and the VAR Menu



A *variable* is a named storage location that contains an object. Variables are used in the HP 48 (instead of numerical storage registers) because they 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.

There are two types of variables:

- *Global* variables are common variables that remain in memory until you purge them.
- *Local* variables are temporary variables created by programs. They exist only while the program is being executed and can't be used outside the program.

This chapter covers global variables. Local variables are covered in chapter 25.

Example: Using Variables.

Display the VAR menu. Initially empty, it contains a label for each global variable you create along with any reserved variables created by the HP 48. (Your VAR menu may not be empty.)

[VAR]



To create the variable *G*, first enter the value and the variable name.

9.81 [←] [UNITS] SPEED M/S
[←] [UNITS] TIME [→] S
[α] G [ENTER]



Execute STO (store) to create the variable and store the value in it. Then redisplay the VAR menu to see the menu label for the variable.

[STO] [VAR]



The value and name are removed from the stack, and the VAR menu now contains a label for variable *G*.

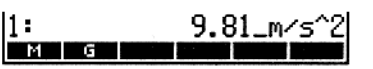
Create the variable *M* with the value 7. A label appears for variable *M*.

7 [ENTER]
[α] M [ENTER] [STO]



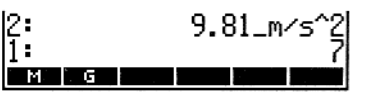
Now, retrieve the contents of *G* using the VAR menu.

[→] G



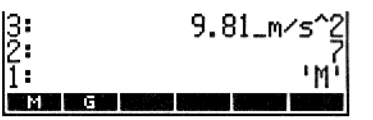
You can also evaluate the contents of a variable without using the VAR menu. Now, retrieve the value of *M*.

M [ENTER]



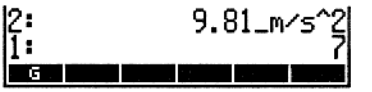
Put the *name* 'M' on the stack.

[α] M [ENTER]
or
[α] M [ENTER]



Now that its name is on level 1 of the stack, purge *M*.

◀ PURGE



Creating a Variable

The STO Command

The STO command takes an object and name from the stack and stores the object in a variable with that name. If the variable doesn't exist, it is created in the current directory. (Directories are covered in chapter 7. If you haven't created any directories, all your variables are created in the HOME directory.) If the variable does exist, the stack object replaces the old value.

You can store any object type in a variable.

Example: Creating a Variable. Create the variable *VCT1* containing the vector [1 2 3].

Enter the vector [1 2 3].

◀ [ ] 1 SPC 2 SPC 3 ENTER



Key in the variable name and press STO. If it's not displayed, display the VAR menu to see the new variable.

' VCT1 STO
VAR



The DEFINE Command

The DEFINE command can be used to create variables from algebraic equations. If in level 1 you have an equation with a valid name as its left side, executing DEFINE stores the expression on the right side of the equation in the name on the left. For example, the keystrokes 'A=6' ◀ DEF store 6 in the variable *A*.

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

Variable Names

Names can contain up to 127 characters, and can contain letters, digits, and most other characters.

A maximum of 10 characters can be displayed in a menu label. If a name is too long to fit in a label, the beginning of the name is shown.

The following characters cannot be included in names:

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

Names cannot begin with a digit. You cannot use command names (for example, SIN, *i*, or π) as variable names. Also, PICT is a special name used by the HP 48 to contain the current graphics object and cannot be used as a variable name.

Certain names are legal variable names, but are used by the HP 48 for specific purposes. These are called *reserved variables*:

- *EQ* refers to the current equation used by HP Solve and Plot applications.
- *CST* contains data for custom menus.
- Σ*DAT* contains the current statistical matrix.
- *ALRMDAT* contains the data for an alarm being built or edited.
- Σ*PAR* contains a list of parameters used by STAT commands.
- *PPAR* contains a list of parameters used by 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.

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.

Using the Contents of a Variable

Once you've created a variable, there are two ways to return its contents:

- *Evaluate* the variable's name. (This is the most common way of using variables.)
- *Recall* the variable's contents.

Evaluating a Variable's Name

Evaluating a variable's name evaluates the object stored in the variable. For objects other than programs, directories, and names, a copy of the contents of the object is returned to the stack. You evaluate a variable's name when:

- You press its key in the VAR menu. For example, G evaluates *G* and displays its contents.
- You key in the name (unquoted) and press ENTER. For example, G ENTER evaluates *G*.

Example: Evaluating a Variable's Name from the VAR Menu. Create three variables—*A* containing 2, *B* containing 5, and *ALG* containing the algebraic expression 'A+B'. Then evaluate them from the VAR menu.

Display the VAR menu and create the variables.

VAR
2 ENTER ' A STO
5 ENTER ' B STO
' A + B ENTER
' ALG STO



Evaluate *ALG*, *B*, and *A*.

ALG
B
A



The contents of the three variables are now on the stack.

Variables Containing Programs. Evaluating the name of a variable containing a program *runs* the program.

Example: Evaluating a Variable Containing a Program. Create the variable *ADD2* containing the program « + + », and then run it to add 3, 4, and 5.

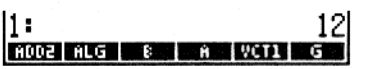
Create the variable.

↶ « » + + ENTER
' ADD2 STO



Now, put 3, 4, and 5 on the stack and evaluate *ADD2*. The program performs two successive additions.

3 SPC 4 SPC 5
ADD2



Variables Containing Directories or Names. If a variable contains a directory, evaluating its name switches to that directory. (Directories are covered in chapter 7.)

If a variable contains a name, evaluating the variable proceeds to evaluate the stored name. For example, if variable *D* contains the name *ALG*, and *ALG* contains 'A+B', pressing D puts 'A+B' on the stack.

Recalling the Contents of Variables

Recalling a variable puts its contents on the stack, without evaluating it. There are two ways to recall a variable:

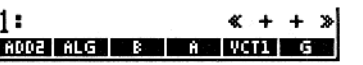
- In the VAR menu, press ↱ followed by a menu key. For example, ↱ ADD2 recalls the contents of *ADD2*.
- Put its name on the stack and execute RCL (↱ RCL).

Since recalling requires more keystrokes than evaluating a variable's name, it is used primarily to get a copy of a stored variable containing a program, directory, or name. (For other object types, just evaluate the name.)

Example: Recalling the Contents of a Variable.

Recall the program stored in *ADD2*

' ADD2 ENTER
→ RCL



Changing the Contents of Variables

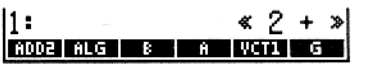
There are three ways to change the contents of a variable:

- With the new contents in level 2 and the variable name in level 1, execute STO.
- With the new contents in level 1, press ← followed by the menu label for the variable.
- With the variable name in level 1, press → VISIT, edit the contents, and then press ENTER.

Example: Changing the Contents of a Variable. Change the contents of *ADD2* from « + + » to « 2 + ».

Enter the new contents.

← « » 2 + ENTER



Store the new contents in *ADD2*.

← ADD2



Using Quoted Versus Unquoted Variable Names

The previous examples used both quoted ('ADD2') and unquoted (ADD2) variable names. The ' delimiter is very important:

- When you delimit a name with ', pressing [ENTER] places the *name* of the variable on the stack. The quote prevents the HP 48 from immediately evaluating the name. Use quoted names when the name is to be the argument of a command, such as STO, RCL and PURGE, or when it is to be used for symbolic calculations.
- Enter an unquoted name when you want to immediately evaluate the name. The action taken depends on the object type of the contents. (See the previous topic, "Evaluating a Variable's Name.")

If you execute an unquoted name that does not exist (no variable has been created with that name), the quoted name is put on the stack. The ability to use names without having to create variables enables you to do symbolic math with the HP 48.

The VAR Menu and REVIEW Catalog

The VAR menu ([VAR]) contains a label for each global variable you've created in the current directory. It provides:

- A way to evaluate a variable's name—simply press the menu key.
- A way to recall a variable's contents—press [→] followed by the menu key.
- A way to change a variable's contents—enter the new contents on the stack, and then press [←] followed by the menu key. (For a full description of this feature, see "Changing the Contents of Variables" on page 111.)
- Typing aids for the variable names. The menu keys act as typing aids when the command line is in Algebraic- or Program-Entry mode.
- A REVIEW "catalog" of your variables. [←][REVIEW] displays the full names and contents of the variables on the current page of the VAR menu.

If the menu contains more than six labels, use [NXT] and [←][PREV] to change pages.

Example: Using the VAR Menu and REVIEW Catalog.

Create a variable named *OPTION* containing 6.011991 and display the VAR menu.

6.011991 ENTER
↑ OPTION STO
VAR



Recall the value of the variable.

→ OPTIO



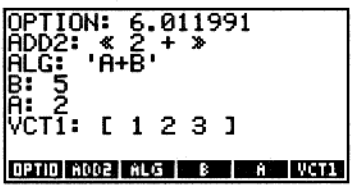
Enter the name of the variable.

↑ OPTIO ENTER



Display a catalog of the variables.

← REVIEW



The next keystroke you make cancels the review, redisplays the stack, and then executes the keystroke itself.

Reordering the VAR Menu

To reorder the contents of the VAR menu:

1. Create a list containing the variable names, in the order you want them to appear in the VAR menu. The list does not have to include all the names. The names that are omitted will be positioned after the names in the list.
2. Execute the ORDER command (← MEMORY ORDER).

One way to create the list (step 1) is to execute the VARS command (← MEMORY VARS). VARS returns a list containing all the variables in the current directory. You can then edit the list.

Purging Variables

Purging Individual Variables

To purge a variable, enter its name and press ◀ PURGE.

Example: Purging a Variable.

Purge the variable *OPTION* created in the previous example (you can type the name or use the variable's menu key as a typing aid):

' OPTION ◀ PURGE



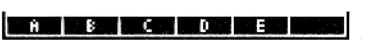
or

' OPTIO ◀ PURGE

Purging More Than One Variable

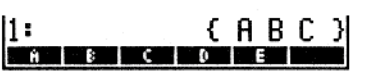
To purge more than one variable at a time, create a list containing the names of the variables to be purged.

Example: Purging Several Variables Together. Suppose you had created a number of variables, so that the VAR menu looked like this:



To purge *A*, *B*, and *C*, create a list containing their names. (Notice that { } places the command line in Program-entry mode.)

◀ { } A B C ENTER



Execute the PURGE command.

◀ PURGE



Purging All Variables

The CLVAR (clear variables) command purges all the variables in the current directory. Since this command can erase a great deal of data, there is no immediate-execution key for it. However, [→][PURGE] is a typing aid that displays CLVAR in the command line. Press [ENTER] to execute the command.

Error Recovery

If you accidentally overwrite or purge a variable by pressing [STO] or [←][PURGE], you can recover from the error by pressing [→][LAST ARG] before executing any other operations. This returns the contents of the variable prior to the error and the variable name to the stack. Then, press [STO] to restore the variable.

Variable Arithmetic

The variable arithmetic commands perform arithmetic operations on a variable's contents without retrieving the contents to the stack. The commands are located in the MEMORY Arithmetic menu ([→][MEMORY]).

Variable Arithmetic Commands

Keys	Programmable Command	Description
[→][MEMORY]:		
STO+	STO+	Adds two objects where one is taken from the stack and the other is the contents of a variable specified by a name on the stack. The new object is stored in the variable.

Variable Arithmetic Commands (continued)

Keys	Programmable Command	Description
STO-	STO-	Subtracts two objects on the stack where one is taken from the stack and the other is the contents of a named variable. The new object of the variable is the difference between the object in level 2 and the object in level 1.
STO*	STO*	Multiplies two objects where one is taken from the stack and the other is the contents of a variable specified by a name on the stack. The product becomes the new value of the variable.
STO/	STO/	Divides two objects on the stack where one is taken from the stack and the other is the contents of a named variable. The new value of the variable is the quotient of the object in level 2 divided by the object in level 1.
SINV	SINV	Computes the inverse of the contents of the variable named on the stack; the result replaces the original contents of the variable.
SNEG	SNEG	Negates the contents of the variable named on the stack; the result replaces the original contents of the variable.
SCON	SCONJ	Conjugates the contents of the variable named on the stack; the result replaces the original contents of the variable.

Variable Arithmetic Examples

Command	Previous Contents of ABC	Stack Contents	Final Contents of ABC
STO+	10	2: 'ABC' 1: 6	16
	10	2: 6 1: 'ABC'	16
STO−	10	2: 'ABC' 1: 3	7
	10	2: 3 1: 'ABC'	−7
STO*	10	2: 'ABC' 1: 5	50
	10	2: 5 1: 'ABC'	50
STO/	20	2: 'ABC' 1: 2	10
	20	2: 5 1: 'ABC'	0.25
SINV	10	1: 'ABC'	0.1
SNEG	10	1: 'ABC'	−10
SCONJ	(−2,−3)	1: 'ABC'	(−2,3)

Two additional commands, INCR and DECR, are used primarily in programming. They are covered in chapter 27, "Loop Structures."

7

Directories



Directories let you organize variables into meaningful groupings. They also let you "bury" information that you use infrequently and protect data that you don't want programs (or people) to alter accidentally.

This chapter covers these topics:

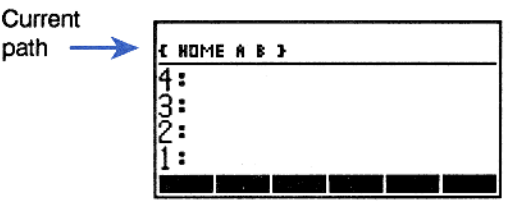
- Understanding HP 48 directories.
- Creating subdirectories.
- Creating and accessing variables in directories.
- Changing, purging, and manipulating directories.

Directory Concepts

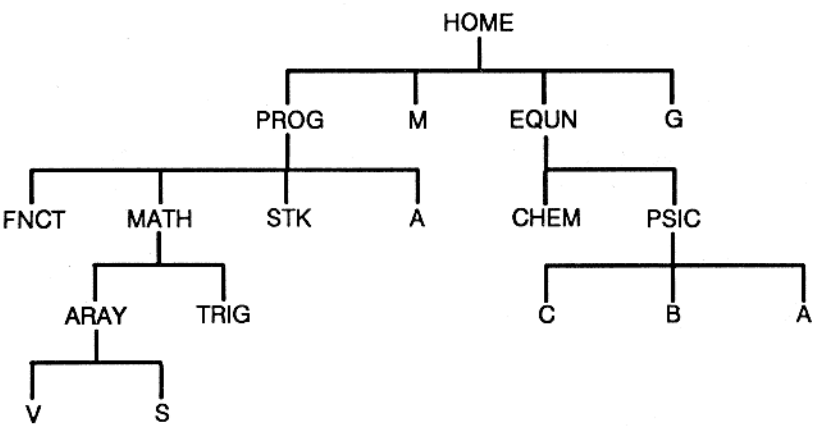
To use directories, you need to understand several concepts:

- Directories are stored in named variables and appear in the VAR menu. To distinguish them from other variables in the VAR menu, each directory has a bar over the left side of its menu label.
- *HOME* is the top-level directory. It's the only directory that exists when the calculator is turned on for the first time.

- At any time, one directory is the *current directory*. Pressing [VAR] displays a menu of the variables in that directory. (When you first turn on the HP 48, *HOME* is the current directory.)
- If directory *A* contains the name of directory *B*, then *A* is the *parent* of *B* and *B* is a *subdirectory* of *A*.
- The sequence of directories, starting from *HOME*, that leads to directory *B* is the *path* of *B*. If *B* is the current directory, then its path is the *current path*. The current path is shown in the status area.



The following figure illustrates an example directory structure.



HOME contains four entries; two (*PROG* and *EQUN*) are names of **subdirectories**. *HOME* is the **parent** of *PROG* and *EQUN*. Moving downward, *PROG* is the parent of *MATH*, and *MATH* is the parent of *ARAY*. Another way of stating the relationships is that *PROG* is a subdirectory of *HOME*, *MATH* is a subdirectory of *PROG*, and *ARAY* is a subdirectory of *MATH*.

The **path** of *EQUN* is { HOME EQUN }; the path of *ARAY* is { HOME PROG MATH ARAY }.

The PATH command ([←] [MEMORY] PATH) returns the path to the current directory in the form of a list of directory names. For instance, if *ARAY* were the current directory and you executed PATH, the list { HOME PROG MATH ARAY } would be returned to level 1. (Evaluating a directory list can be used to switch to the directory specified by the path.)

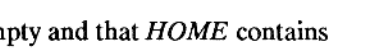
Creating Subdirectories

To create a subdirectory, enter the subdirectory name and press CRDIR. The new name is added to the VAR menu; a bar over the left side of the menu label indicates that it is a directory.

Example: Creating and Using Subdirectories. Create two subdirectories in the *HOME* directory. Name them *EQUN* and *PROG*.

If necessary, press [→] [HOME] to make *HOME* the current directory. (The status area will display { HOME }.) Then, display the VAR menu.

[VAR]



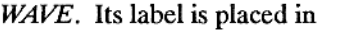
The above display shows that the stack is empty and that *HOME* contains the variable *G*. (Your stack and *HOME* directory may be different.)

Use the CRDIR (create directory) command to create the subdirectories.

' EQUN [←] [MEMORY] CRDIR
' PROG CRDIR [VAR]

The names of the new directories have been added to the VAR menu. A bar over the left side of each label indicates that they are directories. Now, switch to the *EQUN* directory. It is initially empty.

EQUN



Store '`Y=SIN(X)`' into a variable named *WAVE*. Its label is placed in the VAR menu.

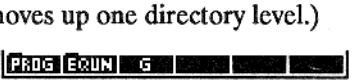
' Y [←] [=] [SIN] X [ENTER]
' WAVE [STO]

Switch to the *HOME* directory. (↰ UP moves up one directory level.)

↱ HOME

or

↰ UP



Creating and Accessing Variables in Directories

Creating Variables

When you create a variable, it is added to the current directory. If the variable name already exists in that directory, the new variable overwrites the previous contents.

Accessing Variables

When you evaluate a name, the HP 48 searches the current directory for that name. If the name isn't there, the HP 48 searches the parent directory, continuing upwards, if necessary, all the way to the *HOME* directory. This provides several useful features:

- A variable in the *HOME* directory can be accessed from any other directory as long as there is no other variable with the same name along the current path. For example, variables *M* and *G* in the diagram on page 119 can be accessed from anywhere. However, if the *PROG* directory contained another variable *M*, that variable would be used when *PROG*, *MATH*, or *ARAY* was the current directory.
- Variables beneath the current directory cannot be accessed. Referring to the diagram on page 119, when *EQUN* is the current directory, you cannot access variables in the *PSIC* directory.
- You can use duplicate variable names. The two variables *A* in the diagram on page 119 are unrelated; one can be accessed from *MATH* and *ARAY*, the other from *PSIC*.

Changing Directories

Switching to a Subdirectory

To switch to a subdirectory, evaluate its name. For example, do one of the following:

- Press the menu key for that directory in the VAR menu.
- Key in the unquoted directory name and press [ENTER].
- Key in a list containing the path to the subdirectory you want and press [EVAL].

Switching to the Parent or *HOME* Directory

Two commands let you move upwards in the directory structure:

- UPDIR ([←][UP]) switches up one level to the parent directory.
- HOME ([→][HOME]) switches to the *HOME* directory.

Also, evaluating the name of any directory in the current path switches to that directory.

Purging Variables and Directories

Purging the Contents of a Directory

Purging Individual Variables. To purge a particular variable, put its name on the stack and execute PURGE ([←][PURGE]).

Purging All Variables in a Directory. The CLVAR command purges all variables in the current directory. (CLVAR has no menu key; however, [→][PURGE] is a typing aid for CLVAR.) If the current directory contains a subdirectory that is not empty, CLVAR errors, leaving that subdirectory as the first entry in the VAR menu.

Purging Variables Using the VARS Command. The VARS command ([↰] [MEMORY] VARS) returns a list containing all the variables and subdirectories in the current directory. This list, or an edited version of it, can be used as the argument of PURGE.

Purging a Directory

A directory must be empty before you can purge it using PURGE; that is, it cannot contain any variables. Once a directory is empty, you can purge it like any other variable—put its name on the stack and execute PURGE. Empty subdirectories are also purged by the CLVAR command.

A non-empty directory can be purged by putting its name on the stack and executing PGDIR ([↰] [MEMORY] [NXT] [NXT] PGDIR). (Take care when using this command—make sure you know what you're deleting before you do it!)

Using Directory Objects

A subdirectory is a variable containing a *directory object*. Creating a subdirectory with CRDIR (create directory) is analogous to creating other variables with STO, except that you are specifically creating a variable containing an empty directory object. For example, ' EQUN CRDIR creates a directory *EQUN* by storing an empty directory object into a variable named *EQUN*.

You can recall a directory to the stack in one of two ways:

- Press [↱] followed by the menu key in the VAR menu for the directory.
- Key in the quoted name of the directory and press [↱] [RCL].

Directory objects are displayed as:

```
DIR
name1          object1
name2          object2
:
END
```

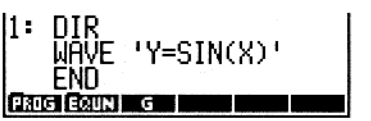
where $name_n$ is the name of a variable in the directory, and $object_n$ is the contents of that variable. The words DIR and END are the delimiters of the directory object. (You can also create or edit a directory of this form in the command line.)

Since subdirectories are variables containing a particular type of object, they can be manipulated like other variables. For example, they can be recalled to the stack and then restored in another directory. This provides a way to copy or move subdirectories. *HOME* is a special directory that is *not* a variable. Therefore, its contents cannot be manipulated in the same way you can manipulate directory objects.

Example: Recalling a Directory to the Stack. Change the directory name *EQUN* to *BIO*.

Recall the directory to the stack.

[ ] EQUN [→] [RCL]



Store it using the new name.

[ ] BIO [STO]



Purge the old directory.

[ ] EQUN
[←] [MEMORY] [NXT] [NXT] PGDIR





Note

If you recall a directory to the stack, and then change the directory contents, the copy on the stack will change as well.

8

More About Algebraic Objects



Algebraic objects (*algebraics* for short) are the vehicle for symbolic mathematics in the HP 48. This chapter addresses topics that will help you understand better the behavior of algebraics:

- Evaluation of algebraics.
- Rules of algebraic precedence.
- Expressions and equations.

Evaluation of Algebraics

Evaluation moves an algebraic towards its numerical value. To evaluate an algebraic, execute EVAL (press [EVAL]) with the algebraic in the command line or level 1.

To understand what to expect when you evaluate an algebraic, recognize that an algebraic is equivalent to a *program* (introduced in chapter 4). A program is simply a series of objects enclosed by « » delimiters. Evaluating a program means: "Put each object in the program on the stack, and, if the object is a command or unquoted name, evaluate it." The same procedure is carried out when an algebraic is evaluated. (The one exception is that names in algebraics cannot be quoted.)

Suppose variable *X* contains the value 3, and *Y* has value 4. When you execute `'X+Y' EVAL`, `7` is returned to the stack. Here's how:

1. The name *X* is evaluated, returning `3` to level 1 of the stack.
2. *Y* is evaluated, returning `4` to level 1 and pushing `3` to level 2.
3. `+` is evaluated, taking the arguments `3` and `4` from the stack and returning `7`.

Now suppose variable *X* contains the value 3 and variable *T* is *formal* (has no value stored in it). When you execute `'X-T' EVAL`, `'3-T'` is returned to the stack. Here's how:

1. *X* is evaluated, returning `3` to level 1.
2. *T* is evaluated. Since *T* has no value associated with it, it just returns `T` to level 1, pushing `3` to level 2.
3. This time `-` takes arguments `3` and `T` from the stack. Since `T` is a formal variable, `-` returns an *algebraic* `'3-T'` to level 1.

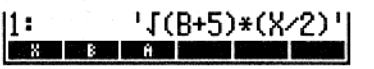
Stepwise Evaluation. Evaluation is a stepwise process. Suppose *A* contains `'B+5'`, *B* contains `'X/2'`, and *X* contains `3`.

[ ] B [+] 5 [ENTER] [ ] A [STO]
[ ] X [÷] 2 [ENTER] [ ] B [STO]
3 [ ] X [STO]
[VAR]



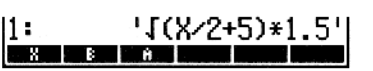
Evaluate `'√A*B'`. Each occurrence of *A* evaluates to `'B+5'` and each occurrence of *B* evaluates to `'X/2'`.

[ ] [√x] A [×] B
[EVAL]



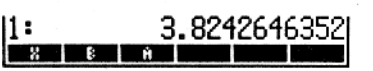
Evaluate the algebraic again. Once again, each occurrence of *B* evaluates to `'X/2'`. Furthermore, each occurrence of *X* evaluates to `3`.

[EVAL]



Evaluate again to complete the process.

[EVAL]

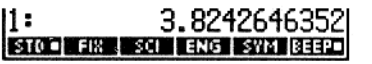


Symbolic Versus Numerical Results

In the previous example, the HP 48 was in *Symbolic Results* mode — repeated evaluation resulted in the progressive resolution of symbolic terms until a numerical result was obtained. This is the default state for the calculator. If you select *Numerical Results* mode, algebraics evaluate *directly to a number* in one step. Numerical Results mode is activated by pressing ◀ MODES SYM■ (or setting flag −3). Note the mode governs execution of *functions* — algebraics are affected indirectly.

Evaluate the algebraic from the previous example in Numerical Results mode.

◀ MODES SYM■
' √x A × B
EVAL



The algebraic evaluates directly to a number.

Note that in Numerical Results mode, evaluation of an algebraic that contains a *formal* variable (a variable in which no object is stored) generates an error because that variable prevents obtaining a numerical result. For example, evaluation of `'X-T'` where *X* has value 3 and *T* is formal leaves `3` in level 2 and `'T'` in level 1, and displays the message:

```
+ Error:
Undefined Name
```

The →NUM Command. If you want to evaluate an algebraic directly to a numerical result while the HP 48 is in Symbolic Results mode, execute the →NUM command with the algebraic as its argument. →NUM:

1. Switches the HP 48 to Numerical Results mode (if Symbolic Results mode is active).
2. Executes EVAL.
3. Turns Symbolic Results mode back on (if it was on before execution of →NUM).

Automatic Simplification

Certain functions, when evaluated, replace certain symbolic arguments or combinations of arguments with simpler forms. For example, when `'1*X'` is evaluated, the `*` function detects that one of its arguments is 1, so the expression is replaced by `'X'`. Here are several examples of automatic simplification:

Original Expression	Simplified Expression
`'SQ(√X)'`	`'X'`
`'X-X'`	`0`
`'X*INV(Y)'`	`'X/Y'`
`'X^(-1)'`	`'INV(X)'`
`'COS(-X)'`	`'COS(X)'`
`'ABS(-X)'`	`'ABS(X)'`
`'EXP(LN(X))'`	`'X'`
`'CONJ(RE(X))'`	`'RE(X)'`

The Rules of Algebraic Precedence

The precedence of operators in an algebraic determines the order of evaluation of terms. Operations with higher precedence are performed first. Algebraics are evaluated from left to right for operators with the same precedence. The following lists HP 48 functions in order of precedence, from highest (1) to lowest (11):

1. Expressions within parentheses. Expressions within nested parentheses are evaluated from the inside out.
2. Functions like SIN or LOG that require arguments in parentheses.
3. ! (factorial).
4. Power (^) and square root (√).
5. Negation (−), multiplication (*), and division (/).
6. Addition (+) and subtraction (−).

7. Comparison operators (== ≠ < > ≤ ≥).
8. Logical operators AND and NOT.
9. Logical operators OR and XOR.
10. The left argument for | (where).
11. Equals (=).

Here are some examples:

- `'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 ^.

Expressions and Equations

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

When an equation is the argument of a function, the result is also an equation, where the function has been applied to both sides. For example, `'X=Y'` `SIN` returns `'SIN(X)=SIN(Y)'`.

In the HP 48, the = sign generally means equating two expressions. The DEFINE command ([↰][DEF]) interprets = differently—it *stores* the expression on the right side of the = sign in the name on the left side.

Related Topics

This chapter does not cover all aspects of algebraic objects — they are used in many different ways in the HP 48. You can find related topics in the following sections of the manual:

- "Symbolic Constants" on page 144 in chapter 9.
- "Using Symbolic Arguments with Common Math Functions" on page 149 in chapter 9.
- "Complex Numbers in Algebraics" on page 164 in chapter 11.
- "Unit Objects in Algebraics" on page 191 in chapter 13.

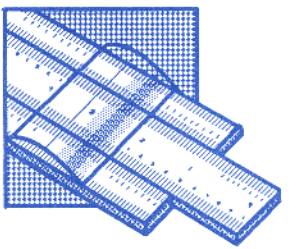
In addition, in chapters 17 — 19, 22, 23, and 25 — 31, you'll see how algebraics are used extensively in the HP Solve application, plotting, algebra, calculus and programming.

Part 2

Hand Tools

9

Common Math Functions

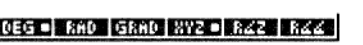


As described on page 42, functions are a subset of commands. The difference between functions and other commands is that functions can be included in algebraics. This means that functions can not only take their arguments from the stack like other commands, but they can also be executed in algebraic syntax as part of an algebraic expression.

Example: Executing a Common Math Function by Using Algebraic Syntax and by Taking Its Arguments from the Stack. Calculate the sine of 30° using an algebraic expression. Then, repeat the calculation using a stack argument.

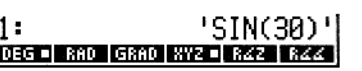
Make sure Degrees mode is set.

◀ MODES NXT NXT DEG



Activate Algebraic-entry mode and enter the expression, supplying the argument in algebraic syntax.

' SIN 30 ENTER



Evaluate the algebraic expression.

EVAL



Now, repeat the calculation using an argument from the stack.

Enter the argument on the stack.

30 ENTER



Take the sine of the argument.

SIN



Keep in mind as you work through the rest of this chapter that the functions described can be executed in both the previous ways. The rest of this chapter describes:

- The various sets of functions for manipulating real numbers. Many of these functions can also be used with other object types.
- The HP 48 built-in symbolic constants—π, e, i, MAXR (maximum real number), and MINR (minimum real number).

The MTH (MATH) Menu

The MTH menu (MTH) is a menu of more specific mathematical menus. Many of the functions described in this chapter are either found on the keyboard or are located in the PARTS, PROB, HYP, and VECTR menus, which are submenus of the MTH menu. Press MTH to see menu labels for these submenus.

Arithmetic and General Math Functions

Many arithmetic and general math functions are found on the keyboard:

Keys	Programmable Command	Description
One-Argument Functions:		
[1/x]	INV	Inverse.
[√x]	√	Square root.
[←][x²]	SQ	Square.
[+/–]	NEG	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).
[←][→Q]	→Q	Converts a decimal value to its best-guess fractional approximation.
Two-Argument Functions:		
[+]	+	Level 2 + level 1.
[–]	–	Level 2 – level 1.
[×]	*	Level 2 × level 1.
[÷]	/	Level 2 ÷ level 1.
[y^x]	^	Level 2 raised to the level 1 power. The algebraic syntax for the ^ command is '*y*^*x*'.
[$\sqrt[x]{y}$]	XROOT	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*)'.

Examples. The following keystrokes show several examples of one- and two-argument math functions.

Calculate $2.7^{1.1 \times 1.6}$. First, enter 2.7.

2.7 ENTER



Next, calculate 1.1×1.6.

1.1 ENTER
1.6 ×



Now, do the exponentiation.

y^x



Calculate $\sqrt[3]{28}$.

28 ENTER 3 ▶ $\sqrt[y]{}$



Enter the complex number (2,4) and negate it.

◀ ( ) 2 SPC 4 ENTER
+/-



Compare the previous results to what happens when you press +/- immediately after keying in the 4.

◀ ( ) 2 SPC 4 +/- ENTER



The function *name* (unevaluated) is used in algebraic expressions.

' $\sqrt{x}$ 5 ENTER



Evaluate the expression.

EVAL



Fraction Conversion Functions

→Q ([→Q]) and →Qπ ([↰] [ALGEBRA] [NXT] [→Qπ]) find a "best-guess" fractional approximation to a real number. The fraction is returned as an algebraic expression.

Example: Converting a Real Number to a Fraction with →Q.
Convert 7.896 to a fraction using →Q.

Key in the number and execute →Q.

7.896 [↰] [→Q]



The accuracy of the fractional approximation is dependent on the display mode. If the display mode is Standard ([↰] [MODES] STD), the approximation is accurate to 11 significant digits. If the display mode is *n* Fix, the approximation is accurate to *n* significant digits.

→Qπ is similar to →Q except that it factors out π. →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. If the denominator of the fractional equivalent of the original number is smaller, that fractional equivalent is returned to the stack; this is the same result as if you had executed →Q. If the denominator of the fractional equivalent of the original number divided by π is smaller, that fractional equivalent, multiplied by π, is returned to the stack.

Exponential, Logarithmic, and Hyperbolic Functions

Exponential and Logarithmic Functions

Keys	Programmable Command	Description
↰ [10^x]	ALOG	Common (base 10) antilogarithm.
↱ [LOG]	LOG	Base 10 logarithm.
↰ [e^x]	EXP	Natural (base e) antilogarithm.
↱ [LN]	LN	Natural (base e) logarithm.

Hyperbolic Functions

Keys	Programmable Command	Description
[MTH] HYP :		
SINH	SINH	Hyperbolic sine: $(e^x - e^{-x})/2$.
ASINH	ASINH	Inverse hyperbolic sine: $\sinh^{-1} x$.
COSH	COSH	Hyperbolic cosine: $(e^x + e^{-x})/2$.
ACOSH	ACOSH	Inverse hyperbolic cosine: $\cosh^{-1} x$.
TANH	TANH	Hyperbolic tangent: $\sinh x/\cosh x$.
ATAN	ATANH	Inverse hyperbolic tangent: $\sinh^{-1}(x/\sqrt{1 - x^2})$.
EXPM	EXPM	$e^x - 1$. Argument x is in level 1. (EXPM is useful when the argument to e^x is close to 0.)

Hyperbolic Functions (continued)

Keys	Programmable Command	Description
LNP1	LNP1	ln (x + 1). Argument x is in level 1. (LNP1, ln *plus* 1, is useful when the argument to ln is close to 1.)

Example. Calculate the hyperbolic sine of 5.

5 MTH HYP SINH



Percent Functions

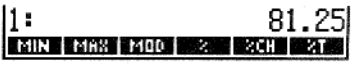
Percent Functions

Keys	Programmable Command	Description
MTH PARTS (page 2):		
%	%	A percent of B (A is in level 2, B is in level 1): $(A \times B)/100$.
%CH	%CH	The percent change between A and B, as a percentage of A (A is in level 2, B is in level 1): $((B - A)/A) \times 100$.
%T	%T	The percent of total (the total, A, is in level 2 and the value, B, is in level 1): $(B/A) \times 100$.

Examples. The following examples demonstrate the percent functions.

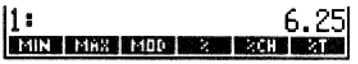
Calculate 12.5% of \$650.

12.5 [ENTER] 650
[MTH] PARTS [NXT] %



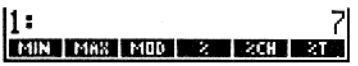
Calculate the percent change between 8 and 8.5.

8 [ENTER] 8.5 %CH



If 35 out of 500 units fail a test, what percentage failed?

500 [ENTER]
35 %T



Angle Mode, Trigonometric Functions, and π

Selecting 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

There are two ways to change the angle mode from the keyboard:

- Press [←] [RAD] to switch between Radians mode and Degrees mode (or between Radians mode and Grads mode, if Grads mode had been previously selected in the MODES menu).
- Use the MODES menu. Press [←] [MODES] [NXT] [NXT], then DEG, RAD, or GRAD; a box in the menu label indicates the active mode.

Trigonometric Functions

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

Trigonometric Functions

Keys	Programmable Command	Description
[SIN]	SIN	Sine.
[←][ASIN]	ASIN	Arc sine.
[COS]	COS	Cosine.
[←][ACOS]	ACOS	Arc cosine.
[TAN]	TAN	Tangent.
[←][ATAN]	ATAN	Arc tangent.

Example. Calculate the sine of 1.1 radians.

Set Radians mode, and then do the calculation.

[←][MODES] [NXT] [NXT] RAD
1.1 [SIN]



Using π

The number π cannot be represented exactly in a finite number of decimal places. The calculator provides a 12-digit approximation (3.14159265359) to π.

The HP 48 also provides a symbolic constant π that represents π exactly; pressing [←][π] enters 'π' onto the stack (as long as flag −3 is clear). In *Radians* mode, the SIN, COS, and TAN functions recognize the symbolic constant and return an exact result. SIN and COS also recognize $\pi/2$.

To replace π with its 12-digit value, press [→][→NUM] (the *to-number* command; see "Symbolic Constants" on page 144 for more information).

Example. Calculate cos ($\pi/2$) and cos ($\pi/4$). (This example assumes the calculator is in Symbolic Results mode—the SYM■ label on page 3 of the MODES menu has a box in it.)

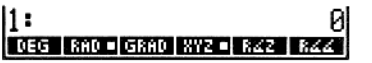
If necessary, switch to Radians mode. Then, put 'π' in level 1 and divide it by 2.

◀ MODES NXT NXT RAD
◀ π 2 ÷



Calculate the cosine.

COS



Now, enter $\pi/4$.

◀ π 4 ÷



Now, calculate cos ($\pi/4$).

COS



The HP 48 retains the symbolic constant π and returns an algebraic expression.

Use →NUM to calculate a numerical result.

▶ →NUM



Switch back to Degrees mode by pressing DEG.

Angle Conversion Functions

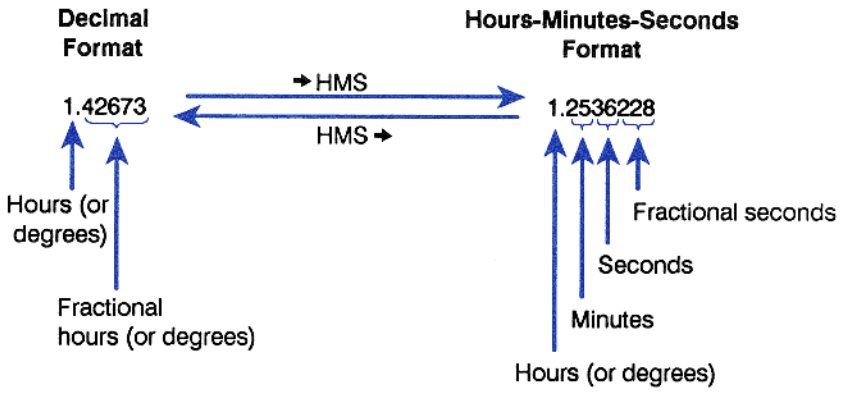
Two commands in the MTH VECTR menu convert values between decimal degrees and radians. Four other commands in the TIME menu let you do calculations using hours(degrees)-minutes-seconds (HMS) format.

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

Angle Conversion Functions

Keys	Programmable Command	Description
MTH VECTR (page 2):		
D→R	D→R	Degrees to radians. Converts a number from a decimal degree value to its radian equivalent.
R→D	R→D	Radians to degrees. Converts a number from a radian value to its decimal degree equivalent.
↰ TIME (page 3):		
→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 numbers in HMS format.
HMS−	HMS−	Subtracts two numbers in HMS format.

The following illustrates the conversion to and from HMS format:



Example. Convert 1.79π radians to degrees.

First, enter 1.79π.

1.79 ENTER
↰ π ×



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

MTH VECTR NXT R→D



Use →NUM to obtain a numerical result.

↱ →NUM



Example. Add 25.2589 degrees to 34 degrees, 5 minutes, 21.22 seconds.

Convert 25.2589 degrees to HMS format.

25.2589 ↰ TIME NXT NXT →HMS



Add 34 degrees, 5 minutes, and 21.22 seconds to the result.

34.052122 HMS+



Symbolic Constants

Using Symbolic Constants

The HP 48 has five built-in numerical constants: π, e, i, MAXR (maximum real number), and MINR (minimum real number). Use lowercase letters for i and e. The examples on pages 141 and 146 illustrate the use of π; i is covered in chapter 11.

Example. The following keystrokes calculate $e^{2.5}$ two different ways—using [e^x] and using e.

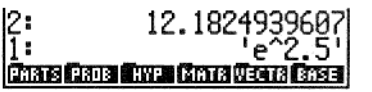
First, use the keyboard function.

2.5 [↰] [e^x]



Enter an algebraic expression for the exponential. (The keystrokes for the letter e are [α], followed by [↰], followed by the menu key with E next to it.)

['] e [y^x] 2.5 [ENTER]



Execute →NUM to completely evaluate the expression to a number.

[↱] [→NUM]



Using Values for Symbolic Constants

You can use the numerical constants in their symbolic form or as their machine-approximated values. When the MODES menu SYM label has a box in it, which is its default state, functions operating on symbolic constants return symbolic results. This state is called *Symbolic Results mode*. When you press the SYM key so that the label does not contain a box, *Numerical Results mode* is active—functions return numerical results.

Example. Assuming Symbolic Results mode is currently active (the MODES menu SYM label has a box in it), compare the effects of entering π and e in Symbolic Results mode and in Numerical Results mode. (The keystrokes for e are [α], followed by [←], followed by the menu key with E next to it.)

Enter π and e in Symbolic Results mode.

[←][π] e [ENTER]



Enter π and e in Numerical Results mode.

[←][MODES] SYM
[←][π] e [ENTER]



Using Flags to Interpret Symbolic Constants

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

- In the default state (both flags clear), evaluating a symbolic constant leaves it unchanged; you must use [→][→NUM] to replace the constant with its numerical value.
- When flag −3 is set, evaluation replaces the constant with its numerical value. (When flag −3 is set, the MODES menu label SYM does not have a box in it.)
- When flag −3 is clear and flag −2 is set, evaluating a symbolic constant returns a numerical result. (However, when the constant is used as an argument of a function, a symbolic result is returned. Pressing [EVAL] then returns a numerical result.)

The →NUM command returns a numerical result, regardless of the flag setting.

Example.

To see the effect of the flag settings, clear both flags and enter π.

2 [+/-] [→] [MODES] [NXT] CF
3 [+/-] CF
[←] [π]



Now, set flag −2 and press [←] [π].

2 [+/-] SF
[←] [π]



Executing π produced a numerical result.

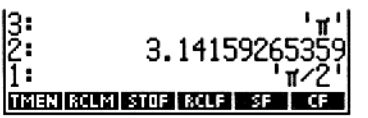
Now, enter the *expression* π.

' [←] [π] [ENTER]



Divide the symbolic π by 2.

2 [÷]



Since flag −3 is clear, the result is symbolic.

[EVAL] returns a numerical result with the current flag settings.

[EVAL]



To return to the default settings, clear flag −2 (press 2 [+/-] CF).

Factorial, Probability, and Random Numbers

Factorial, probability, and random number commands are found in the MTH PROB menu ([MTH] PROB).

Probability Commands

Keys	Programmable Command	Description
[MTH] PROB:		
COMB	COMB	Number of combinations of *n* (in level 2) items taken *m* (in level 1) at a time.
PERM	PERM	Number of permutations of *n* (in level 2) items taken *m* (in level 1) at a time.
!	!	Factorial of a positive integer. For non-integers, ! returns $\Gamma(x + 1)$.
RAND	RAND	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	Takes a real number from level 1 as a seed for RAND. A sequence of random numbers can be repeated by starting with the same seed.

Example. Calculate the number of combinations and permutations of 10 objects taken 4 at a time.

[MTH] PROB
10 [ENTER] 4 COMB
[↱] [LAST ARG] PERM



Other Real-Number Functions

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

Command/Description	Example	
	Input	Output
ABS Absolute value.	1: -12	1: 12
CEIL Smallest integer greater than or equal to the argument.	1: -3.5 1: 3.5	1: -3 1: 4
FLOOR Greatest integer less than or equal to the argument.	1: 6.9 1: -6.9	1: 6 1: -7
FP Fractional part of the argument.	1: 5.234 1: -5.234	1: .234 1: -.234
IP Integer part of the argument.	1: -5.234 1: 5.234	1: -5 1: 5
MANT Mantissa of the argument.	1: 1.23E12	1: 1.23
MAX Maximum; the greater of two arguments.	2: 5 1: -6	1: 5
MIN Minimum; the lesser of two arguments.	2: 5 1: -6	1: -6
MOD Modulo; remainder of $^{A}/_{B}$. *A* MOD *B* = *A* − *B* FLOOR ($^{A}/_{B}$).	2: 6 1: 4	1: 2

Command/Description	Example	
	Input	Output
RND Rounds number according to argument: 0 through 11 to *n* FIX; −1 through −11 to *n* significant digits; 12 to the current display format.	2: 1.2345678 1: 5 2: 1.2345678 1: -5	1: 1.23457 1: 1.2346
SIGN Returns +1 for positive arguments, −1 for negative arguments, and 0 for arguments of 0.	1: -2.7	1: -1
TRNC Truncates number according to argument: 0 through 11 to *n* FIX; −1 through −11 to *n* significant digits; 12 to the current display format.	2: 1.2345678 1: 5 2: 1.2345678 1: -5	1: 1.23456 1: 1.2345
XPON Exponent of the argument.	1: 1.23E45	1: 45

Using Symbolic Arguments with Common Math Functions

Functions that take real numbers as arguments also take symbolic arguments in the same way. For example, the preceding table gives an example of executing ABS on the number −12. If ABS were executed with an argument of *X* instead, the expression `ABS(X)` would be returned to the stack. Then, if the variable *X* contained a value, pressing [EVAL] would evaluate the expression for that value.*

* If flag −3 is set, functions taking symbolic arguments from the stack automatically evaluate to numbers, if possible, when the function is executed.

10

User-Defined Functions



The HP 48 lets you complement the built-in function set with your own *user-defined functions*. Like a built-in function, a user-defined function:

- Takes its arguments from the stack or in algebraic syntax.
- Takes symbolic arguments.
- Can be differentiated.

Example: Differentiating a Built-In Function and a User-Defined Function. Part 1. Calculate

$$\frac{d}{dx}\tan x$$

(This example assumes that variable X does not exist in the current directory.)

First, select Radians mode. Then, enter the algebraic expression `'TAN(X)'`. The expression contains the built-in *function* TAN, which takes its symbolic argument X in algebraic syntax.

◀ RAD (if necessary)
' TAN X ENTER



Enter the variable name X and differentiate the expression with respect to X.

' X ENTER
→ ∂



Part 2. Calculate

$$\frac{d}{dx}\cot x$$

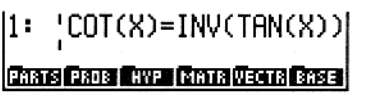
where

$$\cot x = \frac{1}{\tan x}$$

The HP 48 has no built-in cotangent function. However, you can create a user-defined function for cotangent.

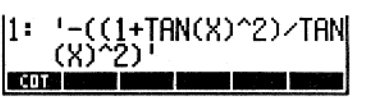
Enter the defining equation for the cotangent function.

' COT ← ( ) X ▶ ← =
1/x TAN X ENTER



Execute DEFINE to create the user-defined function *COT*. Then differentiate the expression `'COT(X)'`.

← DEF
VAR ' COT ← ( ) X ENTER
' X ENTER
→ ∂



You can use any variable as an argument to *COT*—that variable is automatically substituted into the original definition for *COT*.

Creating a User-Defined Function

The DEFINE command lets you create a user-defined function directly from an equation:

1. Enter an equation that defines the function. The equation must have the form *'name(arguments)=expression'*. The left side of the equation consists of the function name followed in parentheses

by its symbolic arguments. The right side of the equation consists of the defining expression for the function.

2. Press ◀ DEF to create the user-defined function.

Example: Creating a User-Defined Function. 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:

$$C = 2^n - 1$$

Enter the equation for *CMB*.

' CMB ◀ ( ) n ▶
◀ = 2 y^x n − 1
ENTER



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.

Example: Executing a User-Defined Function. Execute the user-defined function *CMB* to make the following calculations.

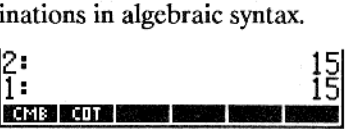
Calculate the total number of ways to combine one or more of four items ($n = 4$).

4 CMB



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

' CMB ◀ ( ) 4
EVAL



Calculate *CMB*(*Z*) in algebraic syntax, where *Z* is a *formal variable* — a variable that does not contain an object.

' Z ◀ PURGE
' CMB ◀ ( ) 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: Nesting a User-Defined Function. Write a user-defined function to calculate the ratio of surface area to volume of a box. The formula for this calculation is

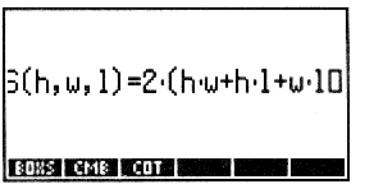
$$\frac{A}{V} = \frac{2(hw + hl + wl)}{hwl}$$

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

Part 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 for the surface area.

◀ EQUATION
BOXS ◀ ( ) h SPC w SPC l ▶
◀ = 2 × ◀ ( ) h × w + h
× l + w × l



Enter the equation and create the user-defined function.

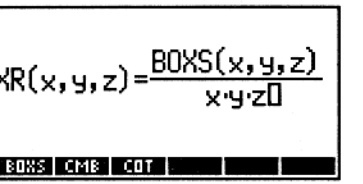
ENTER
◀ DEF



Part 2. Now create a user-defined function *BOXR* to calculate the ratio of surface area to volume.

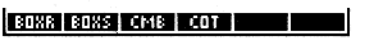
Use the EquationWriter application to key in the equation for the ratio.

[↰][EQUATION]
BOXR [↰][( )] x [SPC] y [SPC] z [▶]
[↰][=] [VAR] BOXS [↰][( )] x
[SPC] y [SPC] z [▶] [÷] x [×] y [×] z



Enter the equation and create the user-defined function.

[ENTER]
[↰][DEF]



Part 3. 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 select the VAR menu and 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.

The Structure of a User-Defined Function

A user-defined function is actually a *program* that:

- Consists *solely of a local variable structure whose defining procedure is an algebraic expression*. The syntax is:

$$\ll \rightarrow name_1\ name_2 \ldots name_n\ \text{'}expression\text{'}\ \gg$$

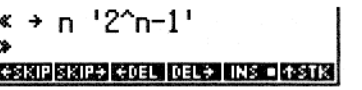
- Takes an unlimited number of arguments (can use an unlimited number of local variables) but returns *one* result to the stack.

Refer to page 473 in chapter 25, "Programming Fundamentals," for a description of local variables and local variable structures.

Example: The Structure of a User-Defined Function. Use VISIT to see the structure of user-defined function *CMB*.

Return *CMB* to the command line, then execute VISIT.

VAR ' CMB → VISIT



You can see that the command sequence:

```
'CMB(n)=2^n-1' DEFINE
```

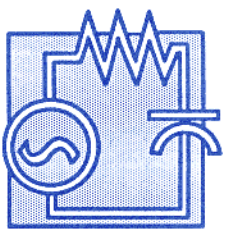
is equivalent to creating the program:

```
« → n '2^n-1' »
```

and storing it in *CMB*.

11

Complex Numbers



This chapter covers:

- Entering complex numbers.
- Interpreting and controlling the display format of complex numbers.
- Assembling and taking apart complex numbers.
- Calculating with complex numbers.
- Using complex numbers in algebraics.
- Determining when to use complex numbers and when to use vectors.

Most functions that work with real numbers also work with complex numbers. So, if you're comfortable using common math functions with real numbers, you should have no trouble with complex numbers.

The examples in this chapter assume the calculator is set to Degrees mode (↰ MODES NXT NXT DEG).

Example: Arithmetic with Complex Numbers. Calculate:

$$\frac{(9 + 4i) + (-4 + 3i)}{(3 + i)}$$

If the R∠Z or R∠∠ annunciator is on, indicating Polar mode is active, press ↱ POLAR to set Rectangular mode. Then, enter the first two complex numbers.

↰ ( ) 9 SPC 4 ENTER
↰ ( ) 4 +/- SPC 3



You do not need to press ENTER before pressing +.

+



Divide the result by 3 + *i*.

↰ ( ) 3 SPC 1 ÷



Entering and Displaying Complex Numbers

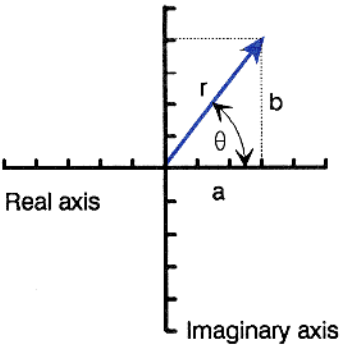
How Complex Numbers Are Displayed

Complex numbers can be displayed in either rectangular form or polar form, depending on whether the HP 48 is in Rectangular coordinate mode or Polar coordinate mode. (Regardless of how complex numbers are displayed, the HP 48 stores them internally in rectangular form.)

In rectangular form, the real part and the imaginary part of the complex number are enclosed in parentheses and separated by a comma.* In polar form, the magnitude and the phase of the complex number are enclosed in parentheses, separated by a comma and angle sign (∠). The polar form is displayed in a form based on the current angle mode (Degrees, Radians, or Grads).

The following diagram shows the display of complex numbers in Rectangular and Polar modes:

* If the Fraction Mark is set to comma, complex numbers are separated by a semicolon.



Display Modes

Rectangular	Polar
(a,b)	(r, ∠θ)

To switch between Rectangular and Polar modes, press [→] [POLAR]. The R∠Z or R∠∠ annunciator indicates that Polar mode is active. (The two annunciators differentiate between Cylindrical and Spherical modes for three-dimensional vectors. For complex numbers, Cylindrical and Spherical modes are interchangeable.)

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

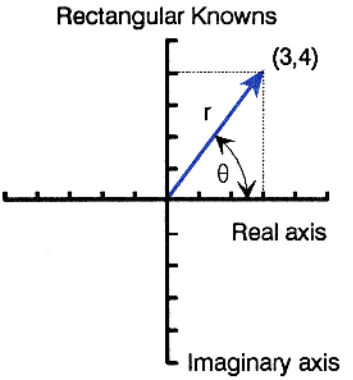
- θ is normalized to the range $\pm 180°$ ($\pm\pi$ radians, ± 200 grads).
- If you key in a negative r, the value is made positive, and θ is increased by 180° and normalized.
- If you key in an r of 0, θ is also reduced to 0.

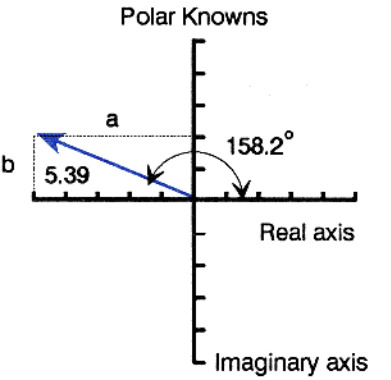
Entering Complex Numbers

There are two ways to enter complex numbers:

- Using parentheses. This method is consistent with the way complex numbers are displayed. It works without any special flag settings and is the method to use for entering complex numbers in the MatrixWriter application. The real and imaginary parts can be separated within the parentheses by either a space ([SPC]), a comma ([←] [.]), or, if the Fraction Mark is set to comma, a semicolon ([←] [.]).
- Using [←] [2D]. This method combines two real numbers on the stack into a complex number. It requires flag −19 to be set. (For more information on [←] [2D], see "Assembling and Taking Apart Complex Numbers" on page 160.)

Example: Entering and Displaying Complex Numbers. The following diagram defines one complex number in rectangular form (3,4) and another complex number in polar form (5.39,∠158.2). Enter these complex numbers and switch the coordinate mode.





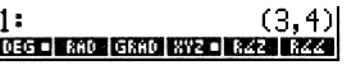
Make sure Degrees angle mode and Rectangular coordinate mode are set.

◀ MODES NXT NXT
DEG XYZ



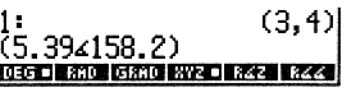
Enter the rectangular complex number using parentheses and a comma.

◀ (3 ◀ , 4 ENTER



Key in the polar complex number using parentheses. (When entering polar complex numbers, you don't need a comma or space to separate the parts—the angle sign acts as the separator.)

◀ (5.39 ▶ ∠ 158.2



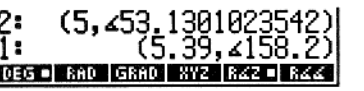
Enter the polar number on the stack; it is converted to match the current coordinate mode (in this case, Rectangular mode).

ENTER



Now change the coordinate mode and watch how the complex numbers change.

[→][POLAR]

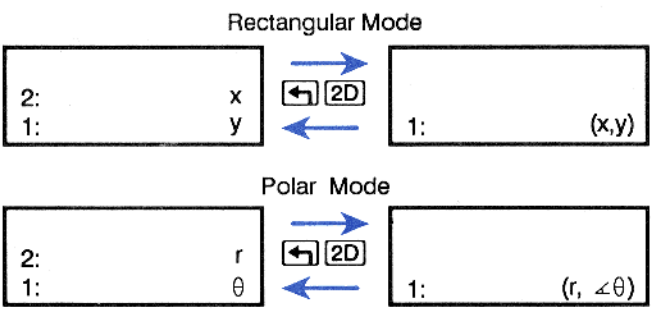


Press [→][POLAR] a few more times to get used to the display conversion. Multiple presses of [→][POLAR] have the same effect as pressing the XYZ and R∠Z (or R∠∠) menu keys.

Assembling and Taking Apart Complex Numbers

Pressing [←][2D] assembles or takes apart a complex number according to the coordinate mode:

- If levels 2 and 1 contain real numbers, and if flag −19 is set, [←][2D] assembles a complex number. If Rectangular mode is set, the real part is taken from level 2 and the imaginary part from level 1; if Polar mode is set, the magnitude is taken from level 2 and the angle from level 1.
- If level 1 contains a complex number, pressing [←][2D] takes it apart. The real part or the magnitude is returned to level 2, and the imaginary part or the angle part is returned to level 1.



Example. Assemble the complex number (3, −5) from its components on the stack, and then take it apart again.

Set flag −19, and make sure Degrees angle mode and Rectangular coordinate mode are set.

19 [+/−] [→] [MODES] [NXT] SF
[←] [MODES] [NXT] [NXT]
DEG XYZ



Enter the parts on the stack.

3 [ENTER] 5 [+/−] [ENTER]



Assemble the complex number.

[←] [2D]



Take apart the complex number.

[←] [2D]



The programmable equivalents of [←] [2D] are the →V2 and V→ commands. (See "Additional Commands for Complex Numbers" on page 165.)

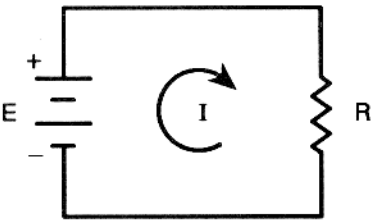
Calculating with Complex Numbers

Comparison with Real Number Calculations

Since a complex number is a single object, like a real number is a single object, calculating with complex numbers is just as easy as calculating with real numbers. Most functions that work with real numbers also work with complex numbers. For instance, to add two complex numbers, you simply put them on the stack and press [+]; to take the sine of a complex number, you simply enter the complex number and press [SIN].

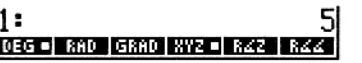
Also, complex numbers are allowed in algebraics just like real numbers. For instance, you can take the sine of (3,4) by entering `'SIN ((3,4))'` and pressing [EVAL].

Example: An Electrical Circuit. Part 1. Ohm's law defines the relationship between the resistance (R), voltage potential (E), and current (I) in the following circuit as $R=E\div I$.

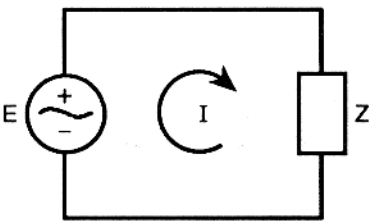


Given a voltage potential of 10 volts and a current of 2 amperes, calculate the resistance.

10 ENTER 2 ÷



Part 2. If the resistance in the previous circuit is replaced by a complex impedance Z, and the simple voltage and current are replaced by phasors (complex numbers representing sinusoidal current or voltage), a problem involving complex numbers results.

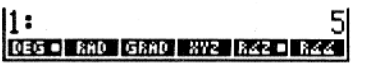


The Ohm's law relationship becomes $Z=E\div I$. For a voltage of $(10,\angle 0)$ and a current of $(2,\angle 30)$, calculate the complex impedance.

Make sure Degrees angle mode and Polar coordinate mode are set.

◀ MODES NXT NXT

DEG R∠Z



Divide the voltage by the current to get the impedance.

[←][( )] 10 [→][∠] 0 [ENTER]
[←][( )] 2 [→][∠] 30 [÷]



Change the coordinate mode to Rectangular.

[→][POLAR]



In polar form, the complex impedance has a magnitude of 5 and a phase angle of −30 degrees. By changing to the rectangular form, you see that the same complex number implies a resistive component of 4.33 ohms and a reactive component of −2.5 ohms. The negative phase and reactance tell an electrical engineer that the impedance is capacitive, rather than inductive.

Complex Results from Real Operations

The complex-number capabilities of the HP 48 can impact the results of real-number operations. Certain calculations that would result in an error on most calculators yield complex results on the HP 48. For instance, 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 invalidated the computation.
- A complex result may be unexpected, but correct, for your problem.

Complex Numbers in Algebraics

Algebraics Containing Complex Numbers

The components of a complex number may be real numbers (for example, in the expression `'X+(1,2)'`) or they may be formal variables (for example, in the expression `'X+(A,B)'`). Upon entry, this second form is automatically converted to an equivalent form, `'X+(A+B*i)'`.

Algebraics containing complex numbers can be manipulated symbolically in the same way as real-number expressions.



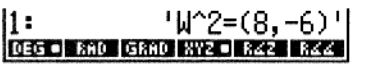
Note

When you enter a complex number as part of an algebraic expression, you must use [◀][,] to separate the real and imaginary parts. (If the Fraction Mark is set to comma, [◀][,] generates a semicolon to separate the parts.)

Example: Using Complex Numbers in Algebraics. Calculate the two square roots of the complex number $8-6i$. Since the $\sqrt{}$ function ([√x]) returns only one root, use the ISOL (isolate) command to solve for W in the equation $W^2=8-6i$.

First, enter the algebraic.

['] W [y^x] 2 [◀][=]
[◀][()] 8 [◀][,] 6 [+/-]
[ENTER]



Now, enter the name of the variable to be isolated (W) and execute the ISOL command.

['] W [ENTER]
[◀] [ALGEBRA] ISOL



The variable ≤1 stands for ±. Thus, the two square roots are $3 - i$ and $-3 + i$. (The ISOL command is covered in chapter 22.)

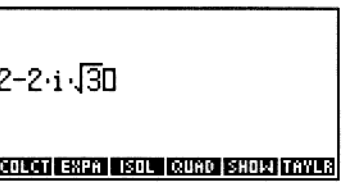
Using the Symbolic Constant i

Complex numbers can be entered as expressions containing the built-in symbolic constant *i*.

Example. Use the EquationWriter application to enter an expression representing the complex number $2 - 2i\sqrt{3}$. Then evaluate the expression to get a complex result. (The answer assumes the HP 48 is set to Rectangular coordinates mode.)

Enter the expression. (The keystrokes for the lowercase letter i are [α], followed by [◀], followed by [CST].)

[◀] [EQUATION]
2 [−] 2i [√x] 3



Use the →NUM command to evaluate the expression and return a complex number object.

[▶] [→NUM]



Additional Commands for Complex Numbers

Most commands that operate on real numbers also operate on complex numbers (for instance, SIN, INV, ^, LN, →Q, etc.). The following table describes additional commands that are especially useful for complex numbers.

Referring to the table, V→ and →V2 are found in the MTH VECTR menu ([MTH] VECTR); NEG is executed in Program-Entry mode by pressing [+/−]; C→R, R→C, and OBJ→ are found in the PRG OBJ menu ([PRG] OBJ); and the remaining commands are found in the MTH PARTS menu ([MTH] PARTS).

Command/Description	Example	
	Input	Output
ABS Absolute value; $\sqrt{x^2 + y^2}$.	1: (3,4)	1: 5
ARG Polar angle of a complex number.	1: (1,1)	1: 45
CONJ Complex conjugate of a complex number.	1: (2,3)	1: (2,-3)
C→R 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 Imaginary (*y*) part of a complex number.	1: (4,-3)	1: -3
NEG Negative of its argument.	1: (2,-1)	1: (-2,1)
OBJ→ Object to stack; separates an object (complex number, array, or list) into its elements.	1: (4,5)	2: 4 1: 5
RE Real (*x*) part of a complex number.	1: (4,-3)	1: 4
R→C Real to complex; combines two real numbers into a complex number (*x*,*y*).	2: -7 1: -2	1: (-7,-2)

Command/Description	Example	
	Input	Output
SIGN 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)
V→ Separates a complex number into two real numbers *x* and *y* or *r* and *θ*, depending on the current coordinates mode. The example assumes Polar and Degrees modes.	1: (3,∠30)	2: 3 1: 30
→V2 If flag −19 is set, assembles two real numbers into a complex number (*x*,*y*) or (*r*,∠*θ*), depending on the current coordinate mode. The example assumes Polar and Degrees modes.	2: 6 1: 50	1: (6,∠50)

Complex Numbers or Vectors?

Complex numbers and two-dimensional vectors can be similar in many ways. Sometimes you may have difficulty choosing the better object type to use for a given problem (and sometimes either type will work).

The main advantages of using complex numbers are that they are allowed as elements of vectors and matrices and that most real number operations work on them. The main disadvantages of using complex numbers are that they are limited to two dimensions and that vector operations like DOT and CROSS don't apply to them.

If you make the wrong choice at the start of a calculation, it's easy to convert from one type to the other.

- If flag −19 is clear and you have a complex number in level 1, pressing ↰ 2D ↰ 2D takes apart the complex number and then

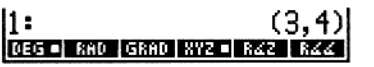
reassembles the parts into a vector.

- If flag −19 is set and you have a two-element vector in level 1, pressing ↰ 2D ↰ 2D takes apart the vector and then reassembles the parts as a complex number.

Example: Converting between Complex Numbers and Vectors. Part 1. Use ↰ 2D to convert the complex number (3,4) into a vector.

Set Rectangular and Degrees modes, and then enter the complex number.

↰ MODES NXT NXT DEG XYZ
↰ ( ) 3 SPC 4 ENTER



Take apart the complex number.

↰ 2D



Clear flag −19 so that ↰ 2D assembles a vector.

19 +/- ↱ MODES
NXT CF



Assemble the vector from the complex parts.

↰ 2D



Part 2. Convert the vector back into the complex number.

Take apart the vector.

↰ 2D



Set flag −19 so that ↰ 2D assembles a complex number.

19 +/- SF



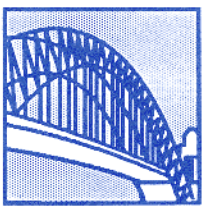
Assemble the complex number from the vector parts.

↰ 2D



12

Vectors



The HP 48 has extensive capabilities for creating and manipulating vectors. These capabilities include a user interface designed specifically for two-dimensional (2D) and three-dimensional (3D) vectors. 2D and 3D vectors are often used to represent forces, velocities, accelerations, torques, and other phenomena of our world.

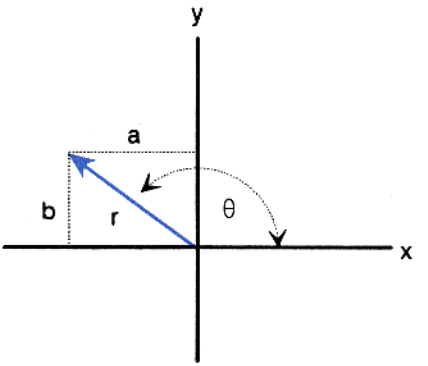
All vectors are array objects, and more mathematically-general vectors are covered in chapter 20, "Arrays." This chapter deals primarily with 2D and 3D vectors and covers the following topics:

- Interpreting and controlling the display format of vectors.
- Entering vectors.
- Assembling and taking apart vectors.
- Performing engineering and physics calculations with vectors.
- Determining when to use complex numbers and when to use vectors.

Displaying 2D and 3D Vectors

How 2D Vectors Are Displayed

Two-dimensional vectors can be displayed in either rectangular form or polar form, depending on whether the HP 48 is in Rectangular coordinate mode or Polar coordinate mode. The following illustration defines the two modes for two-dimensional vectors:



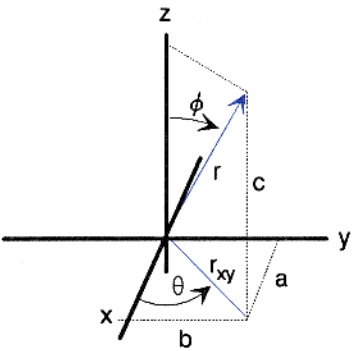
Two-Dimensional Display Modes

Rectangular	Polar
[a b]	[r ∠θ]

To switch between Rectangular and Polar modes, press [→] [POLAR]. The R∠Z or R∠∠ annunciator indicates that Polar mode is active. (The two annunciators differentiate between Cylindrical and Spherical modes for three-dimensional vectors. For two-dimensional vectors, Cylindrical and Spherical modes are interchangeable.)

How 3D Vectors Are Displayed

Three-dimensional vectors can be displayed in rectangular form ([*X Y Z*]), cylindrical form ([*R* ∠ *Z*]), or spherical form ([*R* ∠ ∠]). The following illustration defines these three forms:



Three-Dimensional Display Modes

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

The MTH VECTR menu (MTH VECTR) contains keys for switching between the three coordinate modes:

- Rectangular mode (XYZ■, no annunciator).
- Cylindrical mode (R∠Z■, R∠Z annunciator).
- Spherical mode (R∠∠■, R∠∠ annunciator).

A box in one of the menu labels identifies the current mode.

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

- θ and ϕ are normalized to within ±180° (±π radians, ±200 grads).
- If you key in a negative r, the value is made positive; θ is increased by 180°, ϕ by 90°, 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.

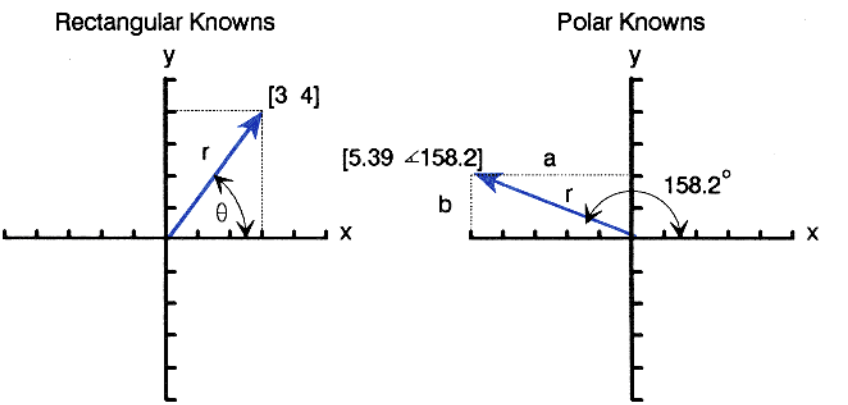
Regardless of how vectors are entered or displayed, the HP 48 stores them internally in rectangular form.

Entering 2D and 3D Vectors

There are two ways to enter 2D and 3D vectors:

- Using square-bracket delimiters ([ ]). This method is consistent with the way vectors are displayed. It works without any special flag settings. Each element is separated within the brackets by a space ([SPC]).
- Using [←][2D] or [→][3D]. This method combines two or three real numbers on the stack into a vector. Assembling a two-dimensional vector using [←][2D] requires flag −19 to be clear. (For more information on [←][2D] and [→][3D], see "Assembling and Taking Apart 2D and 3D Vectors" on page 173.)

Example: Entering and Displaying 2D Vectors. The following diagram defines one vector in rectangular form ([3 4]) and another vector in polar form ([5.39 ∠158.2]). Enter these vectors and look at them in both display modes.



Set Rectangular mode, and, if necessary, Degrees mode.

[MTH] VECTR XYZ
if necessary [←][RAD]



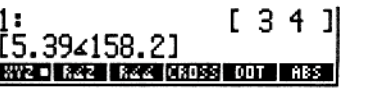
Enter the rectangular vector.

[←][[ ]] 3 [SPC] 4 [ENTER]



Key in the polar vector. (When entering polar vectors, you don't need a space to separate the elements—the angle sign acts as the separator.)

[←][[]] 5.39 [→][∠] 158.2



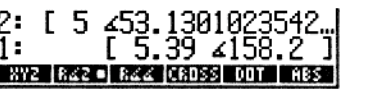
Enter the polar vector on the stack; it is converted to match the current display mode (in this case, Rectangular mode).

[ENTER]



Now change the display mode and watch how the vectors change.

[→][POLAR]



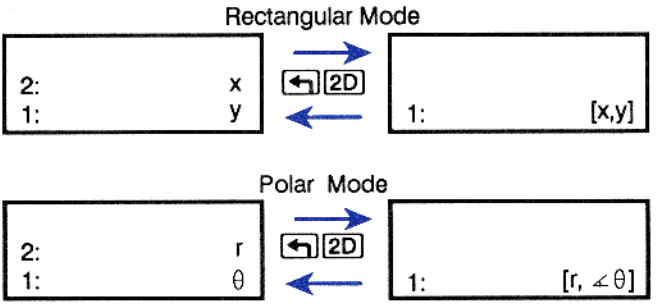
Press [→][POLAR] a few more times to get used to the display conversion. Multiple presses of [→][POLAR] have the same effect as pressing the XYZ and R∠Z (or R∠∠) menu keys.

Assembling and Taking Apart 2D and 3D Vectors

Pressing [←][2D] assembles or takes apart a two-dimensional vector according to the current coordinate mode:

- If levels 2 and 1 contain real numbers, and if flag −19 is clear, [←][2D] assembles a two-dimensional vector from them.
- If level 1 contains a two-dimensional vector, pressing [←][2D] takes it apart.

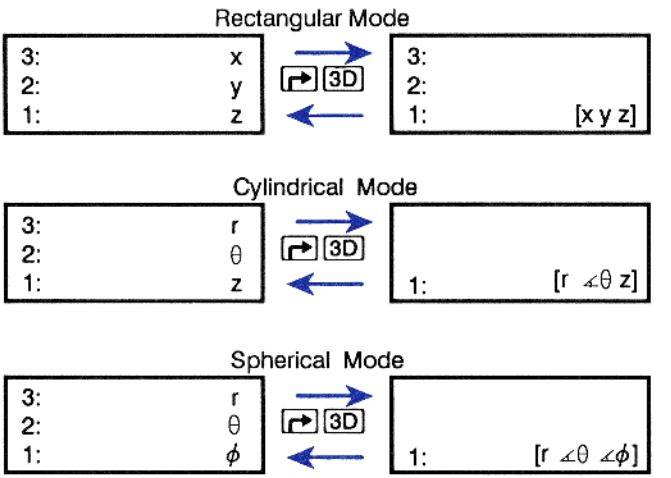
The following illustrates using [←][2D]:



Pressing [↱] 3D assembles or takes apart three-dimensional vectors according to the current coordinate mode:

- If levels 3, 2, and 1 contain real numbers, [↱] 3D assembles a three-dimensional vector from them.
- If level 1 contains a vector with any number of elements, [↱] 3D takes it apart.

The following illustrates using [↱] 3D:



The programmable equivalents of [↱] 3D are the V→ and →V3 commands, and the programmable equivalents of [↰] 2D are V→ and →V2. For descriptions, see "Additional Vector Commands" on page 183.

Example: Assembling and Taking Apart a Two-Dimensional Vector. Use ◀ 2D to assemble and then take apart the two-dimensional vector [3 5]. (This example assumes flag −19 is clear.)

Set Rectangular coordinate mode.

MTH VECTR XYZ



Enter the real number components.

3 ENTER 5 ENTER



Assemble the vector.

◀ 2D

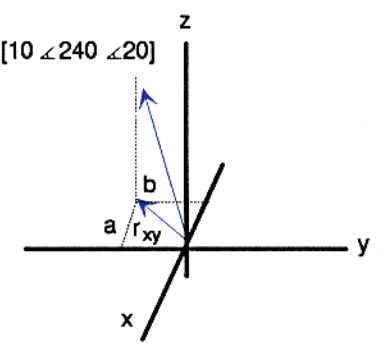


Break the vector apart into its components.

◀ 2D



Example: Assembling and Taking Apart a Three-Dimensional Vector. Use ▶ 3D to assemble and then take apart the following spherical vector:



Remember that when you assemble a 2D or 3D vector, angles are normalized such that they are no larger than 180 degrees (π radians or 200 grads). This causes 240 degrees to be converted to −120 degrees in this example.

Set Spherical coordinate mode and, if necessary, Degrees mode.

MTH VECTR R∠∠
if necessary ↩ RAD



Enter the real numbers associated with the vector.

10 SPC 240 SPC 20 ENTER



Assemble the vector. (Note that 240° is converted to −120°.)

↱ 3D



Break the vector apart.

↱ 3D



Because of the angle normalization, the original stack contents are changed.

2D and 3D Vector Calculations

Because a vector, like a real number, is a single object, you can easily execute many common math functions with vector arguments. 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 — with them. (The absolute value function ABS returns the magnitude of a vector.)

The following examples assume Degrees angle mode is set. Keystrokes for the proper coordinate modes are given where appropriate.

Example 1: Finding the Unit Vector. A unit vector parallel to a given vector is found by dividing a vector by its magnitude. Find the unit vector for [3 4 5].

Set Rectangular coordinate mode and, if necessary, Degrees mode.

MTH VECTR XYZ
if necessary ↩ RAD



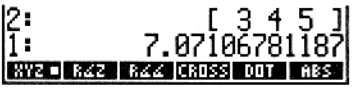
Enter the vector.

3 ENTER 4 ENTER 5 → 3D



Duplicate the vector and compute the magnitude.

ENTER ABS



Divide the vector by its magnitude to get the unit vector.

÷



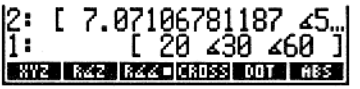
Example 2: Finding the Angle Between Two Vectors. The angle between two vectors is given by

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

Calculate the angle between the vectors [3 4 5] and [20 ∠30 ∠60]. (This example assumes the calculator is originally set to Rectangular mode.)

Enter both vectors. (Notice the change to Spherical mode for entering the second vector.)

3 ENTER 4 ENTER 5 → 3D
MTH VECTR R∠∠
20 ENTER 30 ENTER 60 → 3D



Take the dot product.

DOT



Return the vectors to the stack.

→ LAST ARG



Use the absolute value function to find the magnitude of both vectors.

ABS ↶ SWAP ABS

```
3:          129.641016151
2:                     20
1:           7.07106781187
 XYZ  R▸Z  R▸∠  CROSS  DOT  ABS
```

Multiply the magnitudes and divide the result into the dot product.

× ÷

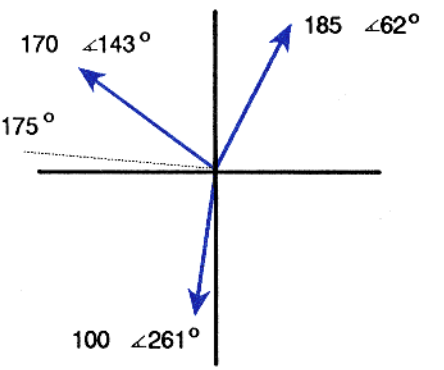
```
1:           .916700416405
 XYZ  R▸Z  R▸∠  CROSS  DOT  ABS
```

Take the arccos to find the angle.

↶ ACOS

```
1:           23.5516253446
 XYZ  R▸Z  R▸∠  CROSS  DOT  ABS
```

Example 3: Finding the Component of a Vector in a Particular Direction. The following diagram represents three, two-dimensional vectors. Find their sum, and then use DOT to resolve them along the 175° line.



Set Polar mode (either Cylindrical or Spherical mode will work), and then enter the three vectors.

MTH VECTR R▸Z
170 ENTER 143 ↶ 2D
185 ENTER 62 ↶ 2D
100 ENTER 261 ↶ 2D

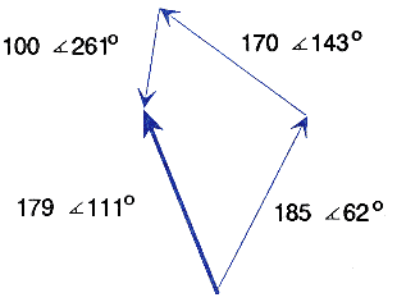
```
3:          [ 170 ∠143 ]
2:           [ 185 ∠62 ]
1:          [ 100 ∠-99 ]
 XYZ  R▸Z  R▸∠  CROSS  DOT  ABS
```

Add them.

+ +

```
1: [ 178.937160532
   ∠111.148894255 ]
XYZ R▲Z ■ R▲∠ CROSS DOT ABS
```

Here's an illustration of the sum:

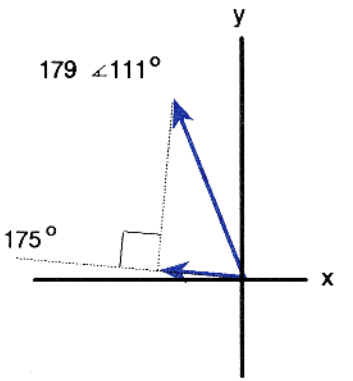


Enter the unit vector of 175°.

1 ENTER 175 ↰ 2D

```
2: [ 178.937160532 ∠1…
1:                 [ 1 ∠175 ]
XYZ R▲Z ■ R▲∠ CROSS DOT ABS
```

Here's a picture of where you are now:

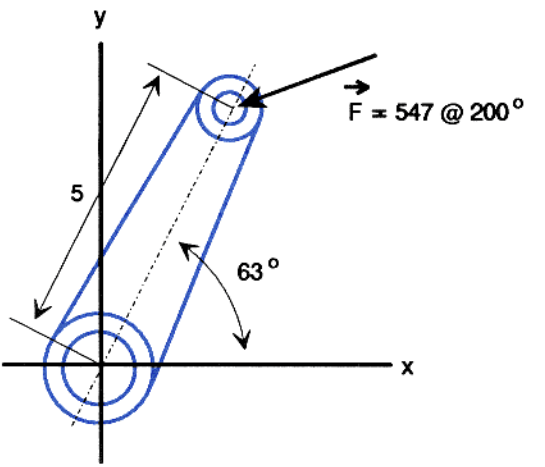


Find the scalar representing the magnitude of the force along the 175° line.

DOT

```
1:          78.8585649505
 ▶V2  R▶Z  R▶Z∡  CROSS  DOT  ABS
```

Example 4: Movement of a Crank. Part 1. For the crank below, what is the moment about the origin, and how much force is transferred along the axis of the crank?



The moment is found by taking the vector cross product of the crank radius and force vectors. (To take a cross product, enter the vectors in the same order that they appear in the cross-product formula: $\mathbf{M} = \mathbf{r} \times \mathbf{F}$.)

Set Polar mode and enter the radius and force vectors.

MTH VECTR R∡∡
5 ENTER 63 ↰ 2D
547 ENTER 200 ↰ 2D

```
2:              [ 5 ∡63 ]
1:          [ 547 ∡-160 ]
 ▶V2  R▶Z  R▶Z∡  CROSS  DOT  ABS
```

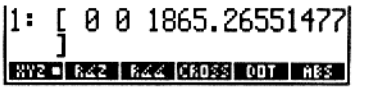
Take the cross product. (Notice that the result is a three-dimensional vector.)

CROSS

```
1: [ 1865.26551477 ∡0
   ∡0 ]
 ▶V2  R▶Z  R▶Z∡  CROSS  DOT  ABS
```

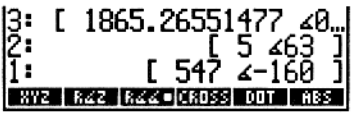
You would expect this three-dimensional vector to be positive and parallel to the z axis. This can be verified by inspection and the right hand rule. Change to Rectangular mode to make the verification easier.

[→][POLAR]



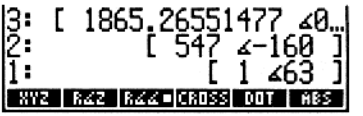
Now return the original vectors to the stack and change back to Polar mode.

[→][LAST ARG] [→][POLAR]



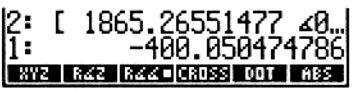
Divide the radius by its magnitude to get the unit vector.

[↶][SWAP] [ENTER] ABS [÷]



Take the dot product to find the scalar representing the magnitude of the force along the crank.

DOT

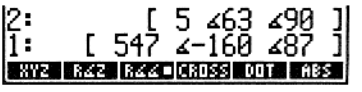


The negative magnitude indicates that the force is opposed to the direction of the crank's unit vector.

Part 2. To add a small twist to the example, suppose the force vector is not on the same plane as the crank. If the force is [547 ∡200 ∡87] (thus the force vector rises out of the paper at a modest 3°), what is the moment, the force transmitted along the axis of the crank, and the thrust force along the z axis?

Enter the radius and force vectors. (Use Cylindrical mode with a z-value of 0 for the radius vector, and use Spherical mode for the force.)

R⇔Z
5 [ENTER] 63 [ENTER] 0 [→][3D]
R⇔∠
547 [ENTER] 200 [ENTER] 87 [→][3D]



Take the cross product.

CROSS



This time the resulting moment is not directed precisely along the z axis. Switch to Rectangular mode and see the z-axis component.

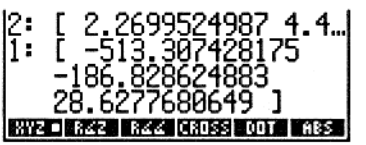
XYZ



The useful moment along the crank has a magnitude of almost 1863.

Now get the original vectors back on the stack. Notice that the thrust problem has been solved through the switch to Rectangular mode.

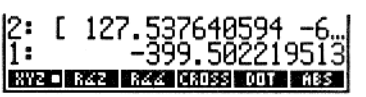
↷ LAST ARG



The thrust (the z-component of the vector) is approximately 28.6. Note that it is positive and comes out of the paper the same as the force vector. (The same value could have been calculated through a more general approach of calculating the dot product of the unit vector associated with the z axis ([0 0 1]) and the force vector.)

Lastly, compute the force along the crank.

↶ SWAP ENTER
ABS ÷ DOT



Additional Vector Commands

The following commands interpret their arguments and return results using the current coordinate mode. These commands are found on page 2 of the MTH VECTR menu (MTH VECTR NXT).

Command/Description	Example	
	Input	Output
V→ Separates a vector (or complex number) into its coordinate elements.	1: [8 9] 1: [5 ∠90]	2: 8 1: 9 2: 5 1: 90
→V2 When flag −19 is clear, creates a 2-element vector.	2: 2 1: 20	Rectangular Mode: 1: [2 20] Polar Mode: 1: [2 ∠20]
→V3 Creates a 3-element vector.	3: 2 2: 20 1: 5	Rectangular Mode: 1: [2 20 5] Cylindrical Mode: 1: [2 ∠20 5] Spherical Mode: 1:[2 ∠20 ∠5]

Additional commands for manipulating vectors are →ARRY, GET, GETI, OBJ→, PUT, and PUTI. These are covered in the table starting on page 90.

Complex Numbers or Vectors?

Complex numbers and two-dimensional vectors can be similar in many ways. Sometimes you may have difficulty choosing the better object type to use for a given problem (and sometimes either type will work).

The main advantages of using complex numbers are that they are allowed as elements of vectors and matrices and that most real number operations work on them. The main disadvantages of using complex numbers are that they are limited to two dimensions and that vector operations like DOT and CROSS don't apply to them.

If you make the wrong choice at the start of a calculation, it's easy to convert from one type to the other.

- If flag −19 is clear and you have a complex number in level 1, pressing ↰ 2D ↰ 2D takes apart the complex number and then reassembles the parts into a vector.
- If flag −19 is set and you have a two-element vector in level 1, pressing ↰ 2D ↰ 2D takes apart the vector and then reassembles the parts as a complex number.

13

Unit Management



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 unit conversions. For example, you can convert the unit object `10_ft` to `120_in` or `3.048_m`.
- Factor a unit with respect to another unit. For example, you can factor `20_W` with respect to `1_N` and return `20_N*m/s`.
- Execute mathematical operations on unit objects. For example, you can add `10_ft/s` to `10_mph` and return `24.67_ft/s`.

A Unit-Management Example. The ideal gas equation of state is:

$$PV = nRT$$

where

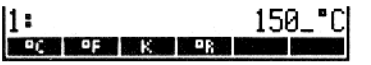
P is the pressure exerted by the gas (in atmospheres).
V is the volume of the gas (in liters).
n is the amount of the gas (in moles).
R is the ideal gas constant (0.082057 liter-atmosphere/kelvin-mole).
T is the the temperature of the gas (in kelvins).

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

Part 1. First, convert the temperature to kelvins.

Select the UNITS Catalog menu and then the TEMP submenu. Create the unit object `150_°C`.

◀ UNITS
NXT TEMP
150 °C



Convert to kelvins.

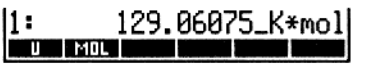
◀ K



Part 2. Execute the calculation for *P*.

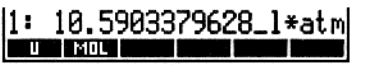
Multiply *T* (already in level 1) by *n*.

◀ UNITS MASS NXT NXT
.305 MOL
×



Multiply *nT* by *R*.

.082057 ◀ UNITS
VOL NXT L
◀ UNITS NXT PRESS ATM
◀ UNITS NXT TEMP ▶ K
◀ UNITS
MASS NXT NXT ▶ MOL
×



Divide by *V*.

.95 ◀ UNITS VOL NXT L
÷

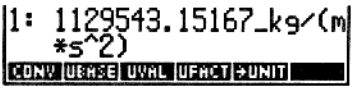


The pressure (in atmospheres) is returned to level 1.

Part 3. Express atmospheres in SI base units.

Select the UNITS Command menu and convert to SI base units.

→ UNITS
UBASE



The pressure, expressed in SI base units, is returned to level 1.

Note that in this example the temperature conversion from °C to K is executed *before* subsequent operations append additional units to the unit object. The conversion to SI base units in Part 3 would have produced an incorrect result otherwise. For more information on temperature conversion, see page 197.

How the Units Application Is Organized

The Units application consists of two menus:

- The UNITS Catalog menu, which contains all the HP 48 units, organized by subject. You use the UNITS Catalog menu to create unit objects and to execute unit conversions between related units in the catalog.
- The UNITS Command menu, which contains commands for unit conversion and for other kinds of unit-object management.

Definition of Terms

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 UNITS Catalog menu contains the seven base units and 141 *compound* units derived from the base units. For example, in (inch) is .0254 m, and Fdy (Faraday) is 96487 A*s.

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 _ character. For example, 2_in (2 inches), X*1_N (*X* Newtons), 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.

A *unit conversion* replaces an old unit expression with a new unit expression (specified by you), and automatically multiplies the number by the appropriate conversion factor.

The UNITS Catalog Menu

To select the UNITS Catalog menu, press ◀ UNITS. This displays a three-page menu of "subject" keys, each of which, when pressed, displays a submenu of related units. For example, press ◀ UNITS NXT PRESS to display a two-page menu of units for pressure.

The individual keys in each submenu behave differently than standard menu keys. When you press the:

- Unshifted key in Immediate-entry mode, the HP 48 *creates* a unit object that corresponds to that key. In Algebraic- or Program-entry modes, the unshifted keys act as typing aids, echoing the corresponding unit name into the command line.
- Left-shifted key in Immediate-entry mode, the HP 48 *converts* the unit object in the command line or stack level 1 to the corresponding unit.
- Right-shifted key in Immediate-entry mode, the HP 48 *divides by* the corresponding unit. This facilitates the creation of unit expressions which have a denominator.

The following sections discuss creating and converting unit objects. In each section, the use of the UNITS Catalog menu is discussed in detail.

Creating a Unit Object

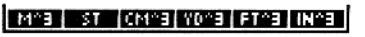
The UNITS Catalog menu provides a simple method for creating a unit object. To create a unit object with the UNITS Catalog menu:

1. Key in the number part of the unit object.
2. Select the subject menu that contains the desired unit.
3. Press the corresponding menu key.

Example 1: Creating a Unit Object. Create the unit object `3.5_ft^3`.

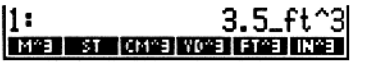
Select the VOL submenu of the UNITS Catalog menu.

◀ UNITS VOL



Key in the number, then append the unit.

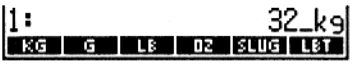
3.5 FT^3



Example 2: Creating a Unit Object. Create the unit object `32_kg*m^2/s^2`.

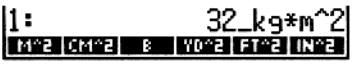
Key in the number and append the first unit.

32 ◀ UNITS MASS KG



Append the second unit.

◀ UNITS AREA M^2



Append the units in the denominator.

◀ UNITS TIME
▶ S ▶ S



How Unit Objects Are Created in the UNITS Catalog Menu. When you press an unshifted menu key in the UNITS Catalog menu, the HP 48 actually:

1. Enters a unit object on the stack consisting of the corresponding unit and a number value of 1.
2. Executes * (multiplies).

Thus, in Example 1, when you keyed in `3.5` and pressed FT^3, the HP 48 entered the unit object `1_ft^3`, pushing `3.5` to level 2, then multiplied to create the unit object `3.5_ft^3`.

When you press a right-shifted menu key in the UNITS Catalog menu, the HP 48:

1. Enters a unit object on the stack consisting of the corresponding unit with a number value of 1.
2. Executes / (divides).

In Example 2, the HP 48 created the unit object `32_kg` by multiplying `32` by `1_kg`. When you pressed `M^2`, the HP 48 entered the unit object `1_m^2` in level 1 and multiplied to create the unit object `32_kg*m^2`. When you pressed [→] `S`, the HP 48 entered the unit object `1_s` and *divided* to create the unit object `32_kg*m^2/s`. When you pressed [→] `S` again, the HP 48 again divided by `1_s` to return the final result.

Creating a Unit Object in the Command Line

To build a unit object in the command line:

1. Key in the number. (It must be a real number.)
2. Key in the _ character (press [→][_]). This activates Algebraic-entry mode.
3. Key in the unit expression as you would an algebraic expression, using the [×], [÷], and [←][( )] keys as required. To key in a unit name, either press the corresponding menu key, or spell the unit name. 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.)

Example: Creating a Unit Object in the Command Line. Enter the unit object `8_Btu/(ft^2*h*°F)`.

Key in the number and the _ character. Then key in the unit expression. (To type °, press [α] [→][6].) Then enter the unit object.

8 [→][_]
Btu [÷] [←][( )] ft [y^x] 2
[×] h [×] °F
[ENTER]



By spelling unit names, as in the previous example, you can create a unit object without switching between submenus in the UNITS Catalog menu. However, you'll often find that fewer total keystrokes are required if you switch between submenus and use the keys to echo the unit name. In addition, use of the menu keys eliminates errors resulting from incorrect spelling or incorrect use of uppercase or lowercase.

Reviewing Unit Names

You can check the correct spelling and case of any unit by selecting the corresponding page in the UNITS Catalog menu and pressing ◀ REVIEW. A temporary display lists each unit on that menu page.

Example: Reviewing Unit Names. Check the correct spelling and case for the unit corresponding to the FT*LB key in the UNITS ENRG submenu.

Select the ENRG submenu and review the unit names.

◀ UNITS NXT ENRG
◀ REVIEW



Press ATTN to return to the stack display.

Unit Objects in Algebraics

Unit objects are allowed in algebraics. 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.

Functions in unit objects follow this precedence order:

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

Thus, `7_m/s^2` is 7 meters per square second; `7_(m/s)^2` is 7 square meters per square second.

+ 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`.

Unit Prefixes

Unit prefixes are letters that you can type in front of a unit name to indicate powers of ten. For example, mA means "milliamp" (amp $\times 10^{-3}$). The following table lists allowable prefixes.

Unit Prefixes

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

(To key in μ, press [α] [→] [STO].)

Most prefixes used by the HP 48 correspond with standard SI notation. There is 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 possible combinations that match built-in units are `Pa`, `da`, `cd`, `ph`, `flam`, `nmi`, `mph`, `kph`, `ct`, `pt`, `ft`, `au`, and `cu`.

Unit-Object Conversion

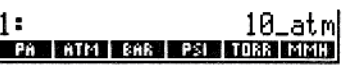
Unit-Object Conversion in the UNITS Catalog Menu

The UNITS Catalog menu lets you convert the unit object in stack level 1 to any dimensionally consistent unit in the menu simply by pressing the corresponding *left-shifted* menu key.

Example 1: Unit-Object Conversion in the UNITS Catalog Menu. Convert `10_atm` (atmospheres) to `inHg` (inches of mercury).

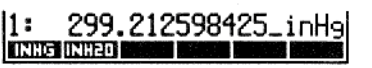
Select the UNITS Catalog menu, and then the PRESS submenu. Create the unit object `10_atm`.

[←] [UNITS]
[NXT] PRESS
10 ATM



Convert to inches of mercury.

[NXT] [←] INHG



The new unit object is returned to level 1.

Example 2: Unit-Object Conversion in the UNITS Catalog Menu. Convert `6_ft*lbf/s` (foot-pound force per second) to `W` (watts).

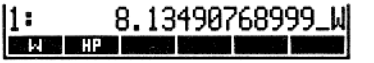
Enter the unit object.

6 ▶ _ ◀ UNITS NXT
ENRG FT*LB
◀ UNITS TIME ▶ S



Select the POWER submenu and convert to watts.

◀ UNITS NXT POWR
◀ W



The new unit object is returned to level 1.

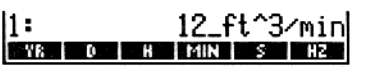
Unit-Object Conversion with CONVERT

You can execute the CONVERT command (▶ UNITS CONV) to do *any* conversion between dimensionally consistent unit expressions. CONVERT takes two arguments from the stack: The level 2 argument is the original unit object; the level 1 argument is a unit object that contains the new unit expression. The number part of the level 1 unit object is ignored.

Example: Unit-Object Conversion on the Stack. Convert `12_ft^3/min` (cubic feet per minute) to `qt/h` (quarts per hour). Since `qt/h` is not in the UNITS Catalog menu, you must explicitly execute CONVERT to do the conversion.

Enter the unit object.

12 ◀ UNITS VOL FT^3
◀ UNITS TIME ▶ MIN



Put the new unit expression on the stack, appended to any number. (The number is ignored.)

1 ▶ LAST MENU NXT QT
▶ LAST MENU ▶ H



Execute the conversion.

▶ UNITS CONV



(Note the use of [LAST MENU] to bypass the main UNITS Catalog menu and directly select the previous submenu.)

Unit-Object Conversion in the CST Menu

If you often execute a specific unit conversion, you may find it convenient to execute that conversion from the CST menu, particularly if the unit expressions are not in the UNITS Catalog menu. (Chapter 15, "Customizing the Calculator," tells you how to build a custom menu.) To execute unit conversions in the CST menu, place unit objects with the desired unit expressions in the CST-menu list. The number parts of the unit objects are ignored when conversions are executed. Once you have placed the unit expressions in the CST menu, you execute unit conversions just as you do in the UNITS Catalog menu:

1. Select the CST menu (press [CST]).
2. Put the unit object on the stack.
3. Press the left-shifted menu key corresponding to the desired unit expression.

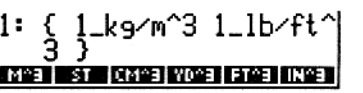
The new unit object is returned to level 1.

Example: Unit-Object Conversion in the CST Menu. Suppose you often execute unit conversions between `kg/m^3` (kilograms per cubic meter) and `lb/ft^3` (pounds per cubic foot).

Part 1. Put the unit expressions in the CST menu. (This example assumes that there are no previous entries in the CST-menu list.)

Build a list that contains the two unit objects. When you press [{ }], the HP 48 switches to Program-entry mode, so you'll have to key in the _ and / characters.

[{ }]
1 [_] [UNITS] MASS KG [÷]
[UNITS] VOL M^3 [SPC]
1 [_] [LAST MENU] LB [÷]
[LAST MENU] FT^3
[ENTER]



Store the list in the variable *CST* and display the CST menu.

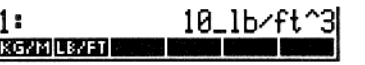
[MODES] MENU



Part 2. Convert `10_lb/ft^3` to `kg/m^3`.

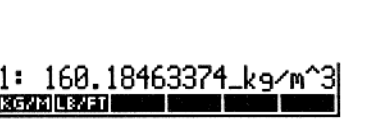
Enter the unit object.

10 LB/FT



Convert to kilograms per cubic meter.

◀ KG/M



Conversion to SI Base Units

UBASE (▶ UNITS UBASE) converts a compound unit into its equivalent SI base units. UBASE takes as its argument a unit object from level 1 of the stack.

Example 1: Conversion to SI Base Units. Convert `8.3_Pa` (Pascals) into SI base units.

Enter the unit object, select the UNITS Command menu and execute the unit object conversion.

◀ UNITS NXT PRESS
8.3 PA
▶ UNITS UBASE



Example 2: Conversion to SI Base Units. Convert `30_knot` into SI base units.

Enter the unit object, select the UNITS Command menu and execute the unit object conversion.

◀ UNITS SPEED
30 KNOT
▶ UNITS UBASE



Temperature Conversion

Conversions between the four temperature scales (K, °C, °F, and °R) involve additive constants as well as multiplicative factors. If both unit expressions consist of a single, unprefixed temperature unit with no exponent, CONVERT performs an *absolute* temperature scale conversion, including the additive constants.

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: Temperature Conversion. Part 1. Convert `25_°C` to `°F`.

Enter the unit object and execute the conversion.

◀ UNITS NXT TEMP
25 °C ◀ °F



Part 2. Convert `20_°C/min` to `°F/s`.

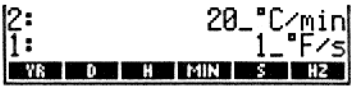
Create the unit object `20_°C/min`.

◀ UNITS NXT TEMP
20 °C
◀ UNITS TIME ▶ MIN



Enter a unit object consisting of the new units and any number.

▶ LAST MENU 1 °F
▶ LAST MENU ▶ S



Execute the conversion.

▶ UNITS CONV



Dimensionless Units of Angle

Plane and solid angles are *dimensionless*. You can use the following dimensionless units as constants in your unit expressions; however the HP 48 can't check for dimensional consistency in dimensionless units.

Dimensionless Unit	Unit Name	Value
Arcmin	arcmin	$^1/_{21600}$ unit circle
Arcsec	arcs	$^1/_{1296000}$ unit circle
Degree	°	$^1/_{360}$ unit circle
Grad	grad	$^1/_{400}$ unit circle
Radian	r	$^1/_{2\pi}$ unit circle
Steradian	sr	$^1/_{4\pi}$ unit sphere

Some photometric units are defined in terms of steradians. These units include a factor of $^1/_{4\pi}$ in their numerical values. Because this factor is dimensionless, the HP 48 can't check for its presence or absence. Therefore, to convert between photometric units that include this factor and photometric units that don't, you should include the dimensionless unit sr. The following table lists photometric units according to whether their definition includes steradians.

Include Steradians	Do Not Include Steradians
Lumen (lm)	Candela (cd)
Lux (lx)	Footlambert (flam)
Phot (ph)	Lambert (lam)
Footcandle (fc)	Stilb (sb)

To convert between photometric units in the same column, the `sr` unit is *not* required. To convert between photometric units in different columns, you must divide the unit in the left column by `sr` or multiply the unit in the right column by `sr`. *Be sure to do so, because the HP 48 can't check that your units are consistent.* Some examples of consistent photometric units are:

- `lm` is consistent with `cd*sr`.
- `fc/sr` is consistent with `flam`.
- `lm/sr*m^2` is consistent with `lam`.

Unit-Expression Factoring

UFACT (→ UNITS UFACT) factors one unit from a unit expression, returning a unit object whose unit expression consists of the factored unit and remainder SI base units. UFACT takes two arguments from the stack: a unit object from level 2 and a unit object from level 1. The level 1 unit object consists of any number and the unit to be factored from the level 2 unit object.

Example: Unit-Expression Factoring. Factor `3.5_kg*m^2/s^2` with respect to `N` (Newtons).

Enter the unit object.

← UNITS MASS 3.5 KG
← UNITS AREA M^2
← UNITS TIME
→ S → S



Key in the unit to be factored, appended to any number.

1 ← UNITS NXT FORCE N



Factor the level 2 unit object.

→ UNITS UFACT



Unit-Object Arithmetic

The HP 48 lets you execute many arithmetic operations with unit objects, just as you execute them with real numbers. Unit objects can be:

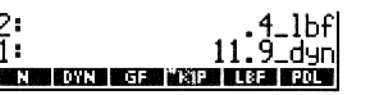
- Added and subtracted.
- Multiplied and divided.
- Inverted.
- Raised to a power.
- Used in percentage calculations.
- Compared in value to each other.

Several additional math operations work only on the number part of the unit object.

Example: Unit-Object Addition. Calculate the sum of `0.4_lbf` and `11.9_dyne`.

Enter the unit objects.

◀ UNITS
NXT FORCE
.4 LBF
11.9 DYN



Add the unit objects. The unit conversion is done automatically for you.

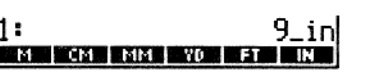
+



Example: Unit-Object Subtraction. Subtract `39_in` from `4_ft`.

Enter the unit objects and subtract. The unit conversion is done automatically for you.

◀ UNITS LENG
4 FT
39 IN
−



Adding and Subtracting Temperature Units. Pure temperature units are converted to absolute temperatures before adding or subtracting, and the sum or difference is then converted to the level 1 unit. For example `32_°F 0_°C +` returns `273.15_°C`.

Example: Unit-Object Multiplication and Division by Real Numbers. Multiply `12_mph` by 10, then divide by 6.

Enter the unit object and multiply by 10.

◀ UNITS SPEED
12 MPH
10 ×

```
1:                    120_mph
M/S  CM/S  FT/S  KPH  MPH  KNOT
```

Divide by 6.

6 ÷

```
1:                     20_mph
M/S  CM/S  FT/S  KPH  MPH  KNOT
```

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

Enter the first unit object. Enter the second unit object and multiply.

◀ UNITS LENG
50 FT
45 FT
×

```
1:                  2250_ft^2
M  CM  MM  YD  FT  IN
```

Key in the third unit object and divide.

◀ UNITS TIME
3.2 D
÷

```
1:              703.125_ft^2/d
YR  D  H  MIN  S  HZ
```

Example: Finding the Reciprocal of a Unit Object. Find the reciprocal of `11.4_g*cm/s^2`.

Enter the unit object and find the reciprocal.

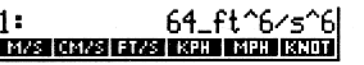
11.4 ◀ UNITS MASS G
◀ UNITS LENG CM
◀ UNITS TIME
▶ S ▶ S
1/x

```
1: 8.77192982456E-2_s^
   2/(g*cm)
YR  D  H  MIN  S  HZ
```

Example: Raising a Unit Object to a Power. Raise `2_ft/s` to the sixth power. Find the square root of the result. Then find the cube root of that result.

Enter the unit object and raise the unit object to the sixth power.

2 ◀ UNITS SPEED FT/S
6 y^x



Note that the unit-expression exponents of the result are six times the unit expression exponents of the original unit object.

Now find the square root of the result.

$\sqrt{x}$



Find the cube root of the result.

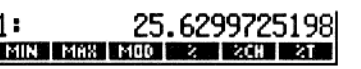
3 ▶ $\sqrt[y]{}$



Example: Percent Calculations with Unit Objects. `4.2_cm^3` is what percent of `1_in^3`?

Enter the unit objects in the correct order and execute the calculation.

◀ UNITS VOL 1 IN^3
4.2 CM^3
MTH PARTS NXT %T

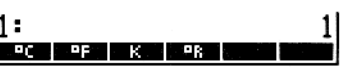


When the unit expression is temperature only, % converts to absolute temperature, executes the calculation, and then converts back to the original units. %CH and %T similarly convert temperature units before executing the calculation.

Example: Comparing Unit Objects. Determine if 12 °C is greater than 52 °F.

Enter the unit objects in the correct order and execute the test.

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



1 is returned to the stack, indicating that the test is true. (12 °C is 53.6 °F.)

Trigonometric Operations with Unit Objects. You can execute the trigonometric operations SIN, COS, and TAN on unit objects whose unit expression is a *planar angular* unit. Planar angular units are radians (`r`), degrees (`°`), grads (`grad`), arc-minutes (`arcmin`), and arc-seconds (`arcs`). The result is a dimensionless real number.

Example: Trigonometric Operations with Unit Objects. Calculate the sine of 45°.

Select the UNITS ANGL menu, enter the unit object, and execute SIN.

◀ UNITS NXT NXT ANGL
45 ° SIN



In algebraic syntax, calculate the tangent of 40 grad.

' TAN 40 ▶ _ GRAD
EVAL



Arithmetic Operations on the Number Part of a Unit Object. The following functions, described in detail in chapter 9, "Common Math Functions," operate on the number part of a unit object. Each function returns a unit object, leaving the unit-expression part of the argument unit object unchanged:

ABS	IP
CEIL	NEG
FLOOR	RND
FP	TRNC

The SIGN function, also described in chapter 9, returns a *number* that indicates the sign of the number; +1 for a positive number, −1 for a negative number, and 0 if the number is 0.

Building Unit Objects with the EquationWriter Application

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 (see page 235 in chapter 16).

Example: Building a Unit Object with the EquationWriter Application. Use the EquationWriter application to create the unit object:

```
32 W/m^2*°C
```

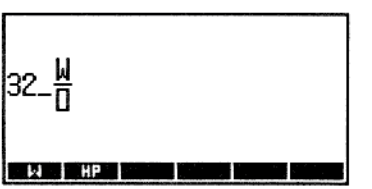
Select the EquationWriter application. Key in the number and start the unit expression.

◀ EQUATION
32 ▶ _



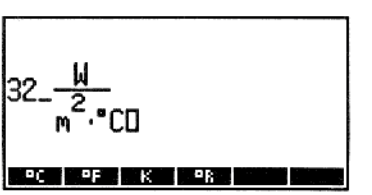
Key in the numerator of the unit expression.

▲ ◀ UNITS NXT POWR
W ▶



Key in the denominator.

◀ UNITS AREA M^2 ▶
◀ UNITS NXT TEMP °C



Put the unit object on the stack.

ENTER



Viewing a Unit Object in the EquationWriter Application. Pressing ▼ when a unit object is in level 1 displays the unit object in the EquationWriter application.

User-Defined Units

Suppose you often use "week" as a unit of time. The unit "week" is not contained in the UNITS Catalog menu. However, you can create a *user-defined* unit for "week" that behaves just like a built-in unit.

To create a user-defined unit:

1. Enter a unit object representing the value of the new unit in terms of built-in or previously defined units.
2. Store the unit object in a variable that names the new unit.
3. Create a unit object consisting of any number and the user-defined unit.

Example: Creating and Using a User-Defined Unit. Part 1. Use the built-in unit d (day) to create the user-defined unit WEEK.

Enter the unit object 7_d. Store the unit object in variable *WEEK*, then enter a list containing the unit object 1_WEEK.

```
←UNITS TIME 7 D
' WEEK STO
←{ } VAR 1 →_ WEEK
ENTER
```

```
1:                { 1_WEEK }
WEEK
```

Store the list in the CST menu and display the menu.

```
→MODES MENU
```

```
WEEK
```

Part 2. Convert 14 days to weeks.

Enter the unit object. Select the CST menu and execute the conversion.

```
←UNITS TIME 14 D
CST ← WEEK
```

```
1:                    2_WEEK
WEEK
```

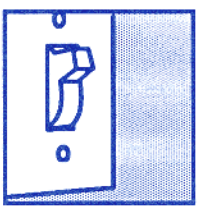
Prefixing User-Defined Units. You can prefix a user-defined unit. However, conflicts between user-defined units (prefixed or otherwise) and built-in units are resolved in favor of the built-in unit.

Additional Commands for Unit Objects

Keys	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.

14

Binary Arithmetic



The HP 48 enables you to do binary arithmetic. This chapter covers the commands and methods for manipulating binary integers.

Binary integer objects contain from 1 to 64 bits, depending on the current *wordsize*. They can be entered and displayed in decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) base. 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`.

Setting the Wordsize

The wordsize is the number of bits used to represent binary integers. The wordsize can range from 1 through 64 bits; its default is 64 bits. To set the wordsize, key in a number from 1 to 64 and execute STWS ([MTH] BASE STWS). (Fractional numbers are rounded to the nearest integer.) To recall the current wordsize, execute RCWS ([MTH] BASE RCWS). If you enter a binary integer that exceeds the current wordsize, the number is displayed with its most significant bits truncated. Any bits over 64 are lost; any bits from the current wordsize to 64 are "hidden" and

can be displayed by increasing the wordsize. However, hidden bits are not used in calculations and are lost when a command is executed on a binary integer.

Also, the wordsize controls the results returned by arithmetic operations and other commands. If an argument exceeds the current wordsize, it is truncated (the most significant bits are lost) to the current wordsize before the command is executed. If necessary, results are also truncated.

Selecting the Base

Binary integers are displayed in decimal, hexadecimal, octal, or binary base. The default base is decimal. To change the current base, display the MTH BASE menu ([MTH] BASE) and press HEX (hexadecimal), DEC (decimal), OCT (octal), or BIN (binary). The box in one of the menu labels identifies the current base.

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 E.)

The choice of current base has no effect on the internal representation of binary integers.

Entering Binary Integers

A binary integer is identified by the # delimiter preceding it.

Example: Entering and Displaying Binary Integers.

Enter the address $24FF_{16}$.

[MTH] BASE HEX
[→] [#] 24FF [ENTER]



The *base marker* h indicates that the binary integer is being displayed in hexadecimal base. You do not need to key in the base marker when entering a number in the current base.

Now, display $24FF_{16}$ in octal base.

OCT



To enter a number that is not in the current base, type the base marker after the digits.

Enter 101101_2 while the current base is octal.

[→][#] 101101b [ENTER]



The number is displayed on the stack in octal base.

Calculations with Binary Integers

Example: Subtraction. Calculate $46AF_{16} - 33D_{16}$.

Switch to hexadecimal base and enter the two numbers.

HEX
[→][#] 46AF [ENTER]
[→][#] 33D [ENTER]



Execute the [−] command.

[−]



Example: Division. If division produces a remainder, only the integer portion of the result is retained.

Divide 64d by 5d.

DEC
[→][#] 64
[→][#] 5 [÷]



The remainder of 4d is lost.

Additional Binary Integer Commands

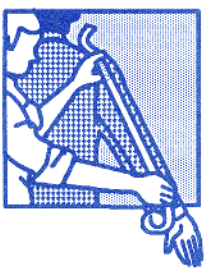
The following table contains commands from the MTH BASE menu ([MTH] BASE) that are useful for manipulating binary integer objects. Unless otherwise stated, each example assumes the wordsize is set to 24.

Command/Description	Example	
	Input	Output
AND Logical bit-by-bit AND of two arguments.	2: # 1100b 1: # 1010b	1: # 1000b
ASR Arithmetic Shift Right. Performs 1 bit arithmetic right shift. The most significant bit is regenerated.	1:# 1100010b 1: # 800000h	1: # 110001b 1: # C00000h
B→R Binary to Real. Converts a binary integer to its real integer equivalent.	1: # 755o	1: 493
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: # F0F0F0h	1: # F0F0Fh
OR Logical bit-by-bit OR of two arguments.	2: # 1100b 1: # 1010b	1: # 1110b
R→B Real to Binary. Converts a real integer to its binary integer equivalent.	1: 10	1: # 1010b

Command/Description	Example	
	Input	Output
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`
XOR Logical bit-by-bit exclusive OR of the arguments.	`2: # 1100b` `1: # 1010b`	`1: # 110b`

15

Customizing the Calculator



This chapter covers:

- Creating and using your own custom menus.
- Defining your own functionality for the keyboard (and invoking this *user keyboard*).
- Setting and clearing the system flags. The system flags control many of the calculator's *modes*.

Custom (CST) 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. The menu is displayed by pressing [CST].

Creating a Custom Menu

When you press [CST], the HP 48 uses the contents of a *reserved* variable named *CST* to display the custom menu. (*CST* is reserved by the HP 48 because its contents determine the contents of the custom menu.) So, the way to create a custom menu involves creating a variable *CST* that contains the objects you want in your menu.

Like other variables, *CST* can be created, along with its associated custom menu, for each directory in memory.

To create and display a custom menu in the current directory:

1. Enter a list containing the objects you want in the menu.
2. Execute MENU ([→][MODES] MENU). MENU stores the contents of the list in *CST* and displays the custom menu.

Alternatively, if you want to create but not immediately display a custom menu, you can store the custom-menu list in *CST* just like you would store a list in any variable—by entering the list on the stack and pressing the left-shifted menu key for the variable. The MODES Customization menu always contains a menu label for *CST* ([→][MODES] CST).

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 from one custom menu to another by simply storing a new name in *CST*.

Custom Menu Functionality

Objects in the custom menu usually have the same functionality they do in built-in menus. For example:

- Names behave like the 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 side of the label; pressing the menu key switches to that directory.
- Unit objects act like unit catalog entries. For instance, they have their left-shifted conversion capability.
- String keys echo the string.

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.

Example: Creating and Using a Custom Menu. 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*.

Enter a list of the objects.

◀ { } PRG OBJ →TAG
1 ▶ _ m y^x 3
▶ " " VOLUME ▶
CST ENTER



Store the list in *CST* and display the custom menu.

▶ MODES MENU



Convert 1075 cm^3 to m^3.

1075 ▶ _ cm y^x 3 ENTER
◀ M^3



Enter the string `"VOLUME"`.

▶ " " VOLU ENTER



Create a tagged object from the contents of levels 2 and 1.

→TAG



Display the current contents of *CST*.

CST



Enhancing Custom Menus

You can enhance the basic custom menu to:

- Have menu labels different from the underlying name, command, or typing aid.
- Specify different actions for the unshifted, left-shifted, and right-shifted keys.

Providing Different Menu Labels. You can supply menu labels that differ from the objects themselves by embedding within your custom list an inner list of the form:

« "*label*" *object* »

For example, storing:

```
« →TAG 1_m^3 « "VOL" "VOLUME"» « "CUST" CST » »
```

in *CST* produces a custom menu with the same functionality as the menu in the previous example, except that the labels are →TAG, M^3, VOL, and CUST.

Providing Shifted Functionality. To provide different shifted actions for custom menu keys, specify within the inner list the three actions (objects) in yet another list. The order in this additional embedded list is the unshifted action, the left-shifted action, and then the right-shifted action. (You must specify the unshifted action in order to have the shifted actions.) For example, suppose you want the custom menu key VOL to contain 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.

Here is the list you use as an argument for the MENU command:

```
{ { "VOL" { « 'VBOX' STO » « * * » "VOLUME" } } }
```

Creating a Temporary Menu

The TMENU command creates a temporary menu without overwriting the contents of the variable *CST*. Temporary menus are most useful in programming and are covered in chapter 29.

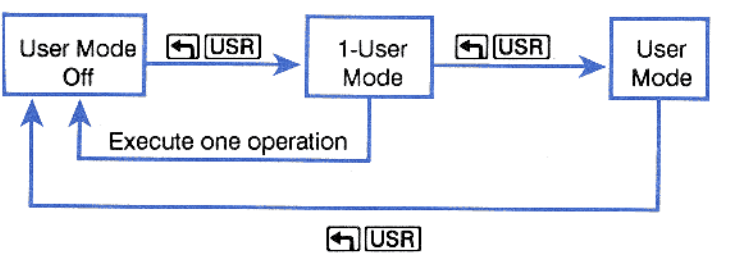
The User Keyboard

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 is active when the calculator is in *User mode*.

The commands for creating and changing the user keyboard are located in the MODES Customization menu (→ MODES).

User Modes

The ← USR key is a three-way switch. With User mode off, press it once to activate 1-User mode; press it a second time to activate User mode; and press it a third time to turn User mode off.



In 1-User mode (1USR annunciator), the user keyboard is active for one operation. In User mode (USER annunciator), the user keyboard remains active until you press ← USR to turn it off.

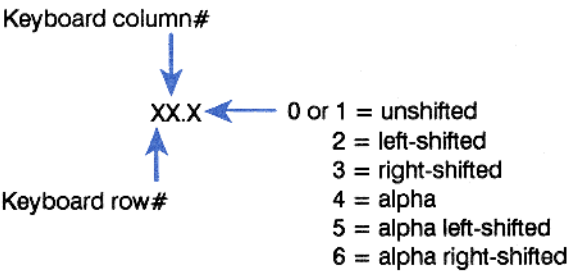
If you set flag −61, [←][USR] becomes a two-way toggle between User mode on and User mode off. The keystrokes to set flag −61 are 61 [+/−] [→][MODES] SF.

Making User Key Assignments

The ASN (assign) and STOKEYS (store keys) commands assign user functionality to the keyboard. ASN ([→][MODES] ASN) is used to make a single user-key assignment, while STOKEYS ([→][MODES] STOK) is used to make multiple assignments.

Making a Single Assignment. ASN takes two arguments:

- In level 2, the object to be assigned to the key.
- In level 1, the three-digit location number that defines where the key is on the keyboard as described in the following diagram:



For example, executing ASN with DUP2 in level 2 and 36.2 in level 1 assigns DUP2 to [←][SWAP] without otherwise affecting the user keyboard. ([←][SWAP] has a location of 36.2 because it is three rows down, six columns across, and left-shifted.) Once this user key is assigned, and when the calculator is in 1-User or User mode, pressing [←][SWAP] executes DUP2.

Making Multiple Assignments. STOKEYS enables you to make several user-key assignments at one time. Its argument takes the form of a list with the syntax:

{ S *definition*$_1$ *location*$_1$ *definition*$_2$ *location*$_2$ ... }

where:

- S (optional) indicates unassigned keys retain their standard definitions. The S is necessary only when the unassigned keys have been previously disabled by the DELKEYS command.
- *definition* is any object. When the user-key is pressed, that object is executed.
- *location* is the three-digit number corresponding to the position of the key on the keyboard (see the previous diagram).

Once a user assignment has been made to a key, it remains in effect until the key is reassigned with another ASN or STOKEYS command, or until the key is cleared with a DELKEYS command.

Example: User Keyboard Assignments. Use the STOKEYS command to:

- Assign the variable *ABC* containing { A B C } to user-key A.
- Assign the program « OBJ→ DROP » to user-key ↱ □.
- Assign the command DROP2 to user-key ↰ DROP.
- Assign the string (typing aid) "HEIGHT" to user-key h.

Create the variable *ABC* containing the list { A B C } and display the VAR menu.

↰ { } A SPC B SPC C ENTER
' ABC STO VAR



Enter the argument for the STOKEYS command.

↰ { } ABC SPC 11.4
↰ « » PRG OBJ OBJ→
↰ DROP ▶ ▶ 75.3
PRG STK NXT DROP2 55.2
↱ " " HEIGHT ▶ 22.5 ENTER



Execute STOKEYS and activate the user keyboard.

↱ MODES STOK
↰ USR ↰ USR



Now, retrieve the list « A B C » and separate it into its components.

A
→ _



Execute DROP2, and put the string "HEIGHT" on the stack.

← DROP
→ " " h ENTER



Press ← USR again to restore normal keyboard function.

Clearing User Key Assignments

The DELKEYS command (→ MODES DELK) clears one or more key assignments. It accepts a single argument, either:

- A three-digit key location (described on page 217). That single key is cleared.
- A list of locations. Each of those keys is cleared.
- An S. The standard key definitions are cleared.
- A zero (0). All user keys are cleared. If unassigned keys were deactivated, they are reactivated.

Reactivating a Single Standard Key

If DELKEYS is executed with S as its argument, the standard keys have no functionality while the HP 48 is in User mode until they are reassigned or until all user keys are cleared. Once the standard keys are "dead," you can reassign an individual key with its standard definition using the ASN command with a special argument, SKEY. These are the steps to reactivate a single standard key:

1. Enter 'SKEY' on the stack.
2. Enter the three-digit location (described on page 217) of the standard key to reactivate.
3. Press → MODES ASN.

For example, with 'SKEY' in level 2 and 16.1 in level 1, pressing ASN reactivates the NXT key for use within User mode.

You can repeat this procedure for as many standard keys as you like. Then, when the HP 48 is in User mode, you will have your user keys active, as well as any standard keys you've reactivated.

You can also include SKEY in the list of key assignments for the STOKEYS command.

Recalling and Editing User Key Assignments

The RCLKEYS command ([→] MODES RCLK) returns to level 1 a list of all the current user key assignments, including the letter S if standard keys are active in User mode.

To edit the assignments, execute RCLKEYS, edit the list, and then execute STOKEYS to reassign the keys.



Note

As "old" user key assignments are deleted, a small amount of memory is reserved by the old assignments. Over time, the memory occupied by old user keys can become significant — each old key assignment reserves between 2.5 and 15 bytes of memory. You can free this memory for other uses by *packing* your user key assignments. The following command sequence packs the user-key assignments into their most compact form:

```
RCLKEYS  0 DELKEYS  STOKEYS
```

Other Customizing Operations

The MODES Menu

The multiple pages of the MODES menu ([←] MODES) contain additional operations that let you customize the way your calculator operates. (When the menu label has a box in the label, for instance SYM■, the operation is active.)

MODES Operations

Keys	Description
◀ MODES:	
SYM	Switches between symbolic (box in label) and numerical evaluation.
BEEP	Switches between errors beeping (box in label) and not beeping.
STK	Switches between saving (box in label) and not saving the last stack. Affects the action of ◀ LAST STACK.
ARG	Switches between saving (box in label) and not saving the last arguments. Affects the action of ▶ LAST ARG.
CMD	Switches between saving (box in label) and not saving in memory the last command line. Affects the action of ◀ LAST CMD.
CNCT	Switches between drawing a continuous line to connect plotted points (box in label) and plotting points only.
ML	Switches between displaying a multiline level 1 as multiple lines (box in label) and as a single line followed by an ellipsis.
CLK	Switches between displaying a clock (box in label) and not displaying a clock.
FM,	Switches between decimal fraction mark and comma fraction mark (box in label).

System Flags

The HP 48 provides a number of modes that also let you customize its operating environment. 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 E.

The commands for setting, clearing, and testing flags are in the MODES Customization menu (→ MODES). (They are duplicated in the PRG TEST menu.) They take flag numbers as arguments. For example, the keystrokes 20 +/− → MODES NXT SF set system flag −20.

Flag Tests

Keys	Programmable Command	Description
→ MODES (pages 2 and 3) or PRG TEST (page 3):		
SF	SF	Sets the flag.
CF	CF	Clears the flag.
FS?	FS?	Returns true (1) if flag is set and false (0) if flag is clear.
FC?	FC?	Returns true (1) if flag is clear and false (0) if flag is set.
FS?C	FS?C	Tests flag (returns true (1) if set and false (0) if clear), then clears the flag.
FC?C	FC?C	Tests flag (returns true (1) if clear and false (0) if set), then clears the flag.

Setting Automatic Alpha Lock. Ordinarily, Alpha-entry mode is locked by pressing α twice in a row. You can choose instead to have a single press of α automatically activate alpha lock. To select Automatic Alpha Lock mode, set system flag −60.

Setting User Mode. Pressing ↰ USR once normally puts your calculator in User mode for one keystroke; pressing it twice in a row locks in User mode until you press it a third time. If you prefer to have User mode "lock in" on the first press, set flag −61.

Evaluating Symbolic Constants. Symbolic constants (e, i, π, MAXR, and MINR) retain their symbolic form when evaluated. If you want them to be automatically evaluated using their HP 48 numerical representations, set flag −2.

More Uses of System Flags. The previous examples show just a few of the ways you can use flags to customize the way your HP 48 operates. You can also use flags to affect the display, math operations, printing, plotting, time management, and various other operations. For the complete listing of all 64 system flags and what they affect, see appendix E.

Part 3

Power Tools

16

The EquationWriter Application



The EquationWriter application lets you enter and review algebraic expressions and equations in the form most familiar to you—the way they appear printed in books and journals, and the way you write them with pencil and paper.

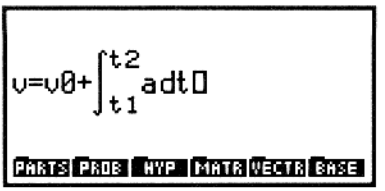
For example, here's an equation taken from a physics text:

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

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:

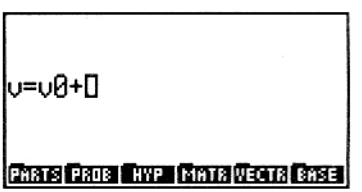


Example: The EquationWriter Application. Use the EquationWriter application to key in the previous equation.

(If you make a mistake while you're keying in the equation, press ◀ to backspace to the error. Note that the HP 48 may take several seconds to redisplay the equation after you've pressed ◀. Later in this chapter, you'll learn how to edit an equation in the command line.)

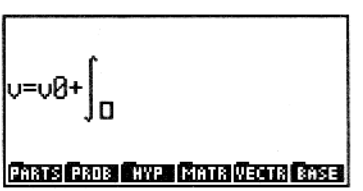
Select the EquationWriter application and key in the equation up to the ∫ sign.

◀ EQUATION
v ◀ = v0 +



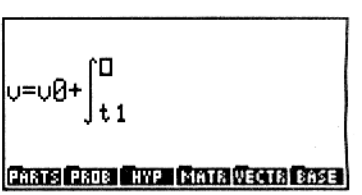
Key in the integral sign.

▶ ∫



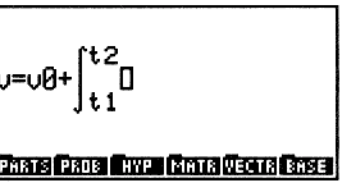
Key in the lower limit and move the cursor to the upper limit.

t1 ▶



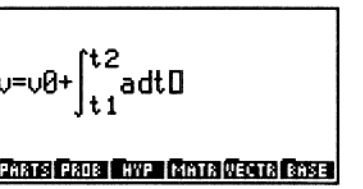
Key in the upper limit and move the cursor to the beginning of the integrand.

t2 ▶



Key in the integrand and the variable of integration.

a ▶ t



Put the equation on the stack.

ENTER



How the EquationWriter Application Is Organized

The EquationWriter application is a special environment where the keyboard is redefined and limited to special operations. You cannot do stack calculations 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 menu — however, only those keys that correspond to algebraic functions are active. Like the function keys on the keyboard, the menu keys do not execute the corresponding function; they simply enter the function name into the equation.

Other keys on the keyboard are defined as follows:

Operations in the EquationWriter Application

[▲]	Starts a numerator.
[▶] or [▼]	Ends a subexpression. ([↱][▶] or [↱][▼] ends all pending subexpressions.)
[◀]	Invokes the EquationWriter *Selection environment*.
[↰][( )]	Enters (to start a parenthesized term. [▶] (or [▼]) ends the parenthesized term.
[SPC]	Enters the current separator (, or ;) for multiple parenthetical arguments of functions and the terms of complex numbers.
[EVAL]	Evaluates the equation and exits the EquationWriter application.
[ENTER]	Returns the equation to the stack and exits the EquationWriter application.
[ATTN]	Exits the EquationWriter application. The equation is not saved.
[↰][GRAPH]	Invokes *scrolling* mode. In scrolling mode, the menu keys are erased; if the equation is larger than the display, the cursor keys scroll the display window over the equation in the indicated direction. Pressing [↰][GRAPH] again (or [ATTN]) restores the menu keys and puts the cursor at the end of the equation.
[↰][EDIT]	Returns the equation to the command line for editing. (See "Editing Equations" on page 241.)
[STO]	Returns the equation to the stack as a *graphics object*. (See "The Structure of the PLOT Application" in chapter 18 and "Working with Graphics Objects on the Stack" in chapter 19 for discussions of graphics objects.)

Operations in the EquationWriter Application (continued)

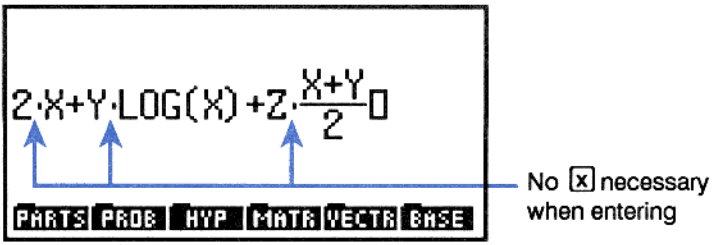
[→][CLR]	Erases the display without leaving the EquationWriter application.
[→][RCL]	Inserts the level 1 object into the equation at the cursor position. (See "Editing Equations" on page 241.)
[←][{ }]	Turns *implicit parentheses* mode off. Pressing [←][{ }] again turns implicit parentheses mode back on. (See "Turning Off Implicit Parentheses" on page 237.)
[→]["]	Returns the equation to the stack as a string.

Constructing an Equation

Numbers and Names. Numbers and names are keyed in exactly the same way they are keyed into the command line. The menu keys in the VAR menu act as typing aids for variable names.

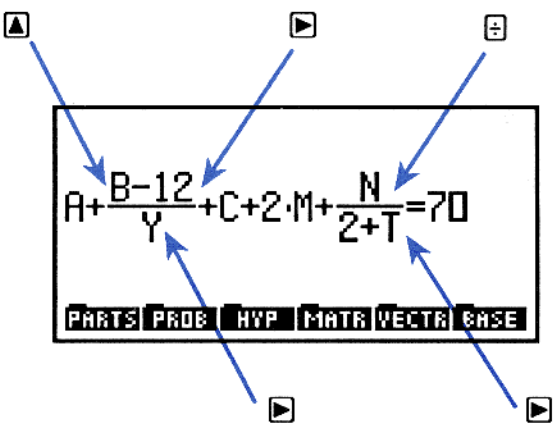
Addition, Subtraction, and Multiplication.

- [+], [−], and [×] enter +, − and ·.
- You can do *implied* multiplication (multiplication without pressing [×]) in some situations — a multiply sign (·) is automatically inserted between:
  - A number followed by an alpha character, a parenthesis, or a prefix function (a function whose argument(s) appear after its name); for example, when you press 6 [SIN].
  - An alpha character and a prefix function; for example, when you press 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, when you press B [▲].



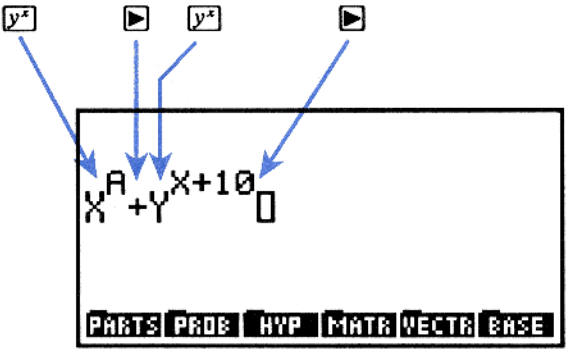
Division and Fractions.

- Primary method:
  1. [▲] starts the numerator.
  2. [▶] ends the numerator ([▼] works too).
  3. [▶] ends the denominator.
- Alternate way for fractions whose numerator consists of *one* term, or a sequence of terms with operators of precedence greater than or equal to / (divide):
  1. Type the numerator without using [▲].
  2. [÷] starts the denominator.
  3. [▶] ends the denominator ([▼] works too).



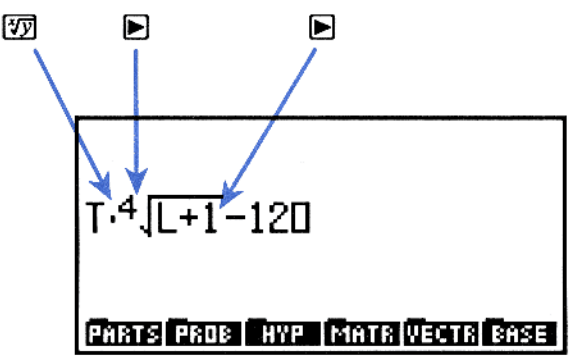
Exponents.

1. [y^x] starts the exponent.
2. [▶] ends the exponent ([▼] works too).



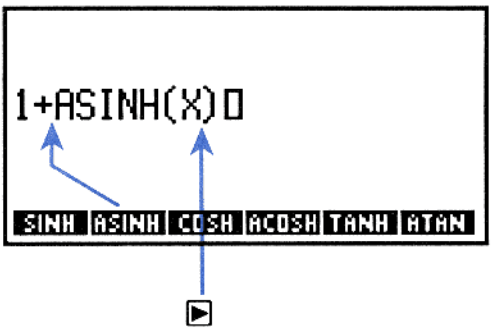
Square Root and xth Root.

- Square root:
  1. [$\sqrt{x}$] draws the √ symbol and starts the term.
  2. [▶] ends the square root term.
- *x*th root:
  1. [↱] [$\sqrt[y]{}$] starts the *x* term outside the √ symbol.
  2. [▶] draws the √ symbol and starts the *y* term inside the √ symbol.
  3. [▶] ends the *x*th root term.



Functions That Take Parenthetical Argument(s).

1. Press the function key, or type the name and [↰][( )].
2. [▶] ends the argument and types).



Parenthesized Terms.

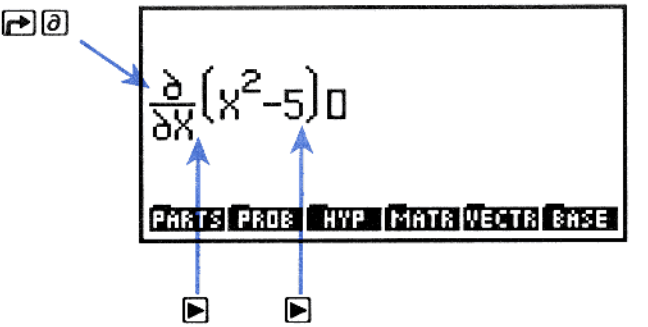
1. [↰][( )] types the (.
2. [▶] types).

Powers of 10.

1. Press [EEX] to display E.
2. For a negative power, press [+/−] to display −.
3. Key in the digits of the power.
4. Any function key ends the power.

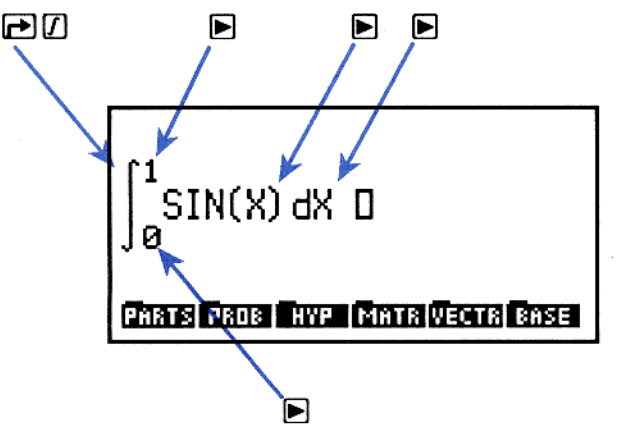
Derivatives. To key in a derivative of the form $\frac{\partial}{\partial x} f(x)$:

1. Press [↱][∂] to display $\frac{\partial}{\partial}$.
2. Key in the variable of differentiation and press [▶]. The EquationWriter application ends the denominator and displays an opening parenthesis.
3. Key in the expression.
4. Press [▶] to end the expression.



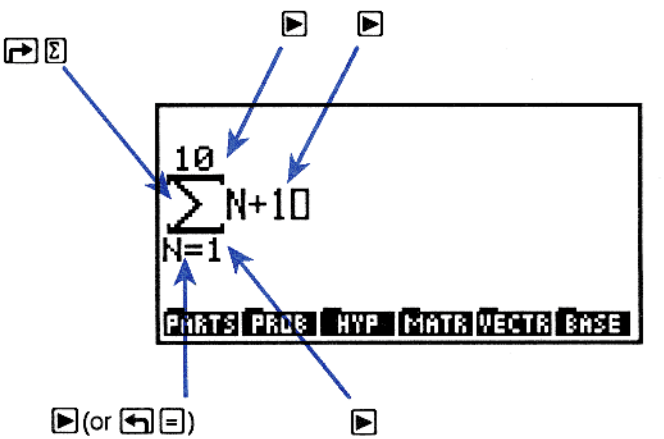
Integrals. To key in an integral of the form $\int_{lower}^{upper} f(x)\,dx$:

1. Press ↱ ƒ to display the ∫ symbol with 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 ▶. The EquationWriter application displays d.
5. Key in the variable of integration.
6. Press ▶ to complete the integral.



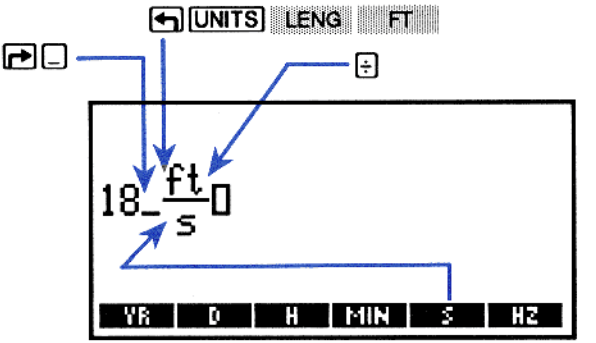
Summations. To enter summations in the form $\sum_{x=lower}^{upper} f(x)$:

1. Press [→][Σ] to display the Σ symbol. The cursor is positioned beneath the Σ.
2. Key in the summation index.
3. Press [▶] (or [←][=]) to key in the = sign.
4. Key in the initial value of the index.
5. Press [▶].
6. Key in the final value of the index and press [▶].
7. Key in the summand.
8. Press [▶] to end the summation.



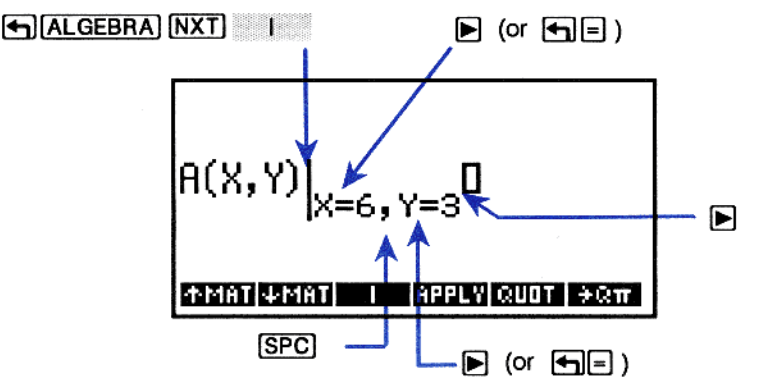
Units. Unit objects (described in chapter 13) can be constructed in the EquationWriter application:

1. Key in the number part of the unit object.
2. Press [→][_] to start the unit expression part of the unit object.
3. Key in the unit expression. Combination units are constructed as though they were algebraic expressions: separate each individual unit in the unit expression by pressing [×] or [÷]. You can key in unit names in one keystroke by pressing the corresponding menu key in the UNITS Catalog menu.
4. Press [▶] to end the expression.



The | (Where) Function. To key in the | (where) function (described in chapter 22):

1. Key in an expression with symbolic parenthetical arguments.
2. Press ALGEBRA 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 =, and SPC to key in the separator between each equation.
4. Press ▶ to end the | (where) subexpression.



Turning Off Implicit Parentheses

The arguments for ÷, √x, and y^x are normally enclosed in "invisible" parentheses, so that only ▶ (or ▼) ends the argument. Pressing ↰ { } turns off implicit parentheses (briefly displaying a message), so that any function key ends the argument. In this mode, pressing ▶ does not end the argument. (If you turn off implicit parentheses after keying in ÷, √x, or y^x, but before supplying the argument, implicit parentheses is *not* applied to those arguments.)

Disabling implicit parentheses is convenient for entering polynomials, for example, where exponents are completed by pressing the key for the function that starts the next term. Pressing ↰ { } again turns implicit parentheses back on. Leaving and then reentering the EquationWriter application also turns implicit parentheses back on.

Example 2 on page 238 demonstrates turning off implicit parentheses.

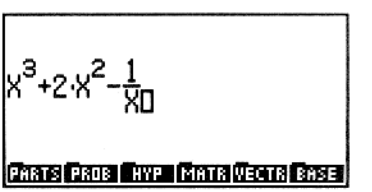
EquationWriter Application Examples

As you work these examples, remember that you can clear the display between examples without exiting the EquationWriter application by pressing ↱ CLR. If you do, then ignore the ↰ EQUATION instruction at the start of each new example.

Example 1. Key in the expression:

$$X^3 + 2X^2 - \frac{1}{X}$$

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



Example 2. Key in the same expression as in Example 1, but this time without implicit parentheses.

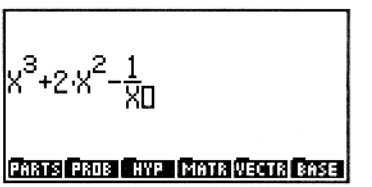
Select the EquationWriter application and turn off implicit parentheses.

[↰][EQUATION]
[↰][{ }]



Key in the expression.

X [y^x] 3 [+] 2X [y^x] 2
[−] 1 [÷] X



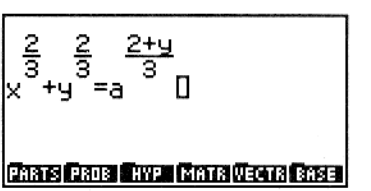
Turn implicit parentheses back on by pressing [↰][{ }].

Example 3. Key in the equation:

$$x^{\frac{2}{3}} + y^{\frac{2}{3}} = a^{\frac{2+y}{3}}$$

Activate lowercase alpha lock by pressing [α] [↰][α]. (When you press [α] and then an alpha character, it will be entered in lowercase.) Then key in the equation.

[↰][EQUATION]
[α] [↰][α]
x [y^x] 2 [÷] 3 [▶] [▶] [+]
y [y^x] 2 [÷] 3 [▶] [▶] [↰][=]
a [y^x] [▲] 2 [+] y [▶] 3
[▶][▶]

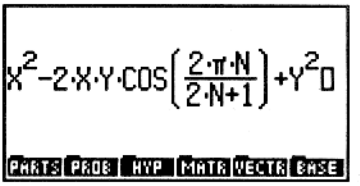


Turn off lowercase alpha lock by pressing [α] [↰][α] again.

Example 4. Key in the expression:

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

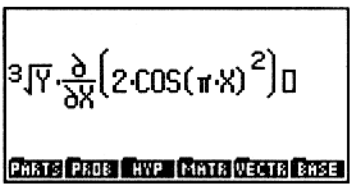
[↰] [EQUATION]
X [y^x] 2 [▶] [−]
2X [×] Y [COS]
2 [↰] [π] [×] N [÷]
2N [+] 1 [▶] [▶]
[+] Y [y^x] 2 [▶]



Example 5. Key in the expression:

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

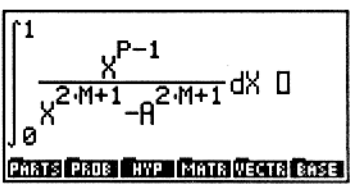
[↰] [EQUATION]
[↱] [$\sqrt[x]{y}$] 3 [▶] Y [▶]
[↱] [∂] X [▶]
2 [COS] [↰] [π] [×] X [▶]
[y^x] 2 [▶] [▶]



Example 6. Key in the expression:

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

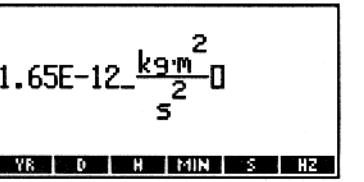
[↰] [EQUATION]
[↱] [∫] 0 [▶] 1 [▶]
X [y^x] P [−] 1 [▶] [÷]
X [y^x] 2M [+] 1 [▶]
[−] A [y^x] 2M [+] 1 [▶] [▶]
[▶] X [▶]



Example 7. Key in the expression:

$$1.65\times10^{-12}\,\frac{\text{kg}\cdot\text{m}^2}{\text{s}^2}$$

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



Viewing Algebraics and Unit Objects in the EquationWriter Application

To view a previously entered algebraic or unit object in the EquationWriter application:

1. Put the object in level 1. (If the object is stored in a variable, put the variable name in level 1.)
2. Press ▼. (If the variable name is in level 1, press ▶┐ ▼.) The algebraic or unit object is put into the EquationWriter application with the cursor positioned at the end of the expression. If the expression is larger than the display, press ◀┐ GRAPH then ◀ to scroll the display window over the rest of the expression.

Note that, depending on the length and complexity of the algebraic or unit object, the HP 48 may take one minute or more to display it in the EquationWriter application.

Editing Equations

There are several editing options available to you in the EquationWriter application:

- Backspace editing.
- Command-line editing.
- Inserting an object from the stack into the equation.
- Replacing a subexpression with an algebraic from the stack.

Backspace Editing

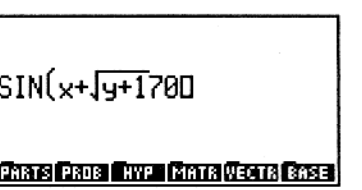
If you make a mistake while entering an expression in the EquationWriter application, you can at any time press ◀ to move the cursor back to the error, erasing characters as you go. Note however, that backspacing into a completed subexpression (an expression ended by pressing ▶) or across a function name is very slow. Usually, backspacing is appropriate only for correcting a mistyped character or digit. For more extensive editing, use command-line editing, described in the next section.

Example: Backspace Editing. Key in the expression:

$$\sin(x + \sqrt{y + 180} + z)$$

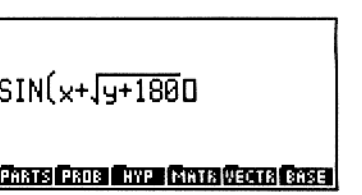
Select the EquationWriter application and start the expression. "Accidentally" key in `170` instead of `180`.

◀ EQUATION
SIN x +
√x y + 170



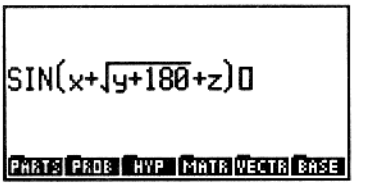
Backspace over `70`, key in the correct number, and finish the subexpression.

◀ ◀
80 ▶



Finish keying in the expression.

+ z ▶



Command-Line Editing

You can edit all or part of an equation in the command line and then return the edited version to the EquationWriter application.

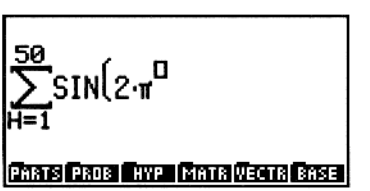
Editing the Full Equation. To return the entire equation to the command line for editing, press ↰ EDIT. If the equation ends in a subexpression that requires argument(s), those arguments must be entered before pressing ↰ EDIT.

Example: Command-Line Editing. Key in the expression:

$$\sum_{i=1}^{50} \sin(2\pi^i)$$

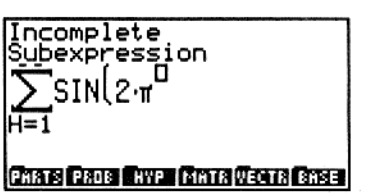
Select the EquationWriter application and key in the expression. "Accidentally" specify the series index as H instead of I.

↰ EQUATION
↱ Σ
H ▶ 1 ▶ 50 ▶
SIN 2 ↰ π y^x



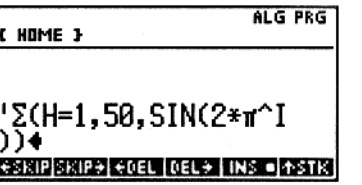
Suppose that you now realize that you need to change the H to I. Try pressing ↰ EDIT

↰ EDIT



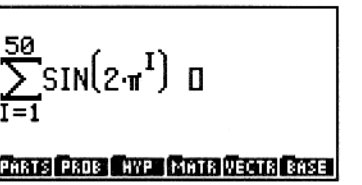
The EquationWriter application briefly displays the message `Incomplete Subexpression` and leaves the cursor at the end of the equation. Key in the index `I`, then press ▶. Now press ◀ EDIT. (The HP 48 takes several seconds to return the expression to the command line.)

I ▶ ◀ EDIT



Change the `H` to `I` and return the expression to the EquationWriter application. (The HP 48 takes about 10 seconds to return the expression to the EquationWriter application.)

▲ ▶ DEL I
ENTER



The Selection Environment. The *Selection environment* is a special part of the EquationWriter application used to specify a subexpression in the equation. It's accessed by pressing ◀ while you're in the EquationWriter application. In the following sections you'll learn how the Selection environment is used to specify a subexpression for subsequent editing.

A *subexpression* consists of a function and its arguments. The function that defines a subexpression is called the *top-level* function for that subexpression. For example, in the expression `'A+B*C/D'`, `*` is the top level function in the subexpression `'B*C'`, `/` is the top-level function in the subexpression `'B*C/D'`, and `+` is the top level function in the subexpression `'A+B*C/D'`.

In the concluding section in this chapter "A Preview of the Rules Application," you'll see how the Selection environment is used to specify a subexpression for subsequent algebraic rearrangement in the Rules application.

Editing a Subexpression. The Selection environment lets you specify a subexpression in the equation for command-line editing:

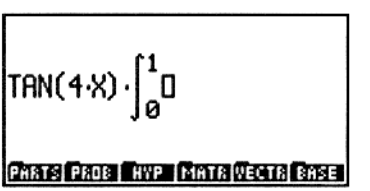
1. If the equation ends in a subexpression whose arguments are not yet keyed in, key in those arguments.
2. Press ◀. This activates the Selection menu and the *selection cursor*. The cursor initially highlights the last object in the equation.
3. Use the cursor keys to move the selection cursor to the top-level function for the subexpression you want to edit. At any time you can press EXPR to highlight the associated subexpression. (You can actually specify an individual object, a name for example, as the "subexpression".)
4. When the selection cursor is positioned to specify the desired subexpression, press EDIT. The subexpression is returned to the command line for editing.
5. After editing, press ENTER to return the revised subexpression to its position in the original equation.

Example: Editing a Subexpression. Key in the expression:

$$\tan\frac{4}{x}\int_0^1 x^y\,dx$$

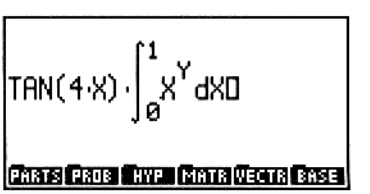
Select the EquationWriter application and start the expression. In the argument for TAN, "accidentally" press × instead of ÷.

↰ EQUATION
TAN 4 × X ▶
↱ ∫ 0 ▶ 1 ▶



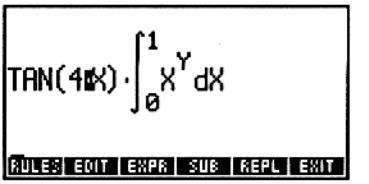
At this point you realize your mistake. However, you must enter the remaining arguments for the integral subexpression before activating the Selection menu and cursor.

X y^x Y ▶ ▶ X



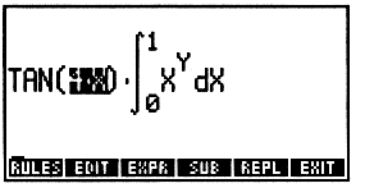
Now activate the Selection menu and cursor. Then move the cursor back to the unintended ·.

◀
then ◀ 7 times



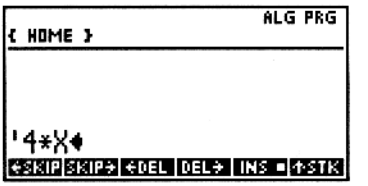
Highlight the subexpression for which this · is the top level function.

EXPR



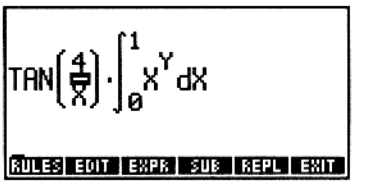
(You can press EXPR again to turn off subexpression highlighting.) For this example, press EDIT now to return the subexpression to the command line for editing. (EDIT always returns the *subexpression* to the command line, regardless of whether object or subexpression highlighting is active.)

EDIT



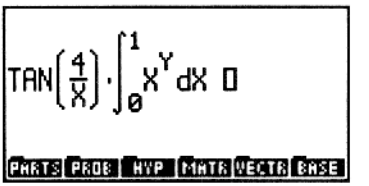
Replace the * with / and press ENTER to return the revised subexpression to the equation. (The HP 48 takes about 15 seconds to return the expression to the EquationWriter application.)

◀ ◆ ÷
ENTER



The selection cursor is now positioned on the divide bar. To leave the Selection environment, press EXIT. In about 10 seconds, the normal cursor reappears at the end of the equation and the last menu is redisplayed.

EXIT



Inserting an Object from the Stack

In addition to backspace editing and command-line editing, the EquationWriter application lets you *insert* an object from the stack at the current cursor position in an equation. You can insert:

- A name.
- A real number.
- A complex number.
- An algebraic.
- A string.

To insert an object from level 1 into an equation, press ↷ RCL. The object is inserted at the cursor position. The delimiters for names, algebraics, and strings are automatically removed.

Example: Inserting an Object from the Stack. Part 1. Enter, then copy, the expression `'X^2-Y'` by pressing:

' X y^x 2 − Y ENTER ENTER

Part 2. Using the EquationWriter application, key in the expression:

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

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

[←] EQUATION
[→] [∫] 0 [▶] 10 [▶]



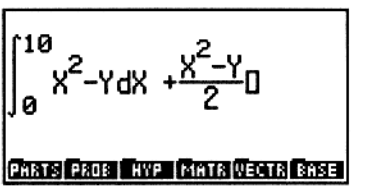
Press [→] RCL to insert the integrand into the expression. The HP 48 takes about 5 seconds to make the insertion.

[→] RCL



Complete the subexpression. Then key in the remainder of the expression, using [→] RCL to insert the second occurrence of the term.

[▶] X [▶]
[+] [▲] [→] RCL [▶]
2 [▶]



Replacing a Subexpression with an Algebraic from the Stack

To *replace* a subexpression with an algebraic currently in level 1 of the stack:

1. If the equation ends in a subexpression whose arguments are not yet keyed in, key in those arguments.
2. Press [◀] to activate the Selection menu and selection cursor.
3. Use the cursor keys to move the selection cursor to the top-level object for the subexpression you want to replace. At any time you can press EXPR to highlight the associated subexpression.
4. Press REPL. The algebraic from level 1 replaces the specified subexpression.

A Preview of the Rules Application

In chapter 22, "Algebra," you'll learn in detail about the Rules application: a set of operations that let you rearrange an algebraic expression or equation *without changing its value*. The Rules application is activated from the Selection environment in the EquationWriter application.

Suppose you want to solve for the variable x in the equation:

$$ax = bx + c$$

The Rules application lets you rearrange this equation so that x appears only once. Then you can execute the ISOL command in the ALGEBRA menu to express the equation in terms of x.

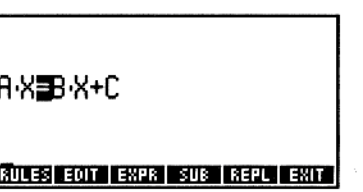
Select the EquationWriter application and key in the expression.

◀ EQUATION
A × X ◀ =
B × X + C



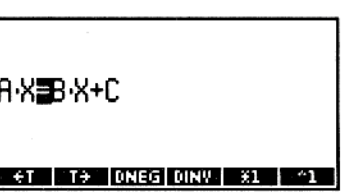
Activate the Selection menu and selection cursor. Then move the selection cursor to the = sign.

◀
then ◀ 5 times



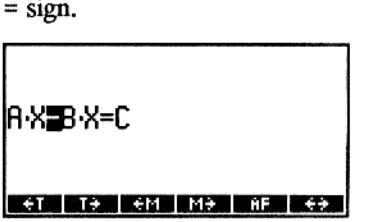
Select the RULES menu.

RULES



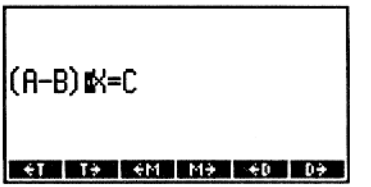
Move the term `B*X` to the left side of the = sign.

`←T`



Now merge the two terms on the left side of the = sign.

`M→`



Now that *x* occurs only once in the equation, put the equation on the stack and select the ALGEBRA menu. Then isolate *x*.

ENTER
◀ ALGEBRA
' X ISOL



17

The HP Solve Application



The HP Solve application lets you numerically solve equations containing any number of variables. You can repeat this process as often as you like, changing the value of one or more variables in the equation and solving for another. In addition, you can at any time review the current value of any variable.

Example: The HP Solve Application. The equation of motion of an accelerating body is:

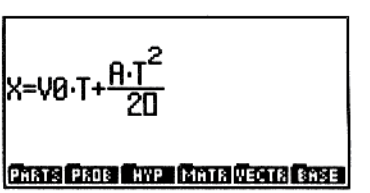
$$x = v_0 t + \frac{at^2}{2}$$

Calculate the distance (x) a body travels in 4 seconds (t) if its initial velocity (v_0) is 2 m/s and it is accelerating (a) at 3 m/s^2.

(This example assumes that variables *X*, *V0*, *T*, and *A* do not exist in the current directory.)

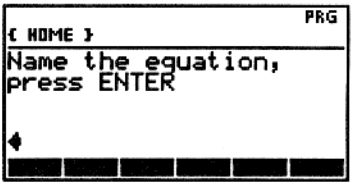
Key in the equation using the EquationWriter application.

◀ EQUATION
X ◀ = V0 × T
+ ▲ A × T y^x 2 ▶
▶ 2



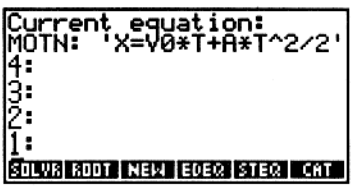
Store the equation as the *current equation*.

ENTER
↰ SOLVE NEW



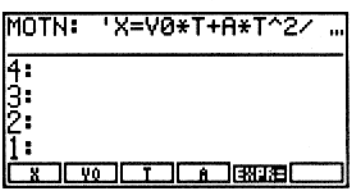
The HP 48 prompts you to enter a variable name and activates the alpha keyboard. Enter the name *MOTN*.

MOTN ENTER



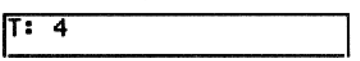
Display the SOLVR menu of variables. The menu contains a label for each variable in the equation, plus EXPR= for evaluating the equation.

SOLVR



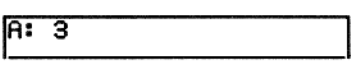
Use the SOLVR menu to store 4 in *T*.

4 T



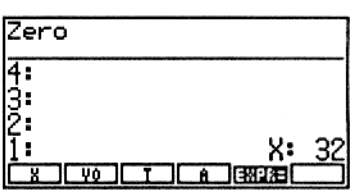
`T: 4` at the top of the display tells you 4 has been stored in *T*. Now, store 2 in *V0* and 3 in *A*.

2 V0
3 A



Use ↰ before the menu key to *solve for* a variable.

↰ X



The distance travelled is 32. Note that the numerical result is tagged with the variable name. The message ZERO in the status area indicates that a root (solution) has been found.

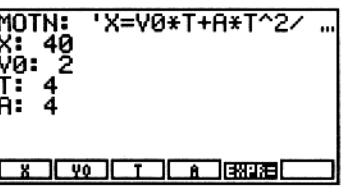
If the object actually travelled 40 meters with the same initial velocity and time, what was its acceleration?

40 [X]
[↰] [A]



Note that the solution for X from the previous calculation is in level 2; this is *not* the current value of X. To see all current values press [↰] [REVIEW].

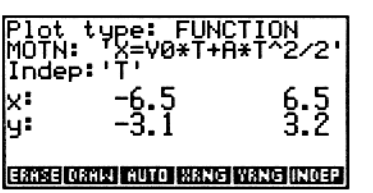
[↰] [REVIEW]



Example: Using the HP Solve Application with the Plot Application. Using the previously stored equation and values of $A = 4$ and $V0 = 2$, calculate T when $X = 30$. Since the equation is quadratic in T, there may be more than one solution. So, use the Plot application to graph the function and select the appropriate root.

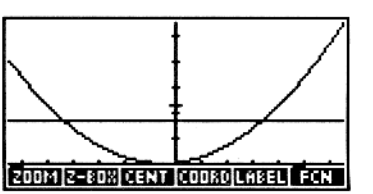
Store the new value for X. Set flag −30 to see both sides of the equation in the plot. Switch to the Plot application, select the FUNCTION plot type, and reset the plot parameters. Specify T as the independent variable.

30 [X]
30 [+/−] [▶] [MODES] [NXT] SF
[↰] [PLOT] PTYPE FUNC
PLOTR [NXT] RESET
[↰] [PREV] ' T INDEP



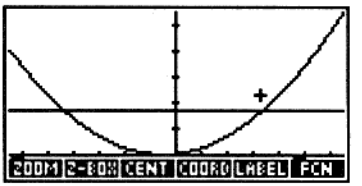
Plot the graph using autoscaling for the vertical axis.

AUTO



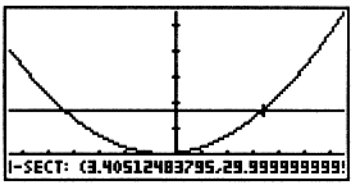
The HP 48 plots the left and right sides of the equation. Since the left side of the equation is X, and $X = 30$, the left side of the equation plots as a straight line. The right side of the equation changes with T. A solution exists where the left and right sides intersect (where the left and right sides are equal). Move the graphics cursor to the vicinity of the x-positive intersection of the two lines. (See the display below).

▶ (hold)



Find the intersection (the value of T that makes the left side of the equation equal to the right side).

FCN ISECT



T is about 3.41 (seconds).

Return to the stack display. The coordinates of the intersection are in level 1.

ATTN ATTN

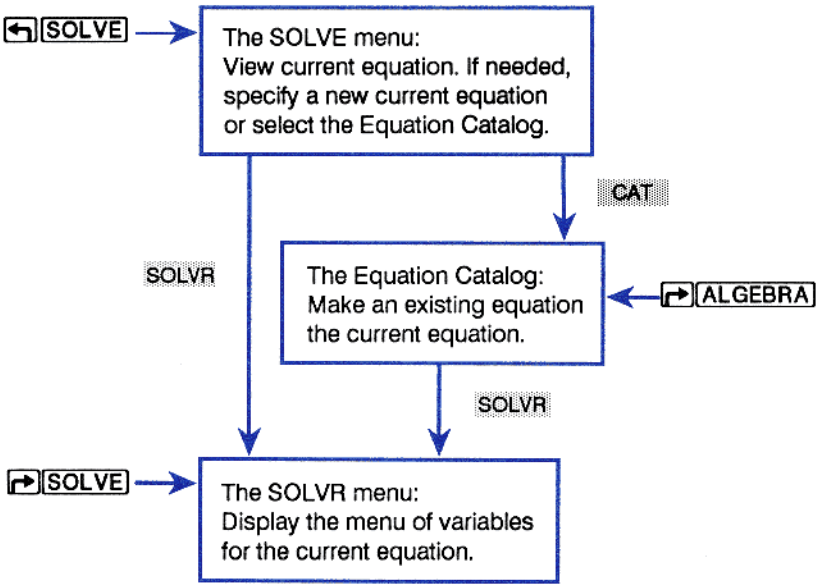


Clear flag −30 by pressing 30 +/− ↱ MODES NXT CF.

The Structure of the HP Solve Application

The HP Solve application consists of two menus, the SOLVE menu and the SOLVR menu, and a reserved variable *EQ* containing the *current equation* — the equation you want to solve. You use the SOLVE menu to view the current equation or to specify a new current equation. The SOLVE menu provides you access to the *Equation Catalog*, used to select and manage existing equations.

The SOLVR menu displays the variables for the current equation, letting you store, solve for, and review the numeric value of each equation variable.



Equations, Expressions, and Programs

The HP Solve application can solve for the numeric value of a variable in:

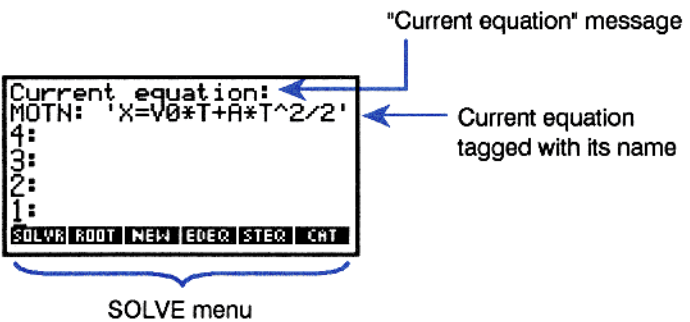
- Algebraic expressions (for example, `'A+B+C'`). A solution is a *root* of the expression — a value of the unknown variable for which the expression has a value of 0.
- Equations. An equation consists of two algebraic expressions separated by = (for example, `'A+B=C'`). A solution is a value of the unknown variable that causes both sides to have the same numerical value.
- Programs. A solution is a value of the unknown variable for which the program returns 0.
- Lists of expressions, equations, or programs.

Throughout this chapter, the term "equation" *refers to all objects used to create SOLVR menus; algebraic expressions and equations, programs, and lists of expressions, equations, or programs.*

The SOLVE Menu — Specifying the Current Equation

Press [↰] [SOLVE] to display the SOLVE menu and a two-line status message regarding the current equation. The message does one of the following:

- Indicates that there is no current equation, in which case instructions for entering a new equation are provided.
- Identifies the current equation and its name.



If you want to work with the equation identified in the status area, you can immediately select the SOLVR menu. Otherwise, you can use the operations in the SOLVE menu to edit the current equation, enter a new equation or choose an equation from the Equation Catalog.

The SOLVE Menu

Keys	Programmable Command	Description
↰ SOLVE:		
SOLVR		Selects the menu of variables for the current equation.
ROOT	ROOT	Solves an equation (in level 3) for an unknown (in level 2), using the guess(es) in level 1. ROOT is principally useful in programs.
NEW		Takes the equation from level 1, prompts for a variable name, stores the equation in that variable, and makes the equation in that variable the *current equation*.
EDEQ		Places the current equation in the command line for editing. ENTER makes the edited version the current equation and stores the edited version back in the variable. ATTN abandons the edit without altering the equation.
STEQ	STEQ	Stores the level 1 equation as the current equation.
↱ STEQ	RCEQ	Recalls the current equation to level 1.
CAT		Selects the Equation Catalog.
↰ REVIEW		Redisplays the status message.

The next section, "Entering a New Current Equation," discusses how to enter a *new* current equation. The subsequent section "The Equation Catalog—Selecting and Managing Existing Equations," describes the Equation Catalog, including how to specify the current equation from the list of equations in the catalog.

Entering a New Current Equation

You can use either NEW or STEQ to enter a new current equation. NEW helps you enter and name a new current equation by displaying an instructive message. STEQ is useful if you want to store an equation in *EQ* without naming it.

Entering a Current Equation with NEW.

1. Key in the equation using the EquationWriter application or command line. If you use the EquationWriter application, press [ENTER] to return the equation to level 1.
2. Press [↰][SOLVE] NEW. This activates the alpha keyboard and displays a prompt for a variable name.
3. Key in a name for your equation and press [ENTER]. The variable name is stored in *EQ*, and the equation itself is stored in the variable. (If you simply press [ENTER] without keying in a name, the equation itself is stored directly in *EQ*.)

If the "equation" is a list or program, NEW automatically adds .EQ to the prompt. * Any variable whose name ends with EQ is automatically stored in the Equation Catalog. (NEW includes the "." so that such names are easier to read.)

Example: Entering a Current Equation with NEW. The following equation calculates the velocity of sound in a gas:

$$v = \sqrt{\frac{\gamma RT}{M}}$$

Use NEW to name the equation and to make it the current equation.

---

* If the fraction mark is specified as "," NEW adds ,EQ.

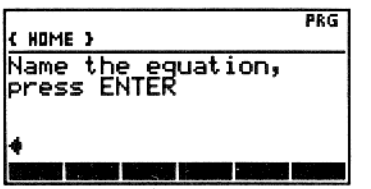
Select the EquationWriter application and key in the equation. (To key in γ, press [α] [→] [MTH].)

[◀] [EQUATION]
V [◀] [=] [√x]
γ [×] R [×] T [÷] M



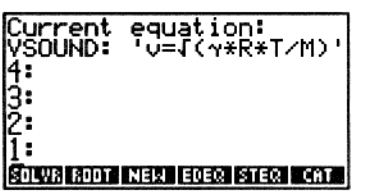
Enter the equation, select the SOLVE menu and execute NEW.

[ENTER]
[◀] [SOLVE] NEW



Name the equation VSOUND and make it the current equation. You don't have to press [α] since NEW locks Alpha-entry mode until you press [ENTER].

VSOUND [ENTER]



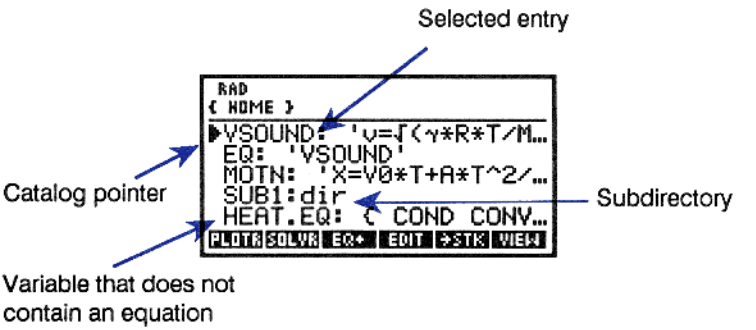
Entering a Current Equation with STEQ. The STEQ command stores the equation (or the equation name) from stack level 1 directly into *EQ*. Note that an unnamed equation in *EQ* is lost the next time a new equation is stored in *EQ*.

The Equation Catalog—Selecting and Managing Existing Equations

The Equation Catalog is a special environment tailored to manage existing equations. In the Equation Catalog, the stack is replaced with a listing of all named equations in the current directory, and the keyboard is redefined to execute operations that are specific to the Equation Catalog. These operations let you select the current equation or combine, edit, reorder, and purge existing equations. The Equation Catalog contains, in addition to named algebraics, all variables ending in *EQ* and all directories for which the current directory is the parent directory.

The HP Solve application shares the Equation Catalog with the Plot application, described in chapter 18. References to the Plot application are made throughout this section.

To select the Equation Catalog, press ◀ SOLVE CAT, ◀ PLOT CAT, or ▶ ALGEBRA.



Move the pointer with ▲ and ▼ to select the desired equation or subdirectory. The operations in the Equation Catalog work on the selected entry.

Operations in the Equation Catalog

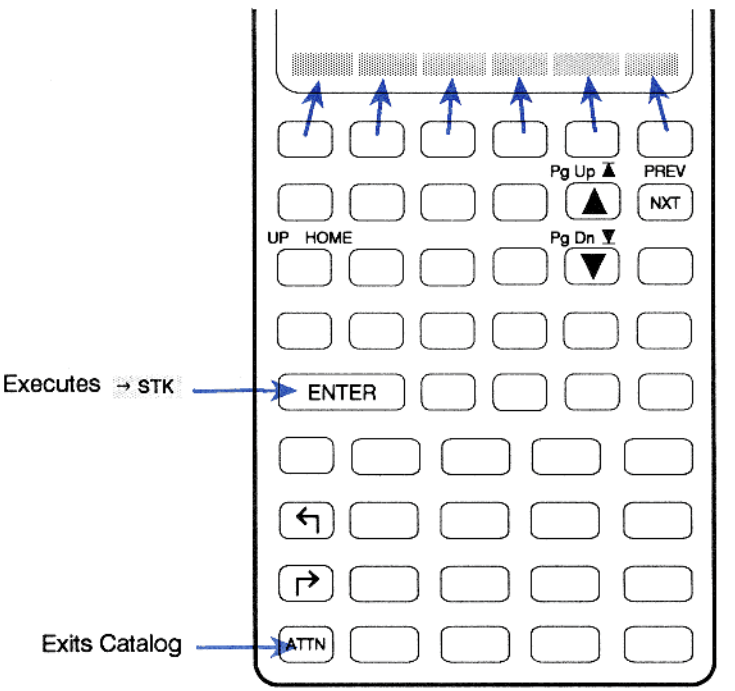
PLOTR	Makes the selected entry the current equation and displays the PLOTR menu.
SOLVR	Makes the selected entry the current equation and displays its menu of variables.
EQ+	Creates or adds to a list of equations (see page 272). (◀ EQ+ removes the last entry from the list.)
EDIT	Places the selected entry in the command line for editing. Press ENTER when editing is complete. Press ATTN to abort editing without implementing any changes.

Operations in the Equation Catalog (continued)

→STK	Copies the selected entry to the stack.
VIEW	Clears the display and shows *only* the selected entry, without its name, until the key is released. If the selected entry is a directory, VIEW switches to that directory.
ORDER	Makes the selected entry the first entry in the catalog. If you create a list of *n* equations with EQ+, ORDER makes those equations the first *n* entries in the catalog.
PURG	Purges the selected entry from the catalog (and from the current directory).
FAST	Enabling FAST▪ shows the names in the catalog (and in status messages) without their contents. (Enabling FAST▪ sets flag −59. The converse is also true.) FAST▪ is useful if the catalog contains many long equations, since such equations are slow to display.
▲	Moves the catalog pointer up one level. When prefixed with ↰, moves the catalog pointer up one page (↰ PgUp in the following keyboard illustration); when prefixed with ↱, moves the catalog pointer to the top of the catalog (↱ ▲ in the following keyboard illustration).
▼	Moves the catalog pointer down one level. When prefixed with ↰, moves the catalog pointer down one page (↰ PgDn in the following keyboard illustration); when prefixed with ↱, moves the catalog pointer to the bottom of the catalog (↱ ▼ in the following keyboard illustration).
ATTN	Exits the Equation Catalog.

Operations in the Equation Catalog (continued)

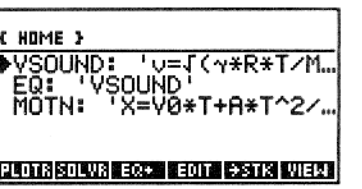
ENTER	Executes →STK (copies the selected *equation* to the stack). If the selected entry is a *directory*, switches to the Equation Catalog in that directory.
↰ UP	Switches to the Equation Catalog of the parent directory.
↱ HOME	Switches to the Equation Catalog of the *HOME* directory.



Example: Using the Equation Catalog to Specify the Current Equation. Use the Equation Catalog to select MOTN as the current equation and display its menu of variables.

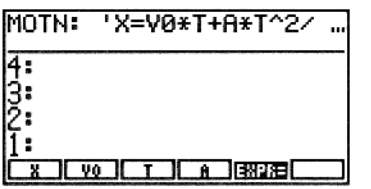
Select the HP Solve application and then the Equation Catalog.

◀ SOLVE CAT



Move the pointer to *MOTN* if necessary. Then make it the current equation and display its menu of variables.

(▼ as necessary.) SOLVR



Note that the selected entry does not become the current equation until you press SOLVR *or* PLOTR.

Exiting the Catalog. The previous example demonstrates how, in the Equation Catalog, pressing SOLVR exits the catalog, makes the selected entry the current equation, and displays the SOLVR menu. Pressing PLOTR in the Equation Catalog exits the catalog, makes the selected entry the current equation, and displays the PLOTR menu (described in chapter 18).

To exit the Equation Catalog and return to the SOLVE (or PLOT) menu without changing the contents of *EQ*, press ATTN.

The SOLVR Menu — Solving the Current Equation

You can display the SOLVR menu (the menu of variables for the current equation) by doing either of the following:

- Pressing SOLVR in the SOLVE menu or Equation Catalog.
- Pressing [→] [SOLVE] from any menu.

The SOLVR menu contains:

- A menu key for each variable in the current equation. (For more information, see "How the Menu of Variables is Created" on page 278.) The labels are white with black letters; this distinguishes the HP Solve application variables from variables in the VAR menu.
- The EXPR= key, discussed later in this section.

The menu keys for the equation variables work differently from standard VAR or CST menu keys:

- The unshifted key *stores* the value from the command line or stack level 1 into the corresponding variable.
- The left-shifted key *solves* for the corresponding variable, returning a tagged result to level 1.
- The right-shifted key *recalls* the value of the corresponding variable to level 1.

The SOLVR menu remains unchanged until a new current equation is specified.

Example: Basic Use of the HP Solve Application. The equation for a simple resistive circuit is:

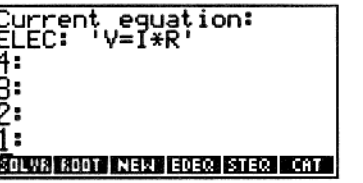
$$V = IR$$

where V is the circuit voltage, I is the circuit current, and R is the circuit resistance. Use the HP Solve application to find the value of I when V is 10 volts and R is 20 ohms.

(This example assumes that variables V, I, and R do not exist in the current directory.)

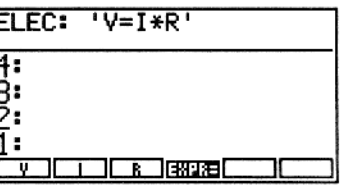
Select the HP Solve application, then key in the equation. Use NEW to name the equation *ELEC* and make it the current equation.

← SOLVE
↑ V ← = I × R
NEW ELEC ENTER



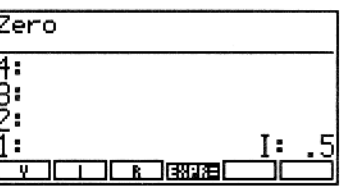
Display the menu of variables for the current equation.

SOLVR



Supply the known values and solve for the unknown.

10 V
20 R
← I

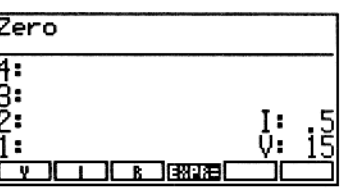


The message ZERO in the status area indicates that a root has been found. (See "Interpreting Results" on page 279 for a description of the messages returned by the HP Solve application.)

If R is 30 ohms for the same value of I, what is V?

Store the new value of R, then solve for V.

30 R
← V



Now review the values of all the variables.

◀ REVIEW



Using EXPR=

EXPR= returns the one or two values, depending on the form of the current equation:

- For an algebraic expression or a program, EXPR= returns the tagged result EXPR: *value* to level 1.
- For an equation, EXPR= returns two tagged results: LEFT: *value on left side of = sign* to level 2, and RIGHT: *value on right side of = sign* to level 1.

Verifying Solutions with EXPR=. An important use of EXPR= is to help you determine if the HP Solve application has found a true solution (called a *root*). For an expression, the closer the result returned by EXPR= is to zero, the more likely it is that the HP Solve application has found a root. For an equation, the closer the two results returned by EXPR= are to each other, the more likely it is that the HP Solve application has found a root. For more information, see "Interpreting Results" on page 279.

Example: Using EXPR= to Verify a Result. Use EXPR= to evaluate the two sides of the current equation from the previous example.

Assuming that *ELEC* is still the current equation, select the SOLVR menu directly and evaluate the equation.

▶ SOLVE
EXPR=



The left and right sides of the equation are both exactly 15, indicating the HP Solve application found a root.

Choosing Guesses

You can supply one or more guesses for the unknown variable before solving for it. To store a guess, store a value into the unknown (press the unshifted menu key). To store two or three guesses to bracket a desired solution, put them in a list and store the list. For example, { 0 10 } X stores 0 and 10 as guesses for unknown *X*.

Good guesses help in two ways:

- If there is more than one solution, guesses control which solution is found.
- Good guesses reduce the time required to find a solution.

See "How the Root-Finder Uses Initial Guesses" on page 277 for more information.

Solving Equations with the Plot Application

The Plot application lets you find the solution or solutions for an equation by working directly with a picture of the equation. This powerful capability is of great value if you don't know what the equation looks like over a range of values. Specifically, equations can have:

- Multiple solutions. The HP Solve application finds only one solution unless you provide additional guesses to direct it to another region of the equation.
- Local minima or maxima. The HP Solve application returns a local minimum or maximum as a solution if the guesses you provide direct it to that region of an equation, even if the equation has "true" solutions (roots) in other regions.

The Plot application lets you graph the equation and then make an intelligent guess by moving the special graphics cursor directly to the region of the equation that contains the desired solution.

See "Analyzing Functions" on page 307 in chapter 18 for more information.

Choosing the More Useful Application. The HP Solve application provides the following advantages:

- You can easily store values for the known variables and solve for the unknown, and you can easily change which variable is the unknown. (In the Plot application, you must use the INDEP command to specify the unknown variable, and you must redraw the plot for each new choice of the independent variable.)
- You can review the values of the variables in the equation.

The Plot application provides the following advantages:

- You can see if an equation has multiple solutions or local extrema.
- You can direct the application to a specific solution simply by moving a cursor, rather than by entering numerical guesses.

Using the HP Solve Application with Unit Objects

The current equation and any of its variables may contain unit objects. The following guidelines apply:

- Before solving, all variables must contain a consistent set of units, *including the unknown variable*. For example, if the equation is `'Y=X/T'` and you've stored `2_m` in *X* and `3_s` in *T*, you must enter a guess for *Y* with the dimensions *length/time*. The solution is computed in the units specified in the guess. Thus, you could enter a guess of `1_ft/yr`, and the answer would be computed in units of `ft/yr`.
- When a variable contains a unit object, you can change the number portion without affecting the unit. For example, if *X* contains `2_m`, the keystrokes 6 [X] store `6_m` in *X*. However, for a list of two or three guesses, one of the guesses must have the appropriate dimensions. (If more than one guess has units, the units of the last guess and only the number parts of the other guesses are used.)
- If the equation you are solving uses or calculates temperature *difference* (as opposed to actual temperature), use K or °R (do not use °C or °F). For temperature *conversion*, use the UNITS catalog menu.

Note

Since the menu of variables allows you to change the number portion of a unit object without affecting the unit, you must purge variables containing unit objects before using them in equations requiring numbers only.

Example: Using the HP Solve Application with Units. Use the equation:

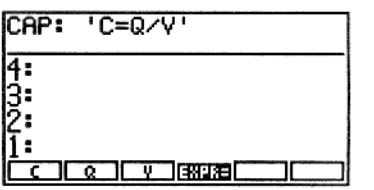
$$C = \frac{Q}{V}$$

to calculate the capacitance C when $Q = 8.9\times10^{-6}$ coulombs and $V = 57$ volts.

(This example assumes that variables C, Q, and V do not exist in the current directory.)

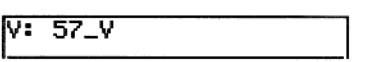
Enter and name the equation, then select the SOLVR menu.

[↰] [SOLVE]
[↑] C [↰] [=] Q [÷] V NEW
CAP [ENTER]
SOLVR



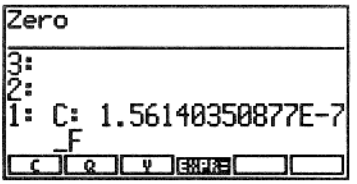
Enter the known values, then store them.

[↰] [UNITS] [NXT] ELEC
57 V
8.9 [EEX] 6 [+/-] C
[↱] [SOLVE] Q
V



Store a guess and solve for the unknown.

→ LAST MENU 1 F
→ LAST MENU C
← C

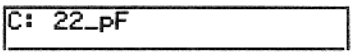


The answer is returned in farads.

Now, solve for V in millivolts for C = 22 picofarads and Q = 1.7×10^{-10} coulombs.

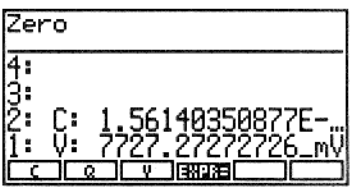
Store the new value of Q. You do not need to append the unit. Store C with its new unit.

1.7 EEX 10 +/- Q
22 → _ pF ENTER C



Store a guess for V in millivolts and solve for the unknown.

1 → _ mV ENTER V
← V



Customizing the SOLVR Menu

The capabilities of the *CST* menu (chapter 15) can be imparted to the SOLVR menu, letting you:

- Specify which equation variables appear in the SOLVR menu, and in what order.
- Build menu keys in the SOLVR menu that execute various objects. In this way, you can solve equations *and* execute associated calculator operations without leaving the SOLVR menu.

To create a customized SOLVR menu:

1. Build a *solver-list* in level 1 that contains:
   - The equation.
   - A nested list that defines the label and functionality of each key in the menu.

   The syntax of the solver-list is:

   « '*equation*' « *key definitions* » »

   The syntax for the sublist of individual key definitions is described in detail in chapter 15, "Customizing the Calculator."
2. Execute NEW to name the solver-list and make it the current equation.

Naming the Solver-List. Remember that variables containing algebraic objects are automatically included in the Equation Catalog. To include a variable that does *not* contain an algebraic object, end the variable name in EQ — any variable whose name ends in EQ is automatically included in the Equation Catalog.

To help you remember this naming convention, NEW automatically adds .EQ to the name when the level 1 object is a list or program.

Example: Specifying the Variables In the SOLVR Menu. The equation:

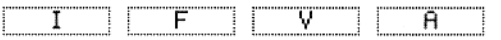
$$I = 2\pi^2 f^2 \rho v a^2$$

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.

The solver-list:

```
{ 'I=2*π^2*f^2*ρ*v*a^2' { I f v a } }
```

when stored in *EQ*, creates the SOLVR menu:

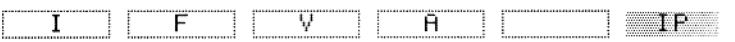


for the equation and suppresses ρ from the menu. To save this solver-list in the equation catalog, store it in a variable ending in EQ, for example, *I.EQ*.

Example: Including an Executable Object in the SOLVR Menu. Suppose you want the command IP available in the SOLVR menu so that you can store integer values in the variables in the SOLVR menu. The following solver-list list amends the solver-list in the previous example to include two additional keys; a blank key and a key that executes IP (integer part):

```
{ 'I=2*π^2*f^2*ρ*v*a^2' { I f v a { } IP } }
```

This list, when stored in *EQ*, creates a menu of variables and functions:



Including Programs in the Solver-List. A solver-list cannot use a program *name* as an executable object in the sublist of key definitions, since names in the sublist are treated as equation variables. To include a program in the SOLVR menu, do one of the following:

- Use the (unnamed) program object itself as the sublist entry.
- Use a list with the following syntax as the sublist entry:

  ```
  { "label" « name » }
  ```

  where *label* is the menu key label and *name* is the name of the program.

Linking Two or More Equations

You may often work with two or more related equations, for example, equations with common variables. You can link the SOLVR menus for a set of equations by creating a list that contains the names of those equations, and then making that list the current "equation".

Using EQ+ to Link Two or More Equations

The EQ+ operation in the Equation Catalog menu lets you link two or more equations together as follows:

1. In the Equation Catalog, position the pointer at each equation you want included in the temporary menu and press EQ+ . A list containing each selected equation is displayed and updated in the status area. (◀ EQ+ removes the last entry from the list.)
2. Press SOLVR.

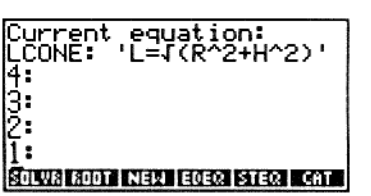
The list is stored in *EQ* and the SOLVR menu for the first equation in the list is displayed, with the additional key NXEQ. Pressing NXEQ rotates the names in the list, moving the second name to the beginning of the list so that variables for that equation appear.

Example: Linking Two Equations with EQ+. Part 1. Create the two equations 'L=√(R^2+H^2)' and 'V=π*R^2*H/3'. Use NEW to name them *LCONE* and *VCONE* respectively, and to store them in the Equation Catalog.

(This example assumes that variables *L*, *R*, *H*, and *V* do not exist in the current directory. If you used *V* in the Units example earlier in the chapter and you're now in the same directory, you'll need to purge *V*.)

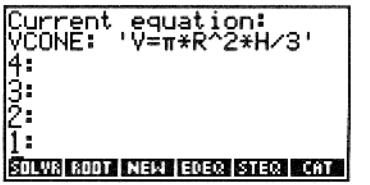
Key in and store the first equation.

◀ EQUATION
L ◀ = √x ◀ ( )
R y^x 2 ▶ + H y^x 2
ENTER
◀ SOLVE NEW LCONE ENTER



Key in and store the second equation.

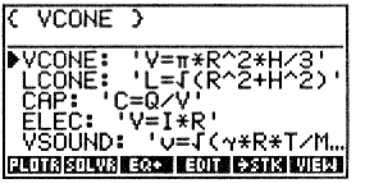
◁ EQUATION
V ◁ = ◁ π ×
R y^x 2 ▷ × H ÷ 3
ENTER
NEW VCONE ENTER



Part 2. Use EQ+ to put the two equations in a list. Then execute SOLVR to store the two equations in *EQ* and display the linked menu of variables.

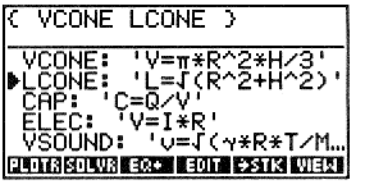
Use the ▲ or ▼ keys as necessary to position the cursor at equation *VCONE*. Execute EQ+ to put the equation in a list.

CAT
(▲ or ▼ if necessary.)
EQ+



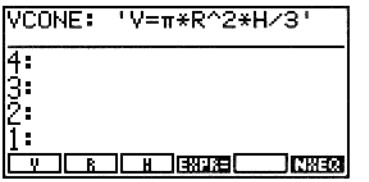
Move the cursor to *LCONE* and execute EQ+.

▼ EQ+



Select the SOLVR menu. Press NXEQ to switch between the two equation menus.

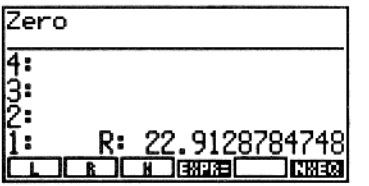
SOLVR NXEQ NXEQ



Part 3. Find the radius of a right circular cone whose height is 10 meters and whose lateral surface area is 25 square meters. Then find its volume.

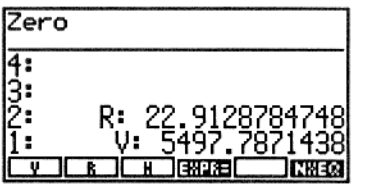
Display the menu of variables for *LCONE*. Then supply values for the known variables and solve for the unknown.

NXEQ 10 H
25 L
↰ R



Switch to the menu of variables for *VCONE*. You can solve directly for the volume since the radius and height are already stored in their variables.

NXEQ ↰ V



Saving a List of Two or More Equations

The list you store in *EQ* by executing EQ+ and SOLVR is unnamed and therefore is not saved if you later change *EQ*. To name the list currently in *EQ*:

1. Execute RCEQ (↰ SOLVE ↱ STEQ) to recall the list to level 1.
2. Execute NEW to give the list a name ending in EQ. Any variable whose name ends in EQ is included in the Equation Catalog.

If you want to name the list before storing it in *EQ*:

1. Build the list in the Equation Catalog with EQ+.
2. Press →STK to copy the list onto the stack.
3. Leave the Equation Catalog with ATTN and execute NEW to give the list a name ending in EQ.

Using the HP Solve Application to Find Solutions of Programs

The HP Solve application accepts a program as the current equation. The program must take nothing from the stack and return only one result. Using a program as the current equation is useful when the relationship between variables can't be written algebraically. For example, the MTH PROB menu contains the UTPC (*upper tail chi-square distribution*) command for calculating the probability that a chi-square random variable with n degrees of freedom is greater than χ. The relationship is:

$$UTP = \text{UTPC}(n, \chi^2)$$

where *UTP* is the unknown variable.

The command UTPC cannot be included in algebraics. However, the relationship can be rewritten as an expression equal to 0:

$$UTP - \text{UTPC}(n, \chi^2) = 0$$

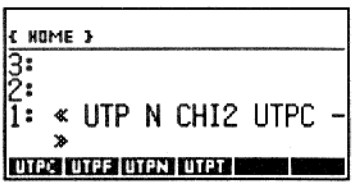
A program for computing the value of the expression is:

```
« UTP N CHI2 UTPC - »
```

Use this program to calculate the upper tail probability (*UTP*) for χ^2 (*CHI2*) = 6.2 and degrees of freedom (*N*) = 5.

Key in and enter the program.

[←] [« »] UTP N CHI2
[MTH] PROB [NXT] UTPC [−]
[ENTER]



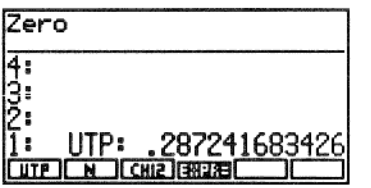
Use NEW to name the program and make it the current equation. Note that NEW automatically prompts you with .EQ since the object is a program.

◀ SOLVE NEW
CHI ENTER



Display the SOLVR menu and store the known values. Then calculate the upper tail probability.

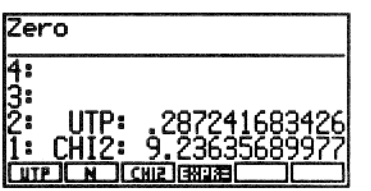
SOLVR
5 N
6.2 CHI2
◀ UTP



Now, calculate χ^2 to a significance of 0.1 with 5 degrees of freedom:

The significance is stored into variable *UTP*.

.1 UTP
◀ CHI2



How the HP Solve Application Works

Pressing a left-shifted menu key in the SOLVR menu activates the numerical *root-finder*, which seeks a solution iteratively. Starting with the guess(es) you have stored in the variable, or the guesses that the calculator itself provides, it generates pairs of intermediate guesses until a solution is found. The HP 48 displays `Solving for` *var name* while the root-finder is executing.

In searching for a solution, the root-finder seeks a value of the unknown for which the value of the expression equals 0. (Equations are treated internally as expressions of the form '*left side–right side*'.) First, the root-finder searches for two points where the expression's value has opposite signs. When it finds a sign reversal, the root-finder tries to narrow the search region until it finds a point where the expression's value is 0.

How the Root-Finder Uses Initial Guesses

You can enter one, two, or three values as guesses. Two or three values are entered as a list.

- **One value.** The number is converted to two values by copying the number and adding a small perturbation to one copy.
- **Two values.** The numbers identify a region where the search will begin. If the two guesses yield expression values with opposite signs, the root-finder usually finds a root between the two numbers rapidly. If the two guesses yield expression values with the same sign, the search generally takes longer.
- **Three values.** The first number should be your best guess for the root. The second and third numbers are used as two values, above.

Halting the Root-Finder

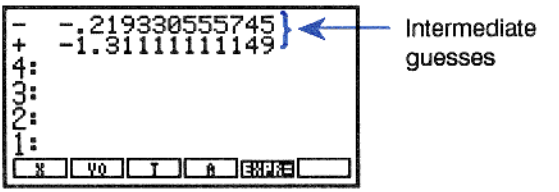
To halt the root-finder, press [ATTN]. The HP Solve application returns a list containing three values: the best value found so far plus two values that identify the region that was being searched. To restart the root-finder from where it left off:

1. Store the list in the unknown variable by pressing the corresponding menu key.
2. Start the calculation by pressing [↶] followed by the menu key.

If you want to restart the search in a different region, store a different list.

Displaying Intermediate Guesses

While the HP 48 is displaying the `Solving for` message, pressing any key except ATTN displays pairs of intermediate guesses and the sign of the values of the current equation for each guess. If the current equation is undefined at the guess, `?` is shown.



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.

How the Menu of Variables Is Created

The menu of variables contains a label for each variable in the current equation. If the variable does not already exist, it is created and added to the current directory when you store a value in it.

If a variable in the current equation contains an algebraic object (or a name or program), the variable itself is not included in the menu of variables. Instead, the variables in the algebraic object are used.

For example, if the current equation is `'A=B+C'`, and *B* contains the expression `'D+TAN(E)'`, the menu of variables is:



Note that for equations that contain a *where* clause (see page 416 in chapter 22, "Algebra") or an integral, summation, or derivative, the *placeholder* variable appears in the SOLVR menu. For example, the SOLVR menu for the equation:

$$A + B - \int_0^1 2X\,dX = 0$$

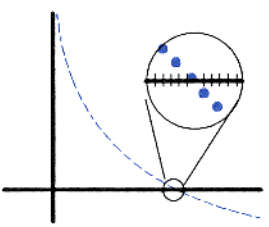
will contain a key labeled [X]. However, you *cannot* solve for this placeholder variable.

Interpreting Results

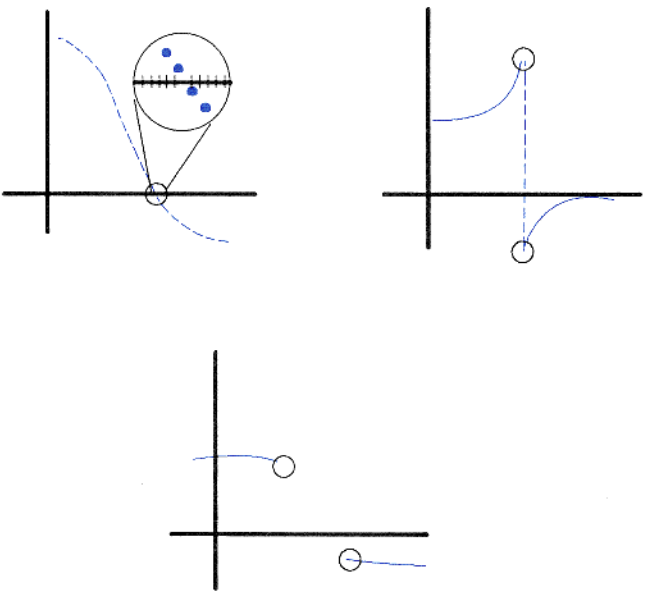
When a Solution is Found

If a root is found, the HP Solve application returns a message describing the root:

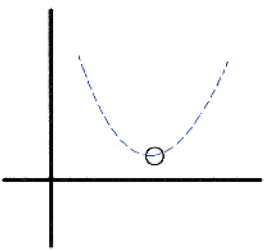
- `Zero`. The HP Solve application found a point where the expression's value is 0 within the calculator's 12-digit precision. (For equations, the left and right sides are equal.)



- `Sign Reversal`. The HP Solve application found two points where the expression's values have opposite signs, but it cannot find a point in between where the expression's value is 0. This may be because:
  - The two points are neighbors (they differ by 1 in the 12th digit).
  - The expression is not real-valued between the two points. the HP Solve application returns the point where the expression's value is closer to 0. If the expression is a continuous real function, this point is the HP Solve application's best approximation of an actual root.



- Extremum. One of the following occurred:
  - The HP Solve application found a point where the expression's value approximates a local minimum (for positive values) or maximum (for negative values.
  - The HP Solve application stopped searching at ±9.99999999999E499, the largest or smallest numbers in the calculator's range of numbers.



To obtain more information, you can:

- Evaluate the equation using EXPR=. For an expression or program, the closer the result is to 0, or for an equation, the closer the two results are to each other, the more likely it is that the HP 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 E). For example, flag −25 indicates whether overflow occurred.

When No Solution is Found

If the HP 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. Therefore, the equation generated an error when it was evaluated at that point.
- `Constant?`. The expression returns the same value at every point sampled.

18

Basic Plotting and Function Analysis



The Plot application lets you:

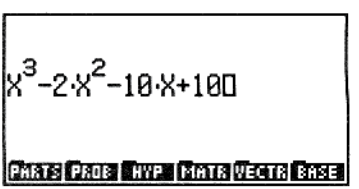
- Draw graphs of mathematical expressions and equations.
- Use plots to calculate roots, intersections, slopes, extrema, and areas.
- Plot two or more expressions or equations with one command.
- Draw scatter plots, bar charts, and histograms for statistical data.
- Add elements such as axes labels, dots, lines, circles, and boxes to plots.

A Plotting Example. Plot the expression:

$$x^3 - 2x^2 - 10x + 10$$

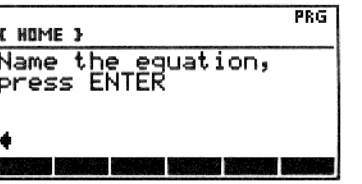
Key in the expression using the EquationWriter application.

[↰] [EQUATION] X [y^x] 3 [▶] [−]
2X [y^x] 2 [▶] [−]
10X [+] 10



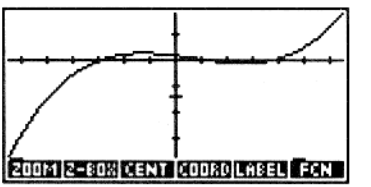
Store the equation as the *current* PLOT equation.

ENTER
◀ PLOT NEW



The HP 48 prompts you to enter a variable name and activates the alpha keyboard. Enter the name *P1*. Reset the plot parameters and draw the graph using autoscaling for the *y*-axis.

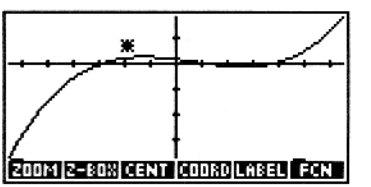
P1 ENTER
PLOTR NXT RESET
◀ PREV AUTO



The HP 48 draws the plot and then displays a menu of operations that let you interact with the plot.

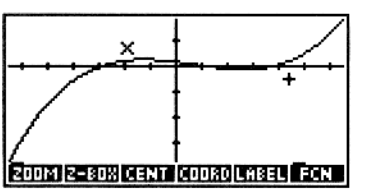
Use the cursor keys to position the *graphics cursor* (+) to the upper left position shown below and press ⊠.

use ▲ and ◀ to move cursor
to upper-left, then
⊠



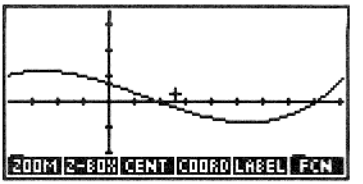
Now move the cursor to the lower right position.

use ▶ and ▼ to move cursor
to lower-right



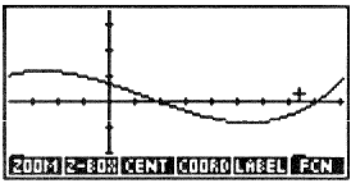
Notice that an ×, called the *mark*, is left at the position where you pressed [×]. The mark and cursor define the corners of a box. Now, zoom in on the box.

Z-BOX



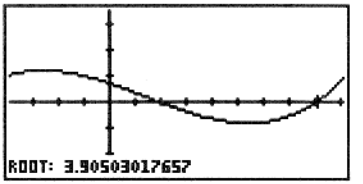
Move the cursor near the right-most root (see the cursor position below).

([▶] repeatedly)



Calculate the root.

FCN ROOT



Note that the cursor moves to the root and that the x-axis coordinate of the root is displayed in the lower left corner of the display.

Return to the stack display and PLOT menu. The root has been returned to the stack as a tagged object.

[ATTN] [ATTN]



The Structure of the Plot Application

To plot an equation by hand you would use the following general procedure:

- Write down the equation you want to plot.
- Select the independent variable, for example x, in the equation. Then determine the range over which to sample the independent variable, and the number of (evenly spaced) sample points to evaluate in the range. From this information, draw an appropriately scaled x-axis. Then draw an appropriately scaled y-axis based on your estimates of the equation's value over the sample range.
- For every value of x, calculate the value of the equation, $f(x)$, and plot the corresponding point $(x, f(x))$.
- Draw a smooth curve through the points.

The HP 48 Plot application contains special data elements that parallel the elements of the procedure described above:

- The reserved variable *EQ* contains the equation you want to plot. The equation in *EQ* is called the *current equation*. Note that *EQ* is also used by the HP Solve application to build the SOLVR menu.
- The reserved variable *PPAR* contains specifications for the independent variable, the display and plotting ranges, the number of sample points in the plotting range, and the axes.
- A part of HP 48 memory called *PICT* is analogous to the piece of paper on which the plot is drawn.

These data elements are tied to two menus and a special environment:

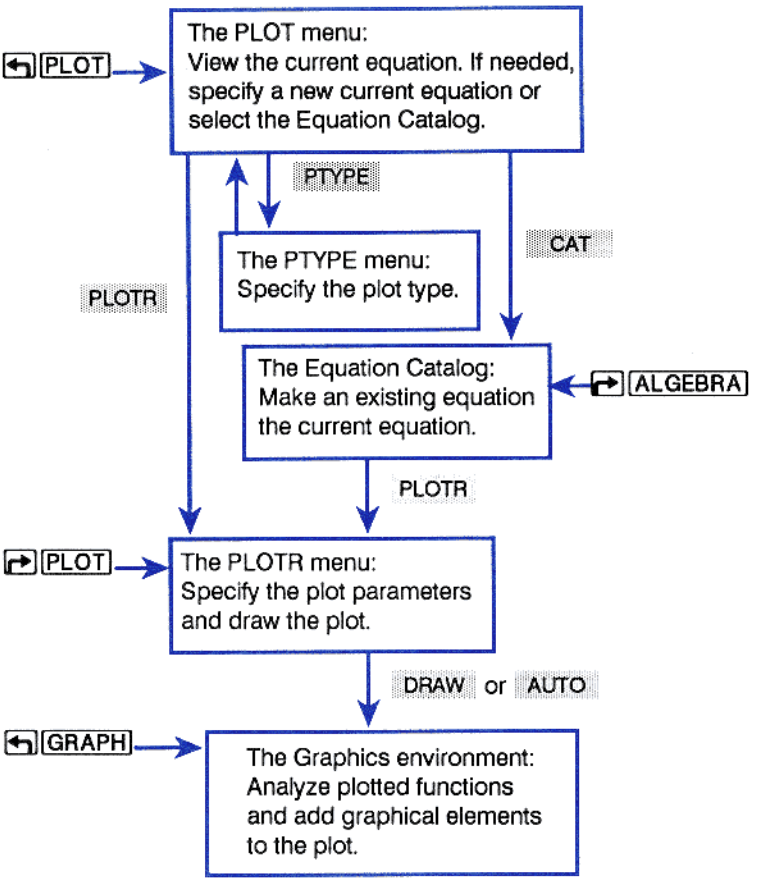
- The PLOT menu is used for the selection or modification of the current equation. The PLOT menu is also used to specify the *plot type*, which determines how the HP 48 interprets the equation. For example, the equation may represent a conic section — in this case, the appropriate plot type is CONIC.
- The PLOTR menu is used to specify the contents of *PPAR* and to draw the plot.

- The *Graphics environment* is used to view the graph, zoom in or out from a region of the plot, analyze the mathematical behavior of the plot, and add graphical elements to it. In this environment, the display serves as a window onto *PICT*, and the keyboard is redefined to execute graphics operations. When the HP 48 finishes a plot, it automatically puts you in the Graphics environment. *PICT* persists after you leave the Graphics environment—you can reenter the Graphics environment at any time to view *PICT*.

  In the Graphics environment you do not have access to the stack. However, function analysis operations in the Graphics environment return their results to the stack. In addition, all or parts of *PICT* can be copied to the stack as an object called a *graphics object*. Commands in the PRG DSPL menu let you work with graphics objects on the stack and let you move a graphics object back into *PICT*.

In general, you use these steps to plot an equation:

1. Use the PLOT menu to store the equation in *EQ* and, if necessary, to specify the plot type.
2. Use the PLOTR menu to set the appropriate plot parameters.
3. Draw the graph.
4. Once the graph is drawn, you can use the operations in the Graphics environment to obtain data from it, or to add graphical elements to it.



This chapter covers basic plotting and analysis of mathematical functions. You'll learn how to use the PLOT menu to specify the current equation, how to use the PLOTR menu to specify plot parameters related to basic function plots, and how to use the Graphics environment to analyze function plots. All examples in this chapter will use the FUNCTION plot type.

In chapter 19, you'll learn about:

- Additional plot parameters in the PLOTR menu.
- How plot parameters are stored.
- Conic, polar, parametric, truth, and statistical plots.
- Adding graphical elements to *PICT*.
- Plotting statistical data.
- Working with graphics objects on the stack.

Equations, Expressions, and Programs

The HP 48 can plot:

- Expressions (for example, `'X^2-2*X'`).
- Equations. An equation consists of two algebraic expressions separated by = (for example, `'Y=X^2-2*X'`).
- Programs (see page 334 in chapter 19).
- Lists of expressions, equations, or programs (see page 300).

Throughout this chapter, unless otherwise stated, the term "equation" *refers to all objects used to create plots: algebraic expressions and equations, programs, and lists of expressions, equations, or programs.*

The PLOT Menu — Specifying the Current Equation and Plot Type

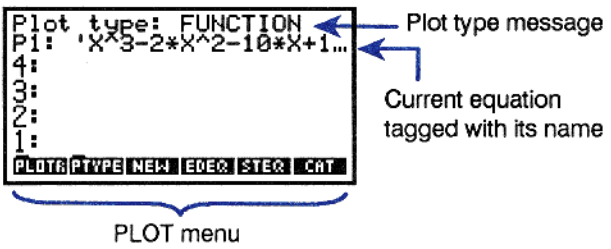
Press ↰ PLOT to select the PLOT menu.

The PLOT Menu

Keys	Programmable Command	Description
↰ PLOT:		
PLOTR		Selects the PLOTR menu for specifying the plot parameters in *PPAR*, and for drawing the plot.
PTYPE		Displays the PTYPE menu for specifying the plot type.
NEW		Takes an equation from level 1, prompts for a variable name, stores the equation in that variable, and makes the equation in that variable the *current equation*.
EDEQ		Places the current equation in the command line for editing. ENTER makes the edited version the current equation and stores the edited version back in the variable. ATTN abandons the edit without altering the equation.
STEQ	STEQ	Stores the level 1 equation as the current equation.
↱ STEQ	RCEQ	Recalls the current equation to level 1.
CAT		Selects the Equation Catalog.
↰ REVIEW		Redisplays the status message.

[↰][PLOT] also displays a two-line status message regarding the current equation. The message either:

- Indicates that there is no current equation, in which case instructions for entering a new equation are provided.
- Identifies the current equation (or current statistical data) and its name in the second line, and the current plot type in the first line.



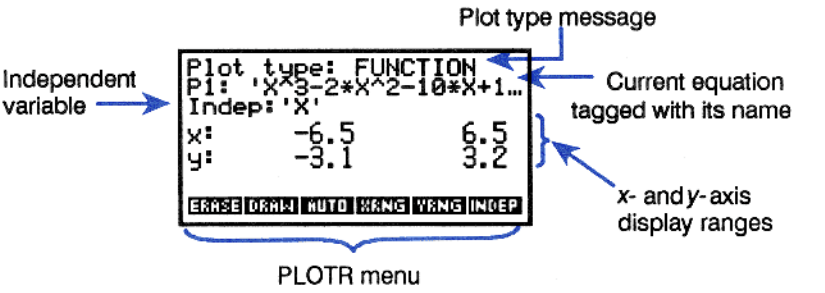
If you want to plot the equation identified in the status area (using the current plot type), you can immediately select the PLOTR menu. Otherwise, you must either enter a new equation or choose an equation from the Equation Catalog. "Entering a New Current Equation" on page 257 in chapter 17 discusses how to enter a *new* current equation. "The Equation Catalog—Selecting and Managing Existing Equations" on page 258 in chapter 17 describes the Equation Catalog, including how to specify the current equation from the list of equations in the catalog.

The PLOTR Menu—Setting Plot Parameters and Drawing the Plot

Pressing [↰][PLOT] PLOTR or [↱][PLOT] displays the PLOTR menu and a status message describing:

- The *plot type*. (Plot types are discussed in chapter 19, on page 327.)
- The current data—either the *current equation* or the *current statistical data*—if there is any.
- The *independent* variable (*X* by default) and, if specified, the *plotting* range. (Specifying a plotting range is discussed in chapter 19, on page 319.)

- The *display* ranges in the horizontal and vertical directions. In this message, x always indicates the horizontal direction, and y always indicates the vertical direction.



The PLOTR menu contains the commands for setting the plot parameters and for drawing the plot.

Basic Plotting Operations in the PLOTR Menu

Keys	Programmable Command	Description
◀ PLOT PLOTR:		
ERASE	ERASE	Erases *PICT*, leaving a blank *PICT* of the same size.
DRAW	DRAW	Draws the plot using the *x*- and *y*-axis ranges. DRAW does not erase *PICT*—the plot is drawn over any previous contents of *PICT*. When executed from a program, DRAW does not include axes in the graph. (◀ DRAW executes STEQ. ▶ DRAW executes RCEQ.)

Basic Plotting Operations in the PLOTR Menu (continued)

Keys	Programmable Command	Description
AUTO	AUTO	Draws the graph using the *x*-axis range, and autoscales the *y*-axis. Any previous plot in *PICT* is erased. When executed from a program, AUTO only autoscales the *y*-axis—it does not draw a graph.
XRNG	XRNG	Sets the display range of the horizontal axis using two real-number arguments—x_{min} and x_{max}. ([→] XRNG recalls the current *x*-axis display range.)
YRNG	YRNG	Sets the display range of the vertical axis using two real-number arguments—y_{min} and y_{max}. ([→] YRNG recalls the current *y*-axis display range.
INDEP	INDEP	Sets the name in level 1 as the independent variable. INDEP can also specify the plotting range for the independent variable (see page 319 in chapter 19). ([→] INDEP recalls the current independent variable, and its plotting range if specified.)
PTYPE		Selects the PTYPE menu.
CENT	CENTR	Takes a complex number (*x*,*y*) and makes it the center coordinate of the display. ([→] CENT recalls the current center coordinate.)

Basic Plotting Operations in the PLOTR Menu (continued)

Keys	Programmable Command	Description
SCALE	SCALE	Takes two real-number arguments. The first argument sets the *x*-scale in units per 10 pixels. The second argument sets the *y*-scale. (▶SCALE returns the *x*- and *y*-scales.)
RESET		Resets all plot parameters except the plot type to their default state and erases *PICT*, restoring it to its default size (131 pixels wide by 64 pixels high).
◀REVIEW		Redisplays the plot parameters.

Specifying the Independent Variable

For function, polar, and parametric plots, only the *independent* variable name is used to generate the plot. The default name of the independent variable is *X*. INDEP lets you specify a different independent variable, taking as its argument the name in level 1.

For example, suppose you want to plot:

```
'S=4*T^2+6'
```

To plot this equation, you must specify the new independent variable *T* by selecting the PLOTR menu and pressing ▶ T INDEP.

The role of the independent variable in building function plots is discussed in more detail in "How DRAW Plots Points" on page 298.

Display Ranges and Scaling

Specifying the Display Range. XRNG (*x*-axis display range) and YRNG (*y*-axis display range) let you specify the horizontal and vertical ranges of values represented by *PICT*, called the *display* ranges. By default, the display range along the *x*-axis is −6.5 to 6.5 units, and along the *y*-axis is −3.1 to 3.2 units. To specify a display range along the *x*-axis from −10 to 40 units, select the PLOTR menu and press 10 [+/−] [ENTER] 40 XRNG.

Note that the specification for the *y*-axis display range is only used when you plot the graph with DRAW. When you use AUTO, the *y*-axis display range is computed for you.

Specifying the Center and Scale. An alternative to specifying the display range is to *scale* the *x*- and *y*-axes using SCALE and specify the coordinates of the center of the display using CENTR. This approach is useful if you want the axes tick marks to represent meaningful values, for example, integer values, or when you want equal scaling for the two axes.

SCALE takes as its arguments two numbers representing the number of units per tick mark on each axis. CENTR takes a complex number argument representing the coordinates of a point in the plot, and makes that point the center of the display. For example, pressing (40,50) CENT 2 [SPC] 5 SCALE, specifies that plot coordinates (40,50) are located in the center of the display, that each *x*-axis tick mark represents 2 units, and that each *y*-axis tick mark represents 5 units.

Note that specifying the center coordinate and axes scales has the effect of determining the display ranges, and vice versa. In either case, the specifications are stored in *PPAR* as display ranges, and are shown in the PLOTR menu status message as display ranges.

Drawing the Graph

Autoscaling the y-Axis. Autoscaling provides a means of drawing a graph when you're unsure of the appropriate y-axis display range. For function plots, AUTO evaluates the equation at 40 values spaced equally across the range of the independent variable. It then computes an appropriate *y*-axis display range and executes DRAW to plot the graph.

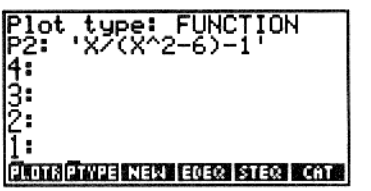
Using the Specified y-Axis Range. The DRAW operation is faster than AUTO and can be used if you've already specified an appropriate *y*-axis display range with YRNG.

Example: Plotting a Graph with AUTO. Plot the equation:

$$\frac{x}{x^2 - 6} - 1$$

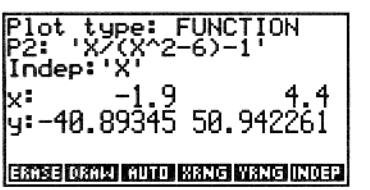
Key in the equation using the EquationWriter application. Name it *P2*.

◀ EQUATION
X ÷ X y^x 2 ▶
− 6 ▶ − 1
ENTER
◀ PLOT
NEW P2 ENTER



Select the PLOTR menu. If you worked the introductory example in this chapter, you'll see the following display. (Your *x*- and *y*-axis display ranges will vary slightly depending on the exact coordinates of the rectangular region you defined for Z-BOX in the introductory example.)

PLOTR



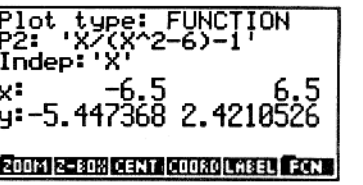
Reset the plot parameters and create a blank *PICT* of default size. Draw the graph using the default *x*-axis display range and autoscaling the *y*-axis display range. (The vertical lines in the plot represent the connecting of points at discontinuities in the function. See "Connected and Disconnected Plotting" on page 300.)

NXT RESET
◀ PREV AUTO



Press [◀][REVIEW] to check the PLOTR menu status message to see the newly computed *y*-axis display range. (Hold the [▼] key down to keep displaying the message).

[◀][REVIEW]



Press [ATTN] to exit the Graphics environment.

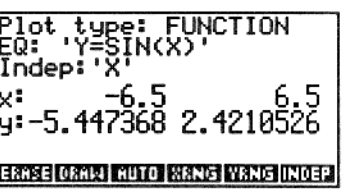
Example: Plotting a Graph with DRAW. Plot the equation:

$$y = \sin(x)$$

Specify a display range of −5 to 5 along the *x*-axis and −1.1 to 1.1 along the *y*-axis.

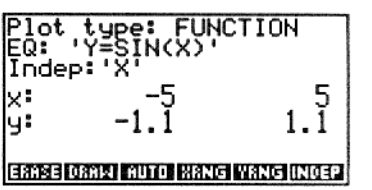
Select Radians mode, key in the equation, and store it directly into *EQ* without naming it. ([◀] DRAW executes STEQ. [▶] DRAW executes RCEQ.)

([◀][RAD] if necessary)
[ ] Y [◀][=] [SIN] X
[▶][PLOT] [◀] DRAW



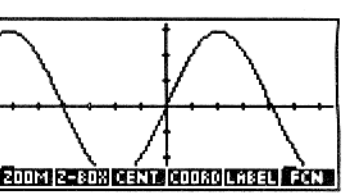
Erase *PICT*, and set the display ranges.

ERASE
5 [+/−] [ENTER] 5 XRNG
1.1 [+/−] [ENTER] 1.1 YRNG



Draw the graph.

DRAW



Press [ATTN] to exit the Graphics environment.

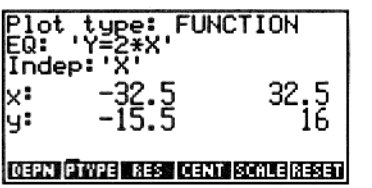
Example: Plotting a Graph Using CENTR and SCALE. Plot the equation:

$$y = 2x$$

To make the slope (2) "look" correct, use SCALE to specify equal scaling for both axes. Use CENTR to put the axes intersection (0,0) in the center of the display.

Key in the equation and store it directly into *EQ* without naming it. Select the PLOTR menu and specify the center and scale. For the scale, specify 5 units per tick mark.

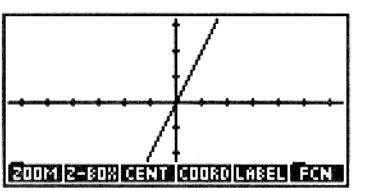
[ ] Y [←] [=] 2 [×] X
[←] [PLOT] STEQ
PLOTR [NXT]
[←] [( )] 0 [←] [,] 0 CENT
5 [SPC] 5 SCALE



Note how the display ranges are recomputed after you execute SCALE.

Now draw the graph.

[←] [PREV] ERASE DRAW



Press [ATTN] to exit the Graphics environment.

How DRAW Plots Points. In this section, it is necessary to reassert the normal distinction between equations, expressions, and programs, because, for function plots, DRAW interprets expressions and programs differently than equations.

- For expressions and programs, DRAW evaluates the expression or program in *EQ* for each of a series of values of the specified independent variable in the specified range, and plots each resultant point. That is, for independent variable x, DRAW plots each point $(x, f(x))$.

Suppose the expression in *EQ* is `'3*X'`, the *x*-axis range is 0 to 5, and the independent variable is *X*. DRAW evaluates `'3*X'` for a series of values of *X*, generating a series of points (*X*, 3*X*), starting at *X* = 0. Thus, the first plotted point is (0, 0) and the last plotted point is (5, 15). The number of values of the independent variable for which the expression is evaluated depends on the *resolution* (discussed on page 321 in chapter 19).

- For equations, the expression on the right side of the = sign is always plotted, just as above. The form of the left side of the equation and the setting of flag −30 determine whether the left side is plotted. If the left side is:
  - A *formal* variable name (a variable with no value stored in it), leave flag −30 *clear* to suppress plotting the left side and plot only the right side. For example, suppose you want to plot `'Y=3*X'`, where the independent variable is *X*, and *Y* is formal. In this case, leave flag −30 clear to plot only the expression `'3*X'`, just as you did above.
  - A variable with a meaningful value — as it is when plotting a SOLVR equation, for example — you have a choice whether or not to plot the variable. If you want to plot it, *set* flag −30 to plot two curves (x, y) and $(x, f(x))$. For example, suppose you want to plot `'Y=3*X'`, where the independent variable is *X*, and *Y* = 9. Set flag −30 to plot `'3*X'` just as you did above *and* to plot a straight line at *Y* = 9. (For another example, see the introductory example in chapter 17.) If you want to suppress plotting the variable, leave flag −30 clear.
  - If the left side of the equation is an *expression*, it is always plotted; that is, two curves $(x, f(x))$ and $(x, g(x))$ are plotted, independent of the state of flag −30. For example, if the equation is `'SIN(X)=COS(X)'`, and if the independent variable is *X*, DRAW plots two separate curves, one of `'SIN(X)'` and another of `'COS(X)'`.

The Dependent Variable in Function Plots. For function plots, the currently specified *dependent* variable is *ignored*. Coordinates of plotted points are generated simply by evaluating the current equation for a series of values of the independent variable.

Connected and Disconnected Plotting. By default, DRAW connects successive computed points with straight line segments. The connections are made regardless of the relative positions of the plotted points. This may be graphically undesirable, such as for a function with a discontinuity. (The example "Plotting a Graph with AUTO" on page 296 plots a function with multiple discontinuities, and connects each point.)

To turn connected plotting off, press ◀ MODES NXT CNC■ (or set flag −31). Pressing CNCT again switches connected plotting back on.

Plotting Two or More Equations

You can plot two or more equations with a single execution of DRAW or AUTO by creating a list that contains the equations or equation names. The EQ+ operation in the Equation Catalog provides a convenient way to build the list using equations from the Equation Catalog:

1. In the Equation Catalog, position the pointer at each equation you want included in the temporary menu and press EQ+. A list containing each selected equation is displayed and updated in the status area. (◀ EQ+ removes the last item from the list.)
2. Press PLOTR. The (unnamed) list is automatically stored in *EQ*.

Saving a List of Two or More Equations. The list you store in *EQ* by executing EQ+ and PLOTR is unnamed and therefore is not saved if you later change *EQ*. To name the list currently in *EQ*:

1. Execute RCEQ (◀ PLOT ▶ STEQ) to recall the list to level 1.
2. Execute NEW to give the list a name ending in EQ. Any variable whose name ends in EQ is included in the Equation Catalog.

If you want to name the list before storing it in *EQ*:

1. Build the list in the Equation Catalog with EQ+.
2. Press →STK to copy the list onto the stack.
3. Leave the Equation Catalog with ATTN and execute NEW to give the list a name ending in EQ.

The Graphics Environment—Zoom Operations and Function Analysis

When you execute DRAW or AUTO, the HP 48, after plotting the graph, enters the *Graphics environment* and displays the GRAPHICS menu. The Graphics environment remains active until you press ATTN to exit it and return to the stack. When you exit the Graphics environment, *PICT* persists—at any time, you can press ← GRAPH to return to the Graphics environment to view *PICT*.

The Graphics environment, like the Equation Catalog, is a special environment where the keyboard is redefined and limited to specific operations. You have access only to the GRAPHICS menu and its submenus. In addition, you do not have access to the stack when you are in the Graphics environment.

The operations in the Graphics environment can be divided into three categories with the following purposes:

- "Zooming" in or out from a region or the plot.
- Function analysis—obtaining mathematical data from the plot, such as the area under a curve.
- Adding graphical elements to the plot.

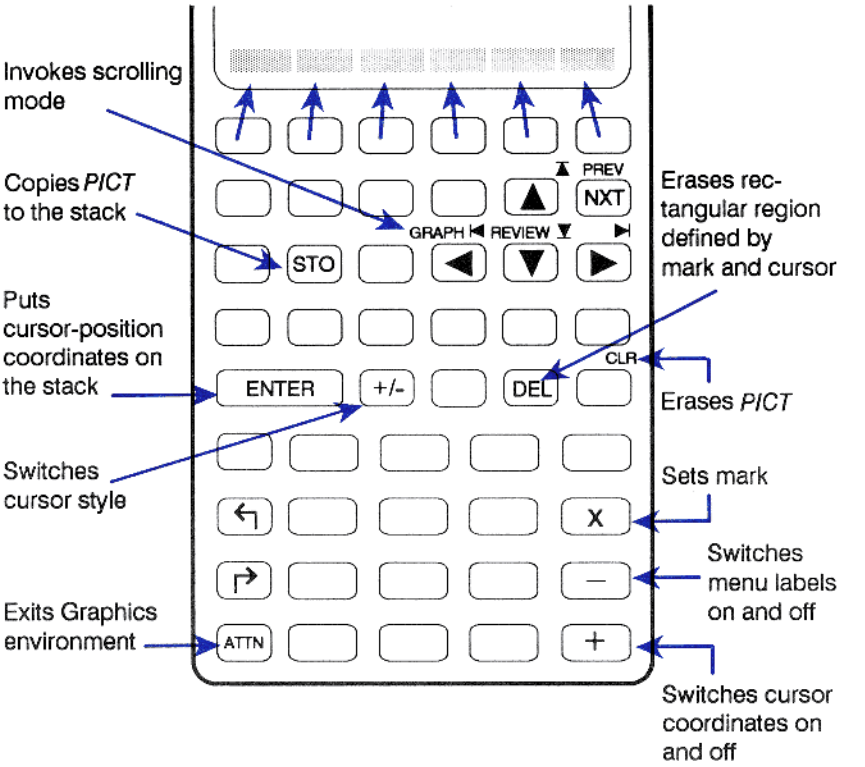
This section discusses zoom operations and function analysis. Adding graphical elements to the plot is discussed in chapter 19 on page 336.

Zoom and Function Analysis Operations

ZOOM	Displays the GRAPHICS ZOOM menu, which allows you to rescale and recenter the plot.
Z-BOX	Redraws the graph so that the rectangular area whose opposite corners are defined by the mark and cursor fills the display. ↰Z-BOX redraws the graph so that the *x*-range defined by the mark and cursor fills the display, and *autoscales* the *y*-axis.
CENT	Redraws the graph with the current cursor position at the center of the display.
COORD	Displays the coordinates of the cursor position, replacing the menu keys. Press any menu key to redisplay the menu labels.
LABEL	Adds axis labels to *PICT*.
FCN	Displays the GRAPHICS FCN menu for analyzing function plots (see "Analyzing Functions" on page 307).
MARK	Sets the mark. If no mark exists, the mark is created at the cursor. If the mark exists at another location, MARK moves the mark to the cursor location. If the mark exists at the cursor location, MARK erases the mark. (All operations that require a mark create a mark at the cursor location if no mark exists.)
+/−	Switches the cursor style. In the default state (+/−), the cursor is always dark. In the alternate state (+/−■), the cross is dark on a light background and light on a dark background.
KEYS	Erases the GRAPHICS menu keys, revealing more of the graph. Press ⊟ or any menu key to restore the GRAPHICS menu.

Zoom and Function Analysis Operations (continued)

▲ ▼ ◀ ▶	Moves the graphics cursor in the indicated direction. When prefixed with ↱, moves the cursor to the edge of the display. If the cursor is at the edge of the display *and* if *PICT* is larger than the display, prefixing with ↱ moves the cursor to the edge of *PICT*.
↰ GRAPH	Selects scrolling mode. In scrolling mode, the menu keys are erased, and, if *PICT* is larger than the display, pressing the cursor keys scrolls the display window over *PICT* in the indicated direction. Press ↰ GRAPH again to return to the normal Graphics environment display.
ENTER	Puts the coordinates of the cursor position on the stack.
×	Sets mark (same as MARK).
+	Switches the cursor coordinates on and off.
−	Switches the menu keys on and off.
+/−	Switches cursor style (same as +/−).
STO	Copies *PICT* to the stack.
↰ REVIEW	Temporarily displays the PLOTR menu status message. Holding the ▼ down displays the status message until the key is released.
↱ CLR	Erases *PICT*.
ATTN	Exits the Graphics environment.



Zoom Operations

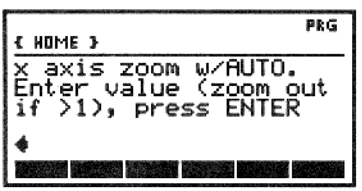
The zoom operations in the Graphics environment let you look at a particular region of the plot in more detail (by zooming in) or look at more of the plot than is currently displayed (by zooming out). The Z-BOX operation lets you zoom in on a box defined by the mark and cursor. The operations in the ZOOM menu let you zoom in and out along the x-axis, the y-axis, or both axes simultaneously.

The GRAPHICS ZOOM Menu. In the Graphics environment, press ZOOM to display a menu of zoom options.

Graphics Environment Zoom Options

XAUTO	Prompts for *x*-axis zoom factor and autoscales *y*-axis.
X	Prompts for *x*-axis zoom factor.
Y	Prompts for *y*-axis zoom factor.
XY	Prompts for single zoom factor to be applied to both axes.
EXIT	Exits to main GRAPHICS menu.

For example, if you press ZOOM XAUTO, you see the following display:



Each option changes the scaling of the specified axis (axes) by the specified zoom factor and redraws the graph using the cursor position as the new center.

Example: Zooming Out. Identify the number of *x*-axis intercepts of the expression:

$$x^2 - 9x - 10$$

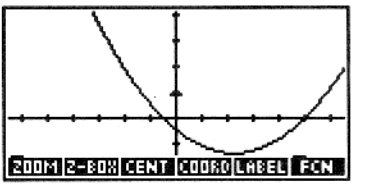
Select the PLOTR menu and store the expression in *EQ*. Then reset the plot parameters. Draw the graph using autoscaling.

▶ PLOT
' X y^x 2 − 9 × X − 10
◀ DRAW
NXT RESET
NXT NXT AUTO



The expression appears to have a second x-axis crossing outside the display range. Zoom out along the x-axis by a factor of 2.

ZOOM X 2 ENTER



The expression has two x-axis crossings.

Zoom-To-Box. Z-BOX (zoom-to-box) lets you zoom *in* on a specified portion of the display:

1. Move the cursor to one corner of the desired area and press Z-BOX (or MARK or ⊠) to mark the location.
2. Move the cursor to the diagonally opposite corner.
3. Press Z-BOX again to rescale and redraw the plot.

Optionally, if the current scaling along the y-axis is satisfactory:

1. Move the cursor to one corner of the desired area and press Z-BOX (or MARK or ⊠) to mark the location.
2. Move the cursor to the horizontally opposite corner.
3. Press Z-BOX to redraw the plot, rescaling it only along the x-axis.

Similarly, you can define vertically opposite corners to rescale only along the y-axis.

Zoom-To-Box with Autoscaling. If you want to zoom in on a new x-axis display range, but you're unsure of the appropriate scale along the y-axis, you can zoom-to-box with autoscaling as follows:

1. Move the cursor to one corner of the desired area and press Z-BOX (or MARK or ⊠) to mark the location.
2. Move the cursor to the horizontally opposite corner.
3. Press ↰Z-BOX to rescale and redraw the plot. The y-axis display range is *autoscaled*.

The example on page 312 uses Z-BOX with autoscaling to identify the roots of the equation.

Analyzing Functions

The GRAPHICS FCN menu lets you analyze the mathematical behavior of plotted functions simply by specifying with the graphics cursor the region or point of interest on the function and then executing the desired calculation from the menu. You can calculate:

- Function values.
- Slopes.
- Areas under curves.
- Roots.
- Extremums or inflection points.
- Intersections of two functions.

When you execute a calculation:

- The cursor moves to the corresponding point on the function (if that point is in the display).
- A message is displayed in the lower left corner of the display showing the result as a tagged object.
- The result is returned to the stack as a tagged object.

You can also plot the derivative of a plotted function.

The current plot type must be FUNCTION and *EQ* must contain an equation, expression, or a list of expressions or equations. It cannot contain a program.

If you've plotted two or more equations by storing a list in *EQ* (see page 300), the function analysis operations use the first equation in the list, unless otherwise stated. NXEQ is provided in the FCN menu to rotate another equation to the beginning of the list.

The GRAPHICS FCN Menu

Key	Description
ROOT	Moves the cursor to a root (intersection of the function with the *x*-axis) and displays the value of the root. If the root is not in the display window, ROOT briefly displays the message OFF SCREEN before displaying the value of the root.
ISECT	Same as ROOT if only one function is plotted. If two or more functions are plotted, ISECT 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, ISECT briefly displays the message OFF SCREEN before displaying the coordinates of the intersection.
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	Calculates and displays the area beneath the curve between two *x*-values defined by the mark and cursor.
EXTR	Moves the cursor to an extremum (local minimum or maximum) or inflection point and displays the (*x*,*y*) coordinates. If the closest extremum or inflection point is not in the display window, EXTR briefly displays the message OFF SCREEN before displaying the value.
EXIT	Exits the GRAPHICS FCN menu back to the main GRAPHICS menu.
F(X)	Displays the function value at the current *x*-value of the cursor, and moves the cursor to that point on the function curve.

The GRAPHICS FCN Menu (continued)

Key	Description
F'	Plots the first derivative and replots the original function. Also adds the expression that defines 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 appends the expression to the front of the list.)
NXEQ	Rotates the list in *EQ* so that the second equation is moved to the beginning of the list and the first equation is moved to the end of the list. Also displays the equation now at the beginning of the list.

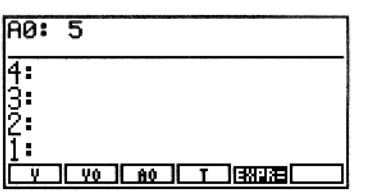
Example 1: Function Analysis. An equation for constant acceleration is:

$$v = v_0 + a_0 t$$

For an initial velocity $v_0 = 10$, and a constant acceleration $a_0 = 5$, find the velocity at $t = 2$ and find the total displacement x between $t = 0$ and $t = 10$. (The displacement is the area under the curve of velocity vs. time.)

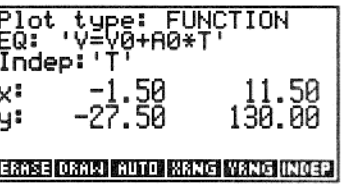
Select the SOLVE menu, key in the equation , and store it into *EQ* without naming it. Select the SOLVR menu and store the values for v_0 and a_0. The SOLVR menu lets you easily store known values for the current equation.

◀ SOLVE
▯ V ◀ = V0 + A0 × T
STEQ
SOLVR 10 V0 5 A0



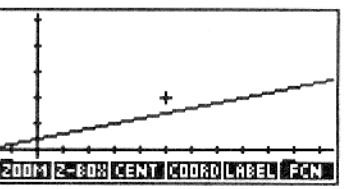
Set the display mode to 2 Fix so that coordinates and function analysis results are easy to read in the Graphics environment. Then select the PLOTR menu. To obtain integer values for the x- and y-axis tick marks, use SCALE to specify 1 unit per x-axis tick mark and 25 units per y-axis tick mark. This will enable exact calculations. Use CENT to specify the plot center at (5,50). Finally, specify T as the independent variable.

◀ MODES 2 FIX
▶ PLOT NXT
1 SPC 25 SCALE
◀ ( ) 5 SPC 50 CENT
◀ PREV ' T INDEP



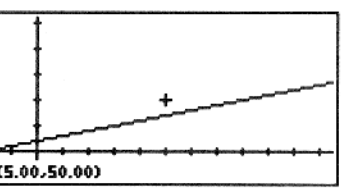
Erase *PICT*, then draw the graph.

ERASE DRAW



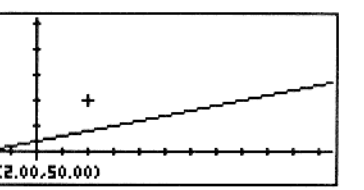
Check the coordinates of the graphics cursor.

COORD (or +)



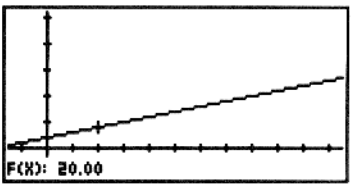
The x-coordinate (in this example the value of T) is 5.00, so hold down the ◀ key until the displayed x-coordinate is exactly 2.00. Note that the cursor moves more slowly when coordinates are displayed.

hold down ◀



Press any menu key (or ▲ or ▼) to redisplay the menu labels. Then select the FCN menu and find the value of the function.

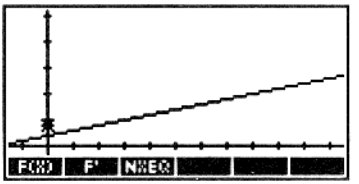
▲
FCN NXT F(X)



The velocity is 20 at $T = 2$.

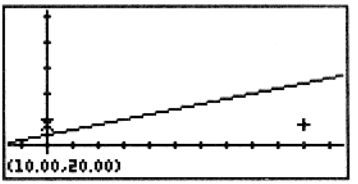
Now calculate the displacement between $T = 0$ and $T = 10$. First, restore the menu keys. Then move the cursor to the y-axis ($T = 0$) and set the mark.

▲
hold down ◀
×



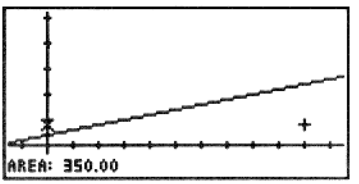
Display the cursor coordinates, move the cursor to the right edge of the display, then back until its x-coordinate is `10.00`.

▲
▶▶
then hold down ◀



Redisplay the menu labels, and calculate the area.

▲ NXT AREA



The displacement at $T = 10$ is 350.

Return to the stack and note that the function value and area have been returned to the stack as tagged objects.

ATTN ATTN



Example 2: Function Analysis. Part 1. For the expression:

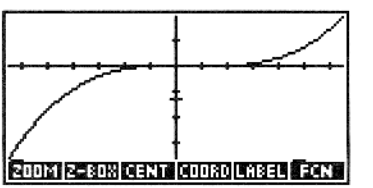
$$x^3 - 2x^2 - x + 2$$

find the following:

- The number of real roots.
- The value of the leftmost root.
- The slope of the expression at the leftmost root.
- The value of the expression at the y-axis ($x = 0$).
- The coordinates of the local minimum.

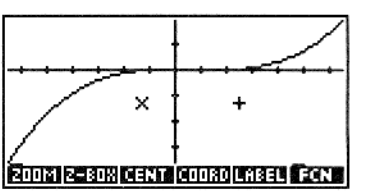
Select the PLOT menu, key in the expression, and store it unnamed in *EQ*. Select the PLOTR menu, reset the plot parameters. Then draw the graph using autoscaling for the y-axis display range.

◀ PLOT
' X y^x 3 − 2 × X y^x 2
− X + 2 STEQ
PLOTR NXT RESET
◀ PREV AUTO



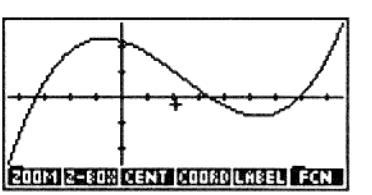
The region of interest needs enlargement, so set the mark and cursor as shown.

hold down ◀
×
hold down ▶



Now zoom-to-box, autoscaling the y-axis.

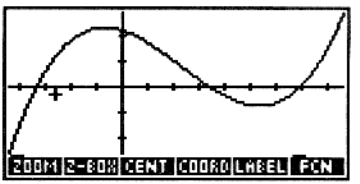
◀ Z-BOX



You can now see that there are three real roots in this region.

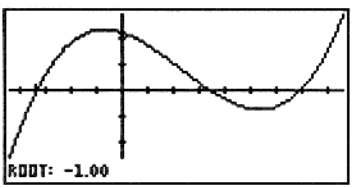
Move the cursor near the leftmost root.

hold down ◀



Find the value of the root.

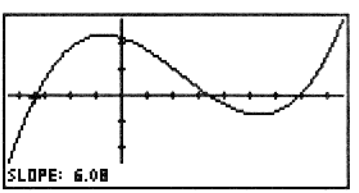
FCN ROOT



The cursor moves to the root and the value of the root is displayed in the lower left corner.

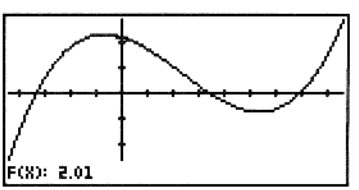
Calculate the slope of the function at the root. (Press any key to redisplay the menu labels.) The value you obtain for the slope will vary slightly from that shown in the following display, depending on the exact coordinates of the rectangular region you defined with ↰Z-BOX.

− SLOPE



Find the value of the function at the y-axis ($x = 0$): restore the menu, then move the cursor to the y-axis and execute F(X).

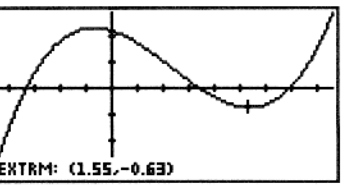
+
then hold down ▶
NXT F(X).



The cursor moves to the corresponding point on the function.

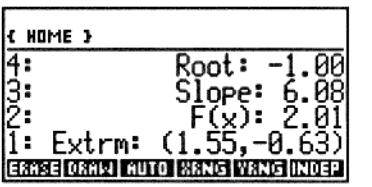
Now find the coordinates of the local minimum: restore the menu, then move the cursor to an x-axis value near the minimum and execute EXTR.

+
hold down ▶
NXT EXTR



Leave the Graphics environment and note that the results have been returned to the stack as tagged objects.

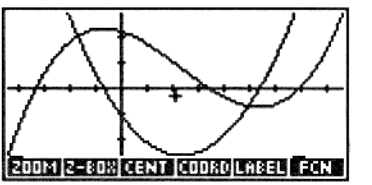
ATTN ATTN



Part 2. Plot the derivative of the expression and find the coordinates of the positive x-axis intersection of the derivative and the original expression.

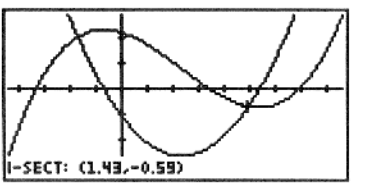
Return to the Graphics environment and plot the derivative.

↰ GRAPH FCN NXT F'



Move the cursor near the positive x-axis intersection and execute ISECT.

hold down ▶
FCN ISECT



Return to Standard display mode by pressing ↰ MODES STD.

Working with Difficult Plots. The examples in this chapter have generated plots in which the intersection of the x- and y-axes is visible in the display, providing you with immediate orientation. However, depending on the expression and the current display ranges, one or both axes may not be visible. In such cases, press ◀ REVIEW to determine what part of the graph you are looking at. For example, suppose you plot a graph with autoscaling and do not have an x-axis in your graph. If you press ◀ REVIEW and see a y-axis display range from 230 to 410, you know that the portion of the graph you are currently viewing is above the x-axis. At this point you have several options:

- If you want to better understand the general shape of the function and its relationship to the axes, you can zoom out to see more of the function. XAUTO is particularly suitable for such exploratory zooming.
- If you want to identify a particular feature of the function, such as a root or extremum, you can execute the corresponding operation in the GRAPHICS FCN menu to return the coordinates of that feature to the stack. Then leave the Graphics environment and execute CENT from the PLOTR menu to bring the feature into view when you redraw the graph. Analysis of the function's shape at the feature may provide insight into the relative position of other features on the curve. Subsequent zoom operations may then be appropriate.

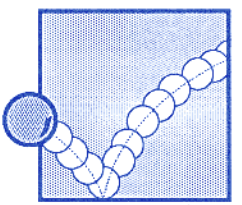
How the Operations in the GRAPHICS FCN Menu Work. The operations in the GRAPHICS FCN menu are linked to commands that are executable outside the Graphics environment. (In this section, the normal distinction between expressions and equations is reasserted.)

How the Function Analysis Operations Work

ROOT	Executes ROOT (the numerical root-finder in the HP Solve application) to find an *x*-axis intersection. If there are multiple roots (intersections), 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	Executes ROOT: For a single expression or for an equation whose left side has not been plotted (flag −30 clear), ISECT works just like ROOT; for an equation whose left and right sides have been plotted (flag −30 set), it finds the nearest intersection of the left and right sides; for two expressions, it finds the nearest intersection of the expressions; for two equations, it finds the nearest intersection of the right sides.
SLOPE	Executes ∂, then evaluates the resultant expression at the *x*-value of the cursor.
AREA	Executes $\int$, using the *x*-values defined by the mark and cursor as limits.
EXTR	Executes ∂, then finds the *x*-value closest to the cursor that causes the resultant expression to evaluate to zero.
F(X)	Evaluates the expression at the *x*-value defined by the cursor.
F'	Executes ∂, then puts the resultant expression in a list in *EQ* with the original expression, and plots the list.

19

More About Plotting and Graphics Objects



The previous chapter covered basic plotting of mathematical functions. The plot type was specified as FUNCTION in all examples and a limited set of plot parameters was discussed and utilized. This chapter builds on the concepts introduced in chapter 18. It covers:

- Specifying refinement options for plots:
  - Plotting part of a display range.
  - Specifying axes labels that are different from the independent and dependent variables.
  - Specifying a different sampling frequency.
- Working with plot coordinates.
- Changing the size of *PICT*.
- Drawing conic, polar, parametric, truth, and statistical plots.
- Plotting programs and user-defined functions.
- Plotting with units.
- Adding graphical elements to *PICT*.

This chapter also discusses commands for working with graphics objects on the stack. These graphics objects commands, which are particularly useful in programs, let you build and animate custom graphical images.

Refinement Options for Plots

The following commands in the PLOTR menu let you tailor a number of plot features.

Commands in the PLOTR Menu For Refining Plots

Keys	Programmable Command	Description
◀ PLOT PLOTR:		
INDEP	INDEP	Sets the name in level 1 as the independent variable. INDEP can also specify the *plotting* range for the independent variable. (▶ INDEP recalls the current independent variable, and its plotting range if specified.)
DEPN	DEPND	Sets the name in level 1 as the dependent variable. (The dependent variable specification is used for conic and truth plots. See pages 329 and 333.) DEPND can also specify the plotting range for the dependent variable. (▶ DEPN recalls the current dependent variable, and its plotting range if specified.)
RES	RES	Sets the plot *resolution*. (▶ RES recalls the current resolution.)

Commands in the PLOTR Menu For Refining Plots (cont.)

Keys	Programmable Command	Description
AXES	AXES	Sets the coordinates of the axes intersection, using the complex-number argument from level 1. AXES can also specify axes labels that are different than INDEP and DEPND. (→ AXES returns the current axes intersection.)
DRAX	DRAX	Adds axes to *PICT*. (Not necessary when DRAW or AUTO is executed from the keyboard.)
LABEL	LABEL	Adds axes labels to *PICT*.
*H	*H	Multiplies the horizontal scale by the level 1 argument *n*. Zooms *in* if $n < 1$.
*W	*W	Multiplies the vertical scale by the level 1 argument *n*. Zooms *in* if $n < 1$.
PDIM	PDIM	Changes the size of *PICT*. (→ PDIM returns the size of *PICT*.)
← REVIEW		Redisplays the plot parameters.

Plotting Range Versus Display Range

The range of the independent variable for which the current equation is evaluated is called the *plotting* range. Unless otherwise specified, the HP 48 uses the x-axis display range (specified by XRNG) as the plotting range. However, you can use INDEP to specify a plotting range that is different from the x-axis display range. For conic and truth plots (pages 329 and 333), which require specification of the *dependent* variable, you can use DEPND to specify, for the dependent variable, a plotting range that is different from the y-axis display range.

INDEP and DEPND take two numbers from the stack. For example, `0 10 INDEP` specifies plotting in the independent variable range 0 through +10. You can also specify both a variable name and a range by supplying a list argument of the form:

`{ 'name' lower upper }`

where *name* is the variable name and *lower* and *upper* define the lower and upper limits of the plotting range. For example, `{ T 0 10 } INDEP` specifies plotting for values of the independent variable *T* in the range 0 through +10.

There are two situations for which specification of a plotting range is valuable:

- In parametric plots (page 332), the *x*-axis display range is unrelated to the appropriate plotting range for the independent variable, so you should always specify the plotting range using INDEP (see the example on page 332).
- For truth plots (page 333), specification of plotting ranges that are smaller than the *x*- and *y*-axis display ranges shortens plotting time (see the example on page 333).

Specifying Axes and Labels

If the axes are in the plotting range, `AUTO` and `DRAW` automatically draw them with tick marks placed at 10-pixel intervals. The axes intersect at (0,0) unless you specify otherwise using AXES. LABEL displays in *PICT* the names of the independent and dependent variables, and the coordinates of the end-points of the axes (using the current display format).

AXES lets you specify axes labels that are different from the independent and dependent variables using a list argument. For example, executing `{ (0,0) "X2" "F(X2)" } AXES` assigns the label `X2` to the horizontal axis and the label `F(X2)` to the vertical axis, regardless of the names of the independent and dependent variables. Subsequent execution of LABEL displays these labels in *PICT*.

Specifying Resolution

The RES (resolution) command determines the interval between values of the independent variable used to generate the plot. RES takes either a real number or binary integer argument. For all plot types, a real number argument determines the interval in user units. For FUNCTION, CONIC, and TRUTH plot types, a binary integer argument determines the interval in pixels. For POLAR and PARAMETRIC plot types, a binary integer argument does not apply. A real number argument 0 or binary integer argument # 0 specifies the default interval value, summarized in the following table.

Plot Type	Default Interval
FUNCTION, CONIC, and TRUTH	1 pixel. (A point is plotted in every column of pixels).
POLAR	2°, 2 grads, or $\pi/90$ radians.
PARAMETRIC	[independent variable range (in user units)]/130

Increasing RES (plotting fewer points) yields faster plots. However, the accuracy of the line connecting the plotted points decreases.

How RES Affects Statistical Plots. For statistical plots, a real number argument specifies user units and a binary integer argument specifies pixels as follows:

- For BAR plot type, RES determines the bar width.
- For HISTOGRAM plot type, RES determines the bin width.
- RES does not apply to SCATTER plot type.

The Plot Parameter Variable (PPAR)

The HP 48 uses a built-in variable named *PPAR* to store the plotting parameters. *PPAR* contains a list with the following objects:

$$\{\ (x_{min}, y_{min})\ (x_{max}, y_{max})\ \textit{indep res axes ptype depend}\ \}$$

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. Meaning varies for statistical data.	0 (Points are 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.	(0,0)
ptype	Command name specifying the plot type.	FUNCTION

Contents of the PPAR List (Continued)

Element	Description	Default
depend	Dependent variable. The name of the variable, or a list containing the name and two real numbers (the vertical plotting range).	*Y*

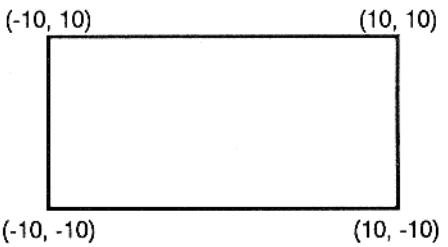
Since *PPAR* is a variable, a different *PPAR* can exist for every directory.

Resetting PPAR. The RESET operation resets the parameters in *PPAR* (except the plot type) to their default states. RESET also erases *PICT* and restores it to its default size.

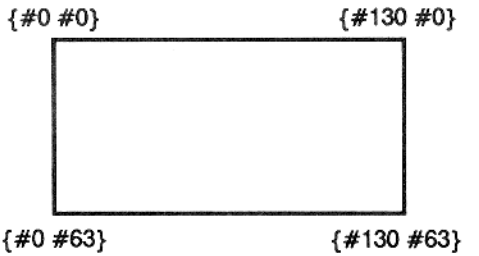
Plot Coordinates

The size of *PICT* (or any graphics object on the stack), and the position of a point within it, are expressed in terms of horizontal and vertical coordinates. There are two unit systems for coordinates:

- *User-unit* coordinates (until now simply called "units") are represented by a complex number. They are interpreted according to the first two parameters in *PPAR*; (x_{min}, y_{min}) and (x_{max}, y_{max}). For example, if (x_{min}, y_{min}) is $(-10, -10)$ and (x_{max}, y_{max}) is (10, 10), coordinates $(-10, 10)$ represent the upper-left pixel in the graphics object. (Graphics objects on the stack do not have user-unit coordinates.)



- *Pixel* coordinates are represented by a list containing two binary integers; {#0 #0} represents the upper-left pixel.



Two commands in the PRG DSPL menu let you convert a given point to the alternate coordinate system.

Coordinate Conversion Commands

Keys	Programmable Command	Description
PRG DSPL (page 2):		
PX→C	PX→C	Converts pixel coordinates to user-unit coordinates. Takes the list argument { #n #m } from level 1 (where #n is the row coordinate and #m is the column coordinate) and returns (x,y).
C→PX	C→PX	Converts user-unit coordinates to pixel coordinates. Takes (x,y) from level 1 and returns { #n #m } to level 1.

Changing the Size of PICT

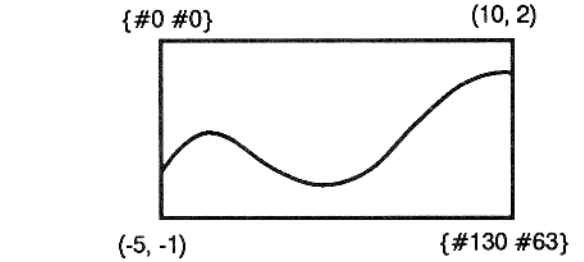
You can make *PICT* larger by executing the PDIM (*PICT* dimension) command. PDIM takes two arguments, either user-unit coordinates or pixel dimensions:

- User-unit coordinates specify the lower left corner (in stack level 2) and upper right corner (in stack level 1).
- Pixel dimensions specify the horizontal dimension (in stack level 2) and vertical dimension (in stack level 1).

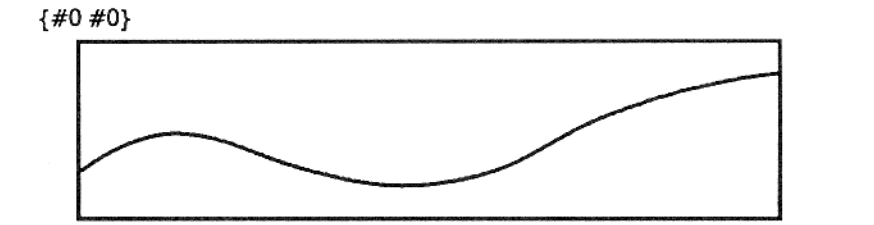
While both argument forms change the size of *PICT*, they have different effects on the scaling of the axes.

For example suppose that *PICT* is currently its default size (#131 wide by #64 high in pixel units), that the current *x*-axis display range is −5 to 10, that the current *y*-axis display range is −1 to 2, and that *PICT* contains the graph shown in figure (a) on page 326:

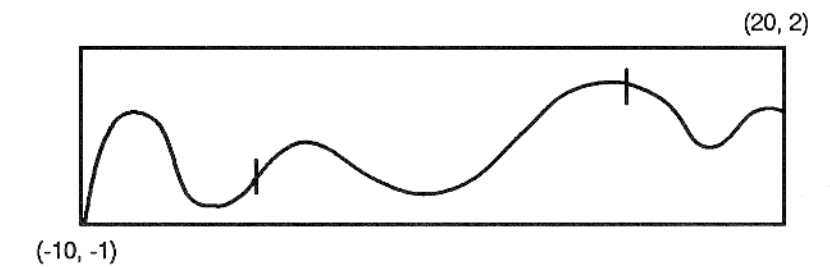
- If you execute `#262 #64 PDIM`, the size of *PICT* doubles in the horizontal direction and the display ranges remain unchanged, so the scale of the *x*-axis doubles. If you redraw the graph, the effect is to "stretch" the graph (figure (b)).
- If you execute `(-10,-1) (20,2) PDIM`, the size of *PICT* doubles in the horizontal direction (to #262 wide by #64 high in pixel units), the *x*-axis display range becomes −10 to 20, and the *y*-axis display range remains −1 to 2. In this case, the scale of the axes does *not* change. If you redraw the graph, the effect is to add more points to the graph at either end (figure (c)).



(a)



(b)



(c)

Changing the Size of *PICT*

Plot Types

The plot type tells the HP 48 how to interpret the current equation (or current statistical data for statistical plot types). The PLOT menu status message indicates the current plot type.

There are eight plot types:

- Five types plot equations: FUNCTION, CONIC, POLAR, PARAMETRIC, and TRUTH.
- Three types plot statistical data using the current statistical matrix (the contents of Σ*DAT*): SCATTER, HISTOGRAM, and BAR.

The default plot type is FUNCTION. To set the plot type, select the PTYPE menu (press PTYPE in the PLOT or PLOTR menu), then press the appropriate menu key to execute the associated command.

Plot Types That Use the Current Equation

Type	Description
FUNCTION	Plots equations that return a unique f(*x*) for each value of *x*.
CONIC	Plots conic sections — circles, ellipses, parabolas, and hyperbolas.
POLAR	Plots expressions that return the radius for each value of the specified polar angle.
PARAMETRIC	Plots equations that return a complex result for each value of the specified independent variable.
TRUTH	Plots expressions that return a true (1) or false (0) value, such as equations with comparison functions.

Plot Types That Use the Current Statistical Matrix

BAR	Draws a bar chart of the data from a specified column (XCOL) of the statistical matrix.
HISTOGRAM	Draws a frequency histogram of the data from a specified column (YCOL) of the statistical matrix.
SCATTER	Plots points from two columns (XCOL and YCOL) of the statistical matrix.

Function Plots

The default plot type is FUNCTION. All the examples in chapter 18 used the FUNCTION plot type.

The FUNCTION Plot Type

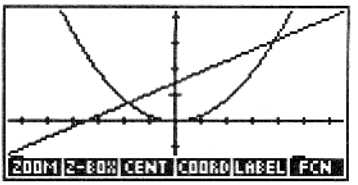
Form of Current Equation	Example	Points Plotted
$f(x)$	$x^3 - 5x^2 - 10x + 20$	$(x, f(x))$
$y = f(x)$	$y = x^2 + x + 4$	flag −30 clear: $(x, f(x))$ flag −30 set: $(x, f(x))$ and (x, y)
$f(x) = g(x)$	$x^2 = 2x + 7$	$((x, f(x))$ and $(x, g(x))$

Example: Plotting an Equation. Plot the equation:

$$x^2 = 2x + 7$$

Make the equation the current equation (unnamed). Set the plot type to FUNCTION. Reset the plotting parameters. Then, plot an autoscaled graph.

```
↰ PLOT
' X y² 2
↰ = 2 × X + 7 STEQ
PTYPE FUNC
PLOTR NXT RESET
NXT NXT AUTO
```



The x-values at which the two plots intersect are roots of the equation.

Conic Sections

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:

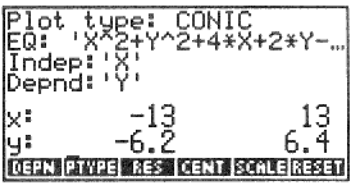
$x^2 + y^2 + 4x + 2y - 10 = 0$	Circle
$5x^2 + 3y^2 - 18 = 0$	Ellipse
$x^2 - 4x + 3y + 2 = 0$	Parabola
$2x^2 - 3y^2 + 3y - 5 = 0$	Hyperbola

Note that the variable specified by DEPND *is* used when the plot type is CONIC. Also note that autoscaling is not useful for conic sections. Use CENT and SCALE instead.

Example: A Conic Section. Plot the conic section for the equation:

$$x^2 + y^2 + 4x + 2y - 5 = 0$$

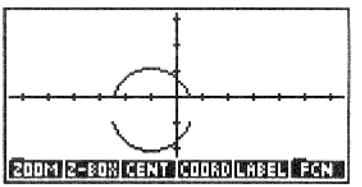
Select the PLOTR menu and key in the equation. Store the (unnamed) equation in *EQ*. Set the plot type to CONIC. Set the plot parameters. Use CENT and SCALE to draw a "round" circle.



[↱] [PLOT]
['] X [y^x] 2 [+]
Y [y^x] 2 [+] 4 [×] X
[+] 2 [×] Y [−] 5 [↰] [=] 0
[↰] DRAW
[NXT] PTYPE CONIC
[NXT] [NXT] ['] X INDEP
[NXT] ['] Y DEPN
[↰] [( )] 0 [SPC] 0 CENT
2 [SPC] 2 SCALE

Plot the conic section.

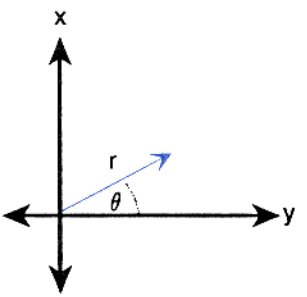
[↰] [PREV]
ERASE DRAW



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, as in the previous example. Specifying a finer resolution (decreasing the interval between plotted points) helps eliminate any discontinuities (see "Specifying Resolution" on page 321).

Polar Plots

In polar plots, the polar angle is the independent variable.



Polar Plots

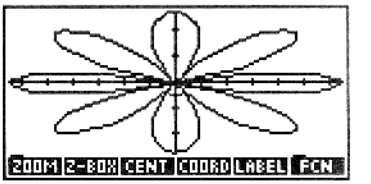
Form of Function	Example	Points Plotted
$f(\theta)$	$\cos(\theta) + \sin(\theta)$	$(f(\theta), \theta)$
$r = f(\theta)$	$r = 2\cos(\theta)$	$(f(\theta), \theta)$
$\theta = constant$	$\theta = 0.2\pi$	Radial line
$f(\theta) = g(\theta)$	$4\sin(\theta) = r^2$	$(f(\theta), \theta)$ and $(g(\theta), \theta)$

Unless you specify otherwise, the plots are drawn for a full circle of θ (0 through 360 degrees, 2π radians, or 400 grads, according to the current angle mode). To specify a different plotting range for θ, supply a list argument for INDEP (see "Plotting Range Versus Display Range" on page 319). If you use autoscaling, the HP 48 computes an appropriate x- and y-axis display range based on the θ-range, but note that the resulting x- and y-axis scales may differ.

Example: A Polar Plot. Plot the polar equation r = 2 cos(4θ) for values of θ in the range 0° through 360°.

Set Degrees mode. Store the equation in *POL*. (To key in θ, press [α] [→] [F]). Select POLAR plot type. Specify the independent variable θ. Then draw the plot using autoscaling.

[←] [RAD] if necessary
[←] [PLOT]
' R [←] [=] 2 [×] [COS] 4 [×] θ
NEW POL [ENTER]
PTYPE POLAR
PLOTR ' θ INDEP
AUTO



In this example, autoscaling generates different x- and y-axis scales, compressing the plot in the vertical direction.

Parametric Plots

In parametric equations, two dependent variables (typically x and y), represented by the horizontal and vertical axes, are expressed as functions of an independent variable (typically t).

For example, consider these parametric equations:

$$x = t^2 - t \quad \text{and} \quad y = t^3 - 3t$$

where t is the independent variable.

To plot these equations, they must be written as an expression or program that returns a complex result $x + yi$:

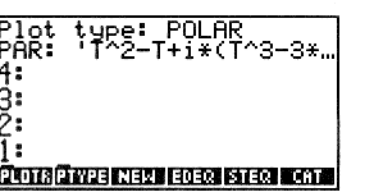
```
'T^2-T+i*(T^3-3*T)'
```

For parametric plots, the x-axis display range is unrelated to the appropriate plotting range for the independent variable, so you should always specify the plotting range using INDEP (see "Plotting Range Versus Display Range" on page 319.) If you use autoscaling, the HP 48 computes an appropriate x- and y-axis display range based on the plotting range of the independent variable.

Example: A Parametric Plot. Plot the equations shown above for values of t in the range −3 through +3.

Key in and store the expression. Name it *PAR*. (To key in the complex number i, press [α] [←] and then [CST].)

[ ] T [y^x] 2 [−] T [+]
i [×] [←] [( )] T [y^x] 3
[−] 3 [×] T
[←] [PLOT] NEW
PAR [ENTER]



Set the plot type to PARAMETRIC and specify the independent variable and its plotting range. Draw the graph using autoscaling.

PTYPE PARA
PLOTR [←] [{ }] T [SPC] 3 [+/−] [SPC]
3 [ENTER] INDEP
AUTO



Truth (Relational) Plots

Truth plots evaluate expressions that return true (any non-zero real number) or false (0) results. Each pixel for which the expression returns true is turned *on*, and each pixel for which the expression returns false remains *unchanged*. Unless otherwise specified, every pixel in the display is evaluated.

The variable specified by DEPND *is* used for truth plots.

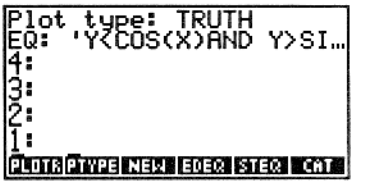
Example: A Truth Plot. Draw a truth plot for the expression:

```
'Y<COS(X)AND Y>SIN(X)'
```

Specify an x-axis display range of $-\pi$ to $\pi/2$ (radians) and a y-axis display range of -1.5 to 1.5. To shorten the plotting time, specify a plotting range of -2.4 to .85 (radians) for X and -1.1 to 1.2 for Y.

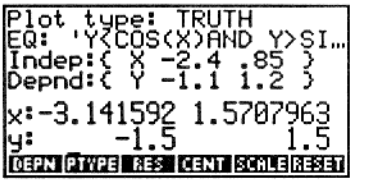
Select Radians mode. Key in the expression, and store it in EQ. (To type <, press [α] [←], then [2]. To type >, press [α] [→], then [2].) Specify the plot type.

[←] [RAD] if necessary
[ ] Y < [COS] X [▶]
[PRG] TEST AND
Y > [SIN] X
[←] [PLOT] STEQ
PTYPE TRUTH



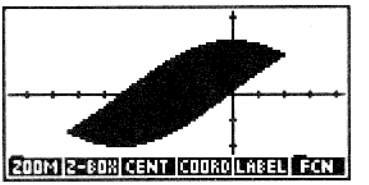
Specify the x- and y-axis display ranges. Specify the independent and dependent variables and their plotting ranges.

PLOTR
[←] [π] [+/−] [→] [→NUM]
[ENTER] 2 [÷] [+/−] XRNG
1.5 [+/−] [SPC] 1.5 YRNG
[←] [{ }] X [SPC] 2.4 [+/−] [SPC] .85
[ENTER] INDEP
[←] [{ }] Y [SPC] 1.1 [+/−] [SPC] 1.2
[ENTER] [NXT] DEPN



Draw the plot. (The HP 48 takes about seven and one-half minutes to draw the plot.)

◀ PREV ERASE DRAW



Plotting Programs and User-Defined Functions

In addition to plotting expressions and equations, the HP 48 can plot:

- Programs equivalent to expressions $f(x)$ (type FUNCTION) or $r(\theta)$ (type POLAR). The program must not take any values from the stack, and it must return exactly one value (untagged) to the stack. For example, the program:

  ```
  « IF 'X<10' THEN '3*X^3-45*X^2+350' ELSE 1000 END»
  ```

  plots:

  $$f(x) = \begin{cases} 3x^3 - 45x^2 + 350 & \text{for } x < 10 \\ 1000 & \text{for } x \geq 10 \end{cases}$$

- Programs that return a single complex result (type PARAMETRIC). For example, plotting the program:

  ```
  « 't^2-2' →NUM 't^3-2t+1' →NUM R→C »
  ```

  plots the parametric equations:

  $$x = t^2 - 2 \quad \text{and} \quad y = t^3 - 2t + 1$$

- User-defined functions written as a function of one variable. For example, if you've created the *COT* (cotangent) user-defined function, you can plot the expression `COT(X)`, where X is the independent variable.

Note that you cannot use the operations in the GRAPHICS FCN menu with plots of programs and user-defined functions.

Plotting with Units

The HP 48 lets you plot equations that contain unit objects with the following conditions:

- If the independent variable requires units for *EQ* to evaluate properly, you must store a unit object in the independent variable before plotting. (The number part of the unit object is ignored.)
- If evaluation of *EQ* returns a unit object, only the scalar part of the unit object is used for plotting.

Note that you cannot associate units with the *x*- or *y*-axis display ranges. Thus, if the intended units of the *x*-axis display range is m (meters), *no* conversion to meters is executed if the independent variable has units of ft (feet).

Plotting Statistical Data

Plots of statistical data are similar to plots of mathematical data, except that:

- The data comes from the reserved variable Σ*DAT*, rather than from *EQ*.
- Instead of specifying independent and dependent variables in *PPAR*, you specify analogous *columns* of statistical data in the reserved variable Σ*PAR*.

The Statistics application provides the easiest way to plot statistical data. Refer to "Plotting" in chapter 21.

You can also plot statistical data from the Plot application by selecting one of the statistical plot types: BAR, HISTOGRAM, or SCATTER. When you do:

- The status message in the PLOT menu changes to show you the contents of Σ*DAT*, rather than *EQ*.
- The status message in the PLOTR menu changes to show you the contents of Σ*DAT*, the columns in Σ*DAT* specified by XCOL and YCOL, and the currently specified statistical model.

Use DRAW or AUTO to plot the graph.

The Plot application lets you specify plot parameters for statistical plots that are unavailable to you in the Statistics application. Here are some examples:

- RES lets you specify the number of bins in a histogram plot.
- CENTR and SCALE let you specify, for a scatter plot, display ranges that are larger than the range of plotted points.
- AXES lets you specify labels for the axes in a bar plot.

Adding Graphical Elements to PICT

Operations in the Graphics environment and analogous programmable commands in the PRG DSPL menu let you add graphical elements to *PICT*.

Operations In the Graphics Environment for Adding Graphical Elements to PICT

Adding Graphical Elements to PICT in the Graph. Env.

DOT+	Activates line-drawing; pixels beneath the cursor are turned on as the cursor is moved across the display. While line-drawing is active, a ▪ is appended to the key (DOT+▪).
DOT-	Activates line-erase. Pixels beneath the cursor are turned off as the cursor is moved across the display. While line-erase is active, a ▪ is appended to the key (DOT-▪).
LINE	Draws a line between the mark and the cursor, and moves the mark to the cursor.
TLINE	(Toggle line.) Toggles pixels on and off on the line between the mark and cursor. Does not move the mark to the cursor.
BOX	Draws a rectangular box using the mark and cursor as opposite corners.
CIRCL	Draws circle centered at the mark with radius defined by the mark and cursor.
MARK	Sets the mark. If no mark exists, the mark is created at the cursor. If the mark exists at another location, MARK moves the mark to the cursor location. If the mark exists at the cursor location, MARK erases the mark. (All operations that require a mark create a mark at the cursor location if no mark exists.)
DEL	Erases the rectangular region whose opposite corners are defined by the mark and cursor.

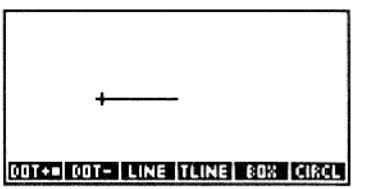
Adding Graphical Elements in the Graph. Env. (cont.)

[→][CLR]	Clears *PICT*.
[×]	Marks the display (same as MARK).
[DEL]	Same as DEL.

Example: Adding Graphical Elements to PICT. Add graphical elements to *PICT* as follows:

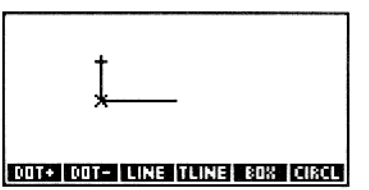
Select the PLOTR menu and erase *PICT*, then select the Graphics environment and use DOT+ to draw a horizontal line from the center of *PICT* halfway to the left edge of *PICT*.

[→][PLOT] ERASE
[◀] [NXT] DOT+
hold down [◀]



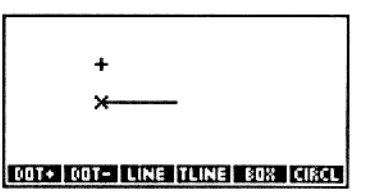
Turn off DOT+■. Then use TLINE to draw a vertical line from the current cursor position halfway to the top edge of *PICT*.

DOT+■
TLINE
hold down [▲]
TLINE



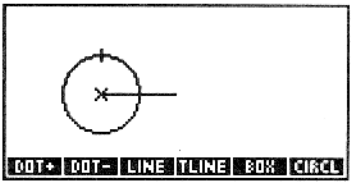
Toggle the line off.

TLINE



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

CIRCL



Programmable Commands for Adding Graphical Elements to PICT

The following commands in the PRG DSPL menu take coordinate arguments in either user-unit or pixel form.

Commands for Adding Graphical Elements to PICT

Keys	Programmable Command	Description
PRG DSPL (pages 1 and 2):		
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 the 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.
PIXON	PIXON	Turns on the specified pixel in *PICT*.

Commands for Adding Graphical Elements to PICT (cont.)

Keys	Programmable Command	Description
PIXOF	PIXOFF	Turns off the specified pixel in *PICT*.
PIX?	PIX?	Returns 1 if the specified pixel is on; returns 0 if the specified pixel is off.
PX→C	PX→C	Converts a pixel coordinate { #*n* #*m* } to user unit coordinate ⟨*x*,*y*⟩.
C→PX	C→PX	Converts a user unit coordinate ⟨*x*,*y*⟩ to pixel coordinate { #*n* #*m* }.

Working with Graphics Objects on the Stack

Like other object types, graphics objects can be put on the stack and stored into variables. On the stack, a graphics object is displayed as:

Graphic *n* × *m*

where *n* and *m* are the width and height in pixels.

(When a graphics object from the stack is placed in the command line, it is displayed as:

GROB *n m h*

where *n* and *m* are the width and height in pixels, and *h* is the pixel pattern represented as hexadecimal digits (0—9 and A—F).)

Stack-Related Operations in the Graphics Environment

The following operations in the Graphics environment use a graphics object from the stack or return a graphics object to the stack.

Stack-Related Operations in the Graphics Environment

REPL	Superimposes the graphics object from level 1 on *PICT*. The upper left corner of the graphics object is positioned at the cursor.
SUB	Puts on the stack the rectangular graphics object whose opposite corners are defined by the mark and cursor.
STO	Copies *PICT* to the stack as a graphics object.

The PICT Command—Working with PICT on the Stack

The PICT command (PRG DSPL PICT) puts the name PICT on the stack. The name can be used as an argument to permit access to the *PICT* graphics object as if it were stored in a variable:

- Press PICT → RCL to recall the *PICT* graphics object to the stack.
- With a graphics object in level 1, press PICT STO to make that graphics object the *PICT* graphics object.
- Press PICT ← PURGE to purge *PICT*.

The name PICT can be used as an argument to several graphics objects commands described in the next section. For example, the SUB command accepts *PICT* as an argument, letting you define a region of *PICT* to return to the stack as a graphics object. This is the stack related equivalent of the SUB operation in the Graphics environment described in the previous section.

Stack Commands for Graphics Objects

The PRG DSPL menu contains programmable commands for creating graphics objects on the stack and controlling the display. These commands are principally useful in programs. Commands that take coordinate arguments can use pixel coordinates (in list form ‹ #*n* #*m* ›) or user-unit coordinates (in complex-number form ‹*x*,*y*›).

Stack Commands for Graphics Objects

Keys	Programmable Command	Description
PRG DSPL		
PVIEW	PVIEW	(*PICT* view.) Displays *PICT* with the specified coordinate at the upper left corner of the graphics display. If the argument is an empty list, displays *PICT* centered in the display, with scrolling mode activated.
SIZE	SIZE	Returns the width and height in pixels of the level 1 graphics object.
→GRO	→GROB	(To graphics object.) Converts an *object* into a graphics object. Takes the object from level 2 and *n* from level 1, where *n* is a real number from 0 to 3 specifying the character size. Character 0 and character 3 are the same, except for algebraic and unit objects, where character 0 specifies that the resultant graphics object is the EquationWriter application picture. If *n* is 1 through 3, the resultant graphics object is a string in small (*n*=1), medium (*n*=2), or large (*n*=3) character.

Stack Commands for Graphics Objects (continued)

Keys	Programmable Command	Description
BLAN	BLANK	Creates a blank graphics object on the stack of size #*n* (in level 2) by #*m* (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 dark on a light background and light on a dark background.
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.
SUB	SUB	(Subset.) Extracts a portion of a graphics object and returns it to the stack. It takes three arguments—a graphics object (level 3) and coordinates (levels 2 and 1) that define the diagonal corners of the rectangle to be extracted.
→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.

Stack Commands for Graphics Objects (continued)

Keys	Programmable Command	Description
LCD→	LCD→	(LCD to stack.) Returns a graphics object to level 1 representing the current stack display.
FREEZ	FREEZE	"Freezes" one or more of three display areas so that they are not updated until a key press. (See page 523 in chapter 29.) Used with PVIEW in a program so that *PICT* persists in the stack display until a key press.
TEXT	TEXT	Displays the stack display.

Reference Programs. Programs *PIE* and *WALK* in chapter 31 use commands discussed in the previous sections. *PIE* uses ARC and LINE to draw a pie chart. It then recalls *PICT* to the stack and executes GOR to merge a label with each slice of the pie chart.

WALK uses a custom graphical image in a program, executing GXOR in a loop structure to animate the image.

20

Arrays



The HP 48 has extensive capabilities for entering and manipulating arrays. *Array* objects represent both vectors and matrices. A *vector* is a one-dimensional array; a *matrix* is a two-dimensional array.

This chapter covers these topics:

- Using the MatrixWriter application to enter and edit arrays.
- Using the command line to enter arrays.
- Doing arithmetic operations with arrays.
- Working with complex-number arrays.

Two-element and three-element vectors are particularly useful in engineering. They are covered in chapter 12, "Vectors."

Entering Arrays

Here is an example of a 3 × 3 matrix as it appears on the stack.

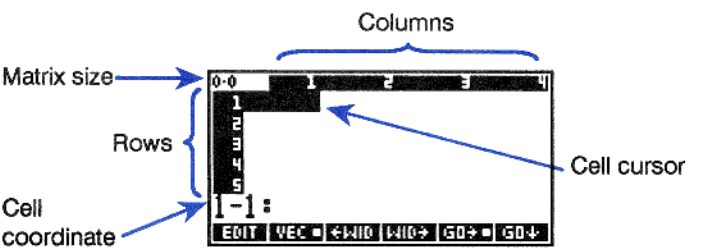
```
[[ 1 2 3 ]
 [ 3 4 5 ]
 [ 7 8 9 ]]
```

Square brackets ([ ]) enclose the matrix and also enclose each row. You can key an array directly into the command line using the square-bracket delimiters (see "Entering Arrays Using the Command Line" on page 349). However, the MatrixWriter application provides an easier way to enter arrays.

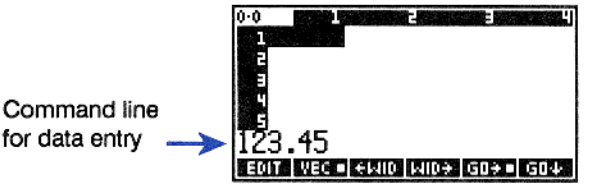
The MatrixWriter Application

To enter a matrix using the MatrixWriter application:

1. Press [→] [MATRIX] to display the MatrixWriter screen and menu.



2. Enter the first row:
   - Key in the value of element 1-1. During digit entry, the cursor coordinate is replaced by the command line.



     Press [ENTER]. The value is placed in the cell, and the cell cursor advances to the next cell in the row. If the number is wider than the cell width, an ellipsis (...) indicates "more to the right." The default cell width is 4 characters. Use ←WID or WID→ to make the cells narrower or wider.
   - You can key in more than one element at a time. To do so, separate elements with spaces, and enter them with [ENTER].
   - You can use the command line to compute elements as you enter them. For instance, the keystrokes 60 [SPC] 10 [÷] [ENTER] would enter 6 into the matrix.
   - To end the first row, press [▼] after entering the last element. This sets the number of columns in the matrix and moves the cursor to the beginning of the next row.
3. Enter the data for the rest of the matrix. There is no need to press [▼] again; the cell cursor automatically wraps to each new row.
4. When all the data has been entered, press [ENTER] to enter the matrix onto the stack. (Note the two uses of [ENTER]: During data entry, [ENTER] enters data into a cell; when a cell coordinate is present, [ENTER] enters the entire matrix onto the stack.)

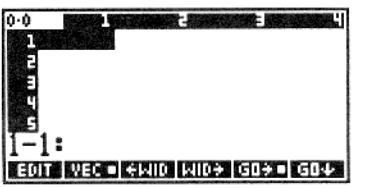
To enter a vector using the MatrixWriter application, you follow the instructions above until you've completed the first (and only) row of data. Then simply press [ENTER] again to enter the vector onto the stack.

Example: Using the MatrixWriter Application. Enter the matrix:

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

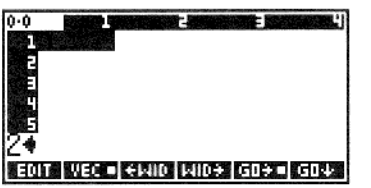
Select the MatrixWriter application.

[→] [MATRIX]



Key in the first element (element 1-1).

2



Enter the value into the cell.

[ENTER]



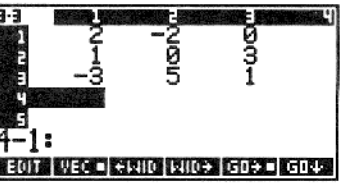
Enter the rest of the first row.

2 [+/−] [SPC] 0 [ENTER]



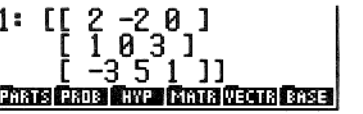
Use ▼ to end the first row. Then, enter rest of the matrix.

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



Enter the matrix onto the stack. (This matrix is used in a later example.)

ENTER



Entering Arrays Using the Command Line

To enter a vector using the command line:

1. Type the delimiters for the vector by pressing ↰[ ].
2. Key in the vector elements, separating each element by a space.
3. Press ENTER.

To enter a matrix using the command line:

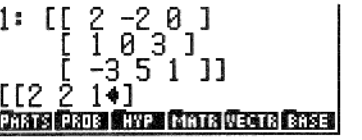
1. Type the delimiters for the matrix and first row by pressing ↰[ ] twice.
2. Key in the first row. When you're finished, use ▶ to move the cursor past the closing row delimiter.
3. Optional: use ↱↵ (carriage return) to start a new row in the display.
4. Key in the rest of the matrix. You do not need to delimit subsequent rows; the delimiters are added automatically when you press ENTER.

Example: Entering a Matrix Using the Command Line. Enter the following matrix:

$$\begin{bmatrix} 2 & 2 & 1 \\ 1 & 0 & 4 \\ 3 & 5 & 2 \end{bmatrix}$$

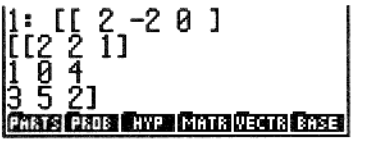
Key in the delimiters and the first row.

◀ [ ] ◀ [ ] 2 SPC 2 SPC 1



Move the cursor past the first] and key in the remaining values.

▶ ▶ ↵ 1 SPC 0 SPC 4
▶ ↵ 3 SPC 5 SPC 2



Enter the matrix onto the stack.

ENTER



How Vectors Are Displayed

Two- and three-element (two- and three-dimensional) vectors are displayed according to the current coordinate mode (Rectangular or Polar) and the current angle mode (Degrees, Radians, or Grads). For information on what these represent, see "Displaying 2D and 3D Vectors" on page 170.

Editing Arrays

To edit a matrix currently in level 1 of the stack, press ▼ when no command line is present. The matrix is displayed in the MatrixWriter environment. (If you want to edit a matrix in the command line instead of in the MatrixWriter environment, press ◀ EDIT to copy the matrix into the command line and display the EDIT menu.)

Within the MatrixWriter application, the cursor keys (◀, ▶, ▲, and ▼) move the cell cursor from cell to cell, and the right-shifted cursor keys (▶ ◀, ▶ ▶, etc.) move the cell cursor to the far left, far right, etc. Operations in the MATRIX menu let you edit the matrix.

The MatrixWriter Operations

Key	Description
EDIT	Places contents of the current cell in the data entry line for editing.
VEC	For one-row arrays, toggles between vector entry and matrix entry. If this key is "on" (a box is displayed in the key), one-row arrays are entered into the command line as vectors (example: [1 2 3]); if it is "off" (no box in the key), 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.

If both GO→ and GO↓ are off (no box in either menu label), the cursor does not advance after an entry is made.

To add a column to the right of the last column, move the cursor to that column and enter a value. The rest of the column is filled with zeros. Use a similar procedure to add a row to the bottom.

Example: Editing a Matrix. Change the matrix entered on page 348 from:

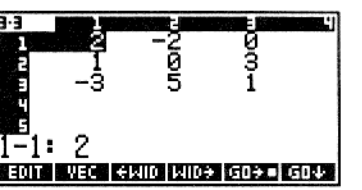
$$\begin{bmatrix} 2 & -2 & 0 \\ 1 & 0 & 3 \\ -3 & 5 & 1 \end{bmatrix}$$

to:

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

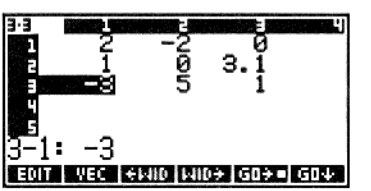
Place the matrix in the MatrixWriter environment. These keystrokes assume that the previous matrix is in level 1 and that GO→ is on (GO→▪).

▼



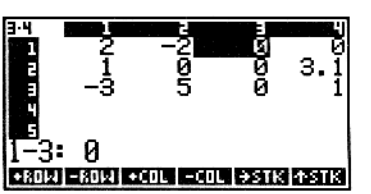
Edit element 2-3:

▼ ▶ ▶
EDIT ▶ .1 ENTER



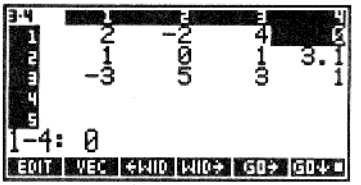
Insert a new column between columns 2 and 3, and move the cell pointer to the top of the new column.

▶ ▶ NXT +COL ▲ ▲



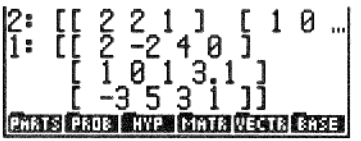
Set top-to-bottom entry mode. Fill in the new column.

NXT GD↓
4 SPC 1 SPC 3 ENTER



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

GD→ ENTER



Arithmetic Operations with Arrays

Doing Arithmetic with Vectors

Addition and Subtraction. The vectors must have the same number of elements. If either vector contains complex elements, the resulting vector is complex.

Multiplication and Division. You can multiply or divide a vector by a real or complex number.

Dot Product, Cross Product, and Length. DOT returns the dot product of two vectors; CROSS returns the cross product. ABS, when applied to a vector, returns its length or magnitude.

For examples of using DOT, CROSS, and ABS with vectors, see "2D and 3D Vector Calculations" on page 176.

Doing Arithmetic with Matrices

Calculating the Reciprocal of a Matrix. The INV command ([1/x]) calculates the reciprocal of a square matrix.

Example. Calculate the reciprocal of the following matrix:

$$\begin{bmatrix} 1 & 2 \\ 1 & 4 \end{bmatrix}$$

Enter the matrix (these keystrokes use the command line).

[↰][ ] [↰][ ] 1 [SPC] 2
[▶] 1 [SPC] 4 [ENTER]



Calculate the reciprocal.

[1/x]



Adding and Subtracting Matrices. Use the [+] and [−] keys to add or subtract matrices in levels 2 and 1. The matrices must have the same dimensions.

Multiplying or Dividing a Matrix by a Number. The result is obtained by multiplying or dividing each element in the array by a real or complex number. For division, the scalar must be in level 1.

Multiplying Matrices. The product is the matrix product of the two arrays. The number of columns in the matrix in level 2 must equal the number of rows in the matrix in level 1.

Example. Calculate the matrix product:

$$\begin{bmatrix} 2 & 2 \\ 4 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 2 & 2 & 1 & 4 \\ 3 & 4 & 2 & 1 \end{bmatrix}$$

Enter the first matrix.

[→][MATRIX]
2 [SPC] 2 [ENTER] [▼]
4 [SPC] 1 [SPC] 2 [SPC] 3 [ENTER]
[ENTER]



Enter the second matrix.

[→][MATRIX]
2 [SPC] 2 [SPC] 1 [SPC] 4 [ENTER] [▼]
3 [SPC] 4 [SPC] 2 [SPC] 1 [ENTER]
[ENTER]



Multiply the matrices.

[×]



Doing Arithmetic with a Matrix and a Vector

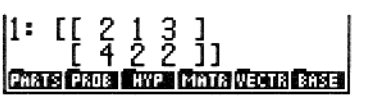
Multiplying a Matrix and a Vector. The matrix must be in level 2. The number of elements in the vector must equal the number of columns in the matrix. (The vector is treated as a column vector.)

Example. Calculate the following product:

$$\begin{bmatrix} 2 & 1 & 3 \\ 4 & 2 & 2 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \end{bmatrix}$$

Enter the matrix.

[→][MATRIX]
2 [SPC] 1 [SPC] 3 [ENTER] [▼]
4 [SPC] 2 [SPC] 2 [ENTER]
[ENTER]



Key in the vector and do the multiplication.

[←][ ] 3 [SPC] 1 [SPC] 1 [×]



Dividing a Vector by a Matrix. The vector must be in level 2. It must have the same number of elements as the number of columns of the square matrix. Vector/matrix division is used to solve a system of linear equations.

Solving a System of Linear Equations

To solve a system of n linear equations with n variables, divide the n-element constant vector by the $n \times n$ *coefficient* matrix.

Example. Solve the following system of three linear, independent equations with three variables:

$$3x + y + 2z = 13$$

$$x + y - 8z = -1$$

$$-x + 2y + 5z = 13$$

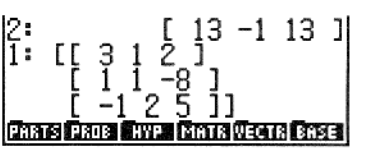
Enter the constant vector.

[→] [MATRIX] 13 [SPC] 1 [+/−] [SPC] 13 [ENTER] [ENTER]



Enter the coefficient matrix.

[→] [MATRIX]
3 [SPC] 1 [SPC] 2 [ENTER] [▼]
1 [SPC] 1 [SPC] 8 [+/−] [SPC]
1 [+/−] [SPC] 2 [SPC] 5 [ENTER]
[ENTER]



Divide the vector by the matrix.

[÷]



The values that satisfy the equations are: $x = 2$, $y = 5$, and $z = 1$.

Complex Arrays

Arrays can contain only real or complex numbers; no other object types are allowed. A *complex array* is a vector or matrix that contains one or more complex-number elements.

Arithmetic with Complex Arrays

If either argument is a complex array, the result is a complex array. For example, if you add a real matrix and a complex matrix, the result is a complex matrix.

Additional Complex Array Commands

With the exception of the coordinate-mode-dependent commands (V→, →V2, and →V3), all the commands that manipulate real arrays can be used with complex arrays. In addition, the following commands are used with complex arrays.

Commands for Manipulating Complex Arrays

Keys	Programmable Command	Description
[+/-]	NEG	Returns an array in which each element is the negative of the argument array.
[PRG] OBJ (page 2):		
R→C	R→C	Combines two arrays into a complex array. The array in level 2 becomes the real part; the array in level 1 becomes the imaginary part.

Commands for Manipulating Complex Arrays (continued)

Keys	Programmable Command	Description
C→R	C→R	Returns arrays containing the real and imaginary parts of a complex array to levels 2 and 1.
MTH PARTS:		
CONJ	CONJ	Returns the complex conjugate of a complex array.
RE	RE	Returns a real array consisting of the real parts of a complex array.
IM	IM	Returns a real array consisting of the imaginary parts of a complex array.

Example: Calculating a Conjugate. Calculate the conjugate CONJ(A) of matrix A:

$$A = \begin{bmatrix} 1+3i & i \\ 3 & 2-4i \end{bmatrix}$$

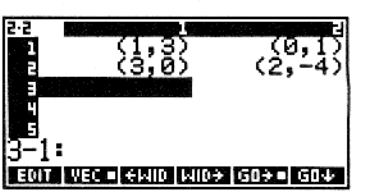
Select the MatrixWriter application and enter the complex numbers.

MATRIX
( ) 1 SPC 3 ENTER
( ) 0 SPC 1 ENTER ▼
3 ENTER
( ) 2 SPC 4 +/- ENTER



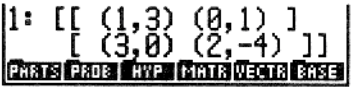
Widen the columns to see the full entry.

WID→ WID→



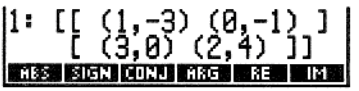
Enter the matrix onto the stack.

ENTER



Compute the conjugate.

MTH PARTS CONJ



Additional Matrix Commands

The following commands are found in the MTH MATR menu (MTH MATR).

Command/Description	Example	
	Input	Output
ABS Frobenius (Euclidean) norm; square root of the sums of the squares of the absolute values of the elements.	1: [[2 2] [2 2]]	1: 4
CNRM Column norm; maximum value (over all columns) of the sums of the absolute values of all elements in a column.	1: [[1 2 3] [4 5 6] [7 8 9]]	1: 18
CON Constant; returns a constant real or complex array according to the dimensions specified by a list {*n*} or {*n m*}.	2: { 2 3 } 1: 7 or 2: [[1 2 3] [4 5 6]] 1: 7	1: [[7 7 7] [7 7 7]]

Command/Description	Example	
	Input	Output
DET Determinant; returns the determinant of a square matrix.	`1:  [[1 2]` `  [3 4]]`	`1:  -2`
IDN Identity; returns an $n \times n$ (in level 1) identity matrix, or replaces the elements of the matrix in level 1.	`1:  2` or `1:  [[1 2]` `  [3 4]]`	`1:  [[1 0]` `  [0 1]]`
RDM Redimension; Redimensions an array. The new dimensions are in a list in level 1. Elements preserve the order of the source array.	`2: [[1 2 3]` `  [4 5 6]` `  [7 8 9]]` `1:  ⟨ 3 2 ⟩`	`1:  [[1 2]` `  [3 4]` `  [5 6]]`
RNRM Row norm; maximum value (over all rows) of the sums of the absolute values of all elements in a row.	`1: [[1 2 3]` `  [4 5 6]` `  [7 8 9]]`	`1:  24`
TRN Transpose; transposition of the argument; an $n \times m$ matrix is replaced by an $m \times n$ matrix. (Complex entries are conjugated.)	`1: [[1 2 3]` `  [4 5 6]]`	`1:  [[1 4]` `  [2 5]` `  [3 6]]`

CON, IDN, RDM, and TRN allow name arguments in place of the array argument. For example, evaluating the sequence `'A1' 7 CON` replaces the array stored in *A1* with a constant array of the same dimensions.

Additional commands for manipulating matrices (GET, GETI, OBJ→, PUT, and PUTI) are covered in the table starting on page 90.

Advanced Topics Relating to Matrices

Improving the Accuracy of System Solutions (the RSD Command). Because of rounding errors during calculation, a numerically calculated solution **Z** is not in general the solution to the original system $\mathbf{AX} = \mathbf{B}$, but rather the solution to the perturbed system $(\mathbf{A} + \Delta\mathbf{A})\,\mathbf{Z} = \mathbf{B} + \Delta\mathbf{B}$.

The perturbations $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ satisfy $\Delta\mathbf{A} \leq \varepsilon\mathbf{A}$ and $\Delta\mathbf{B} \leq \varepsilon\mathbf{B}$, where ε is a small number and **A** is the *norm* of **A**, a measure of its size analogous to the length of a vector. In many cases $\Delta\mathbf{A}$ and $\Delta\mathbf{B}$ will amount to less than one in the 12th digit of each element of **A** and **B**.

For a calculated solution **Z**, the *residual* is $\mathbf{R} = \mathbf{B} - \mathbf{AZ}$. Then $\mathbf{R} \leq \varepsilon\mathbf{AZ}$. So the expected residual for a calculated solution is small. Nevertheless, the *error* $\mathbf{Z} - \mathbf{X}$ may not be small if **A** is ill-conditioned, that is, if $\mathbf{Z} - \mathbf{X} \leq \varepsilon\mathbf{A}\,\mathbf{A}^{-1}\,\mathbf{Z}$.

For the HP 48, which carries 15 accurate digits, the number of correct digits is greater than or equal to $11 - \log(\mathbf{A}-1) - \log n$. In many applications, this accuracy may be adequate. When additional accuracy is desired, the computed solution **Z** can usually be improved by *iterative refinement* (also known as *residual* corrections). Iterative refinement involves calculating a solution to a system of equations, then improving its accuracy using the residual associated with the solution to modify that solution.

To use iterative refinement, first calculate a solution **Z** to the original system $\mathbf{AX} = \mathbf{B}$.

Then **Z** is treated as an approximation to **X**, in error by $\mathbf{E} = \mathbf{X} - \mathbf{Z}$.

E now satisfies the linear system $\mathbf{AE} = \mathbf{AX} - \mathbf{AZ} = \mathbf{R}$, where **R** is the residual for **Z**.

The next step is to calculate the residual and then solve **AE** = **R** for **E**. The calculated solution, denoted by **F**, is treated as an approximation to **E** and is added to **Z** to obtain a new approximation to **X**.

For **F** + **Z** to be a better approximation to **X** than is **Z**, the residual

$$\mathbf{R} = \mathbf{B} - \mathbf{AZ}$$

must be calculated to extended precision. The function RSD does this.

The refinement process can be repeated, but most of the improvement occurs in the first refinement. The / (divide) function does not attempt to perform a residual refinement because of the memory required to maintain multiple copies of the original arrays.

Here is an example of a user program that solves a matrix equation, including one refinement using RSD:

```
« → B A « B A / B A 3 PICK RSD A / + » »
```

This program takes two array arguments **B** and **A** from the stack, (the same as /) and returns the result array **Z**, which will be an improved approximation to the solution **X** over that provided by / itself.

Over-Determined and Under-Determined Systems. An under-determined system of linear equations contains more variables than equations, and the coefficient array has fewer rows than columns. The following program solves an under-determined system **AX** = **B** using the Moore-Penrose technique ($\mathbf{X} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{B}$). The program requires as input the vector **B** in level 2 and the matrix **A** in level 1.

```
« → B A
  « A TRN
    B A A TRN * / *
  »
»
```

An over-determined system contains fewer variables than equations. The next program solves an over-determined system using the least squares method ($\mathbf{X} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}$). Like the previous program, its input is **B** in level 2 and **A** in level 1.

```
« → B A
  « A TRN B *
    A TRN A * /
  »
»
```

21

Statistics



The Statistics application enables you to calculate single-sample and paired-sample statistics, including:

- Total, mean, maximum, and minimum.
- Sample standard deviation and covariance.
- Correlation coefficient.
- Curve-fitting with four models — linear, logarithmic, exponential, and power.
- Summary statistics.
- Upper-tail probabilities for various test statistics.

It also enables you to draw scatter plots, bar charts, and frequency histograms.

A Statistics Example. The following table lists the consumer price index (CPI), producer price index (PPI), and unemployment rate (UR) for the United States over a 5-year period. Enter the data and then do the following:

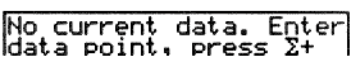
- Calculate the mean, standard deviation, and total of the CPI, PPI, and UR data.
- Calculate the correlation and covariance of CPI and PPI.

- Draw a scatter plot of the data.
- Calculate a predicted value of PPI from a given value of CPI, using a linear model.

Year	CPI	PPI	UR
1	9.1	9.2	8.5
2	5.8	4.6	7.7
3	6.5	6.1	7.0
4	7.6	7.8	6.0
5	11.5	19.3	5.8

Set 2 FIX display mode, enter the Statistics application, and clear any previous statistical data.

◀ MODES 2 FIX
◀ STAT CLΣ



Note

In this example you use the Σ+ key to enter the data. You can also use the MatrixWriter application to enter statistical data. Instructions on how to do this are found in "Entering New Statistical Data" on page 369.

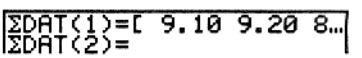
Key in the data for year 1. Since statistical data is stored in a matrix, you must use square brackets to identify the values as one row of the matrix.

◀ [ ] 9.1 SPC 9.2 SPC 8.5



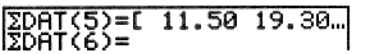
Enter the data into the statistical matrix.

Σ+



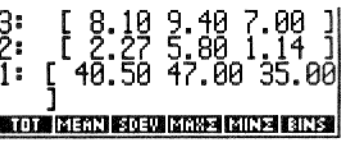
Enter the rest of the data. Once you've entered the first row, the number of columns in the matrix is set, so you no longer need to use square brackets.

5.8 SPC 4.6 SPC 7.7 Σ+
6.5 SPC 6.1 SPC 7 Σ+
7.6 SPC 7.8 SPC 6 Σ+
11.5 SPC 19.3 SPC 5.8 Σ+



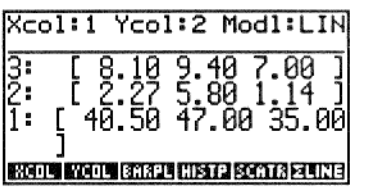
Calculate the mean, standard deviation, and total of the columns.

NXT MEAN
SDEV
TOT



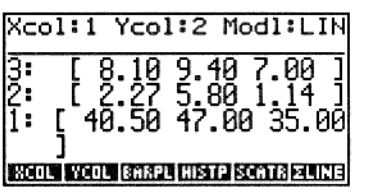
To do paired-sample statistics, switch to the next page (page 3) of the STAT menu. Note the message at the top of the display; if necessary, set columns 1 and 2 as the *x*- and *y*-variables.

NXT
If necessary:
1 XCOL 2 YCOL



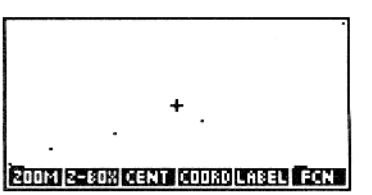
Switch to the next page (page 4) of the menu and make sure the current model is linear (LIN). Then return to page 3 of STAT.

NXT MODL LIN
↰ PREV



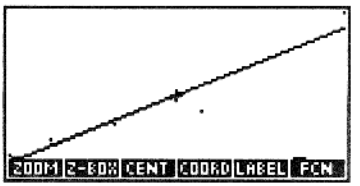
Plot a scatter plot of the data.

SCATR



Draw the best straight line for the data.

FCN



Calculate the linear regression statistics; then calculate the correlation coefficient and covariance.

ATTN NXT
LR
CORR
COV



Now that the linear regression statistics have been calculated, calculate a predicted value for PPI (y) given a CPI of 8.5.

8.5 PREDY



Change the display mode from 2 Fix back to Standard by pressing ↰ MODES STD.

Starting the Statistics Application

Press ↰ STAT to display the first page of the STAT menu. If there is any *current statistical data*, a message in the display shows the last values entered.

The first page of the STAT menu contains keys for entering and manipulating data. The other pages contain commands for doing calculations and drawing graphs.

STAT Commands for Entering and Manipulating Data

Keys	Programmable Command	Description
◀ STAT:		
Σ+	Σ+	Enters data from the stack into the current statistical matrix.
◀ Σ+	Σ−	Deletes the last data point from the statistical matrix and returns it to the stack.
CLΣ	CLΣ	Clears the current statistical matrix.
NEW		Takes a matrix from level 1, prompts for a variable name, stores the matrix in that variable, and makes that matrix the current statistical matrix.
EDITΣ		Places the current statistical matrix in the MatrixWriter environment for editing. Press ENTER when finished editing, or ATTN to cancel the edit without any changes.
STOΣ	STOΣ	Stores the matrix in level 1 as the current statistical matrix.
▶ STOΣ	RCLΣ	Recalls the current statistical matrix to level 1.
CAT		Displays the catalog of matrices and subdirectories in the current directory.

When you execute these operations, the status message relating to the last data entered is erased. You can press ◀ REVIEW to redisplay it.

Designating the Current Statistical Matrix

Statistical data is stored in the form of a matrix. The matrix contains a row for each data point and a column for each variable.

	$\mathbf{var}_1$	$\mathbf{var}_2$	...	$\mathbf{var}_m$
$\mathbf{point}_1$	x_{11}	x_{12}	...	x_{1m}
$\mathbf{point}_2$	x_{21}	x_{22}	...	x_{2m}
⋮	⋮	⋮	⋮	x_{2m}
$\mathbf{point}_n$	x_{n1}	x_{n2}	...	x_{nm}

The *current statistical matrix* is the data used by the STAT commands. It is designated by the contents of a reserved variable named Σ*DAT*. Σ*DAT* can contain either the matrix itself or the name of a variable containing the matrix. Since Σ*DAT* is a variable, you can have a different current statistical matrix for each directory in memory.

If there is no current matrix, or if you want to use data different from that in the current matrix, you can designate a new current statistical matrix by entering new data, editing the current data, or selecting another matrix. A list of other matrices is found in the STAT Catalog.

Entering New Statistical Data (Σ+ and NEW)

There are two ways to enter new statistical data:

- You can use Σ+ to enter the data one point at a time.
- You can create the entire matrix and then store it into Σ*DAT* using NEW. The easiest way to create a matrix involves the MatrixWriter application.

To enter data using the Σ+ command:

1. Press CLΣ to clear Σ*DAT*.
2. Key in the data for the first data point. If the point has more than one value, enter the values as a vector (delimited by square brackets). Press Σ+ to enter the data into Σ*DAT*.

3. To enter each additional point, key in the value(s) and press Σ+. After the first data point, you do not need to enclose values in brackets.

To enter data using the MatrixWriter application and the NEW command:

1. Press [→] [MATRIX] to select the MatrixWriter application. Use it to enter the data into a matrix. (Using the MatrixWriter application is covered in chapter 20.) Press [ENTER] to enter the completed matrix onto the stack.
2. Press NEW. This activates the alpha keyboard and displays a prompt for a variable name.
3. Key in a name for your matrix and press [ENTER]. The variable name is stored in Σ*DAT*, and the matrix itself is stored in the variable. (If you respond to the prompt by simply pressing [ENTER], the matrix itself is stored in Σ*DAT* and no variable name is created.)

Editing Data

Editing the Last Data Point. The Σ− command is useful for changing a data point you just entered into Σ*DAT* using Σ+. It is executed by pressing [←] Σ+. Σ− removes the last point in Σ*DAT* and places it in level 1. To change the point, edit or replace it and use Σ+ to return the corrected data to Σ*DAT*.

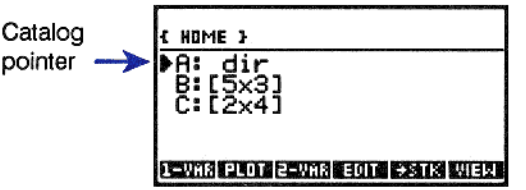
Editing Any Data Point. Press EDITΣ to copy the contents of Σ*DAT* into the MatrixWriter environment. After you edit the matrix, press [ENTER] to make the edited version the current statistical matrix.

Using the STAT Catalog

The STAT Catalog provides you with the ability to make any existing matrix the current matrix. Like other catalogs in the HP 48, it is a special environment where the keyboard is redefined and limited to specialized operations. You cannot go to another menu until you exit the catalog (covered under "Exiting the Catalog" on page 373).

Press ↰ STAT CAT to select the STAT Catalog. What you see is a catalog of:

- All the variables in the current directory that contain matrices.
- All the subdirectories in the current directory.



Move the pointer with ▲ and ▼ to select the desired entry. The entry you select can be manipulated using the STAT Catalog operations:

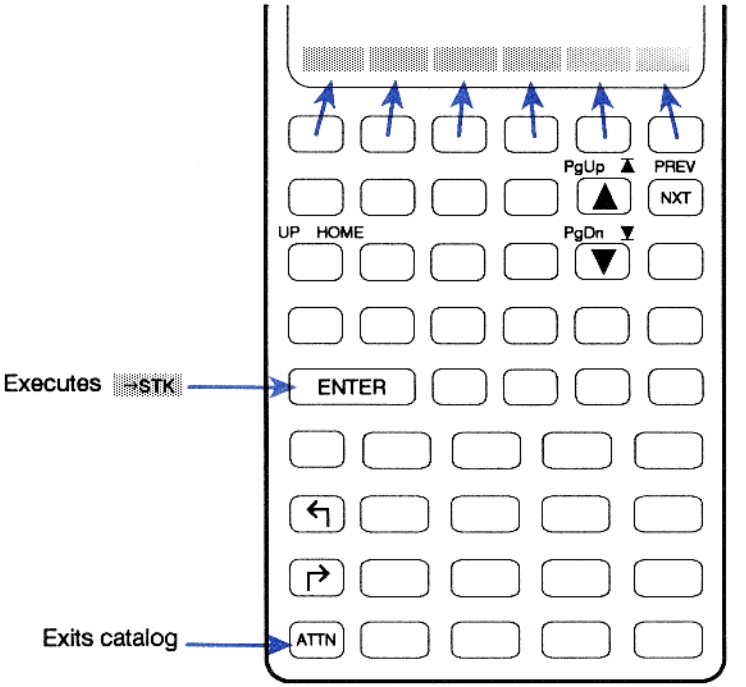
STAT Catalog Operations

1-VAR	Makes the selected entry the current statistical matrix, leaves the catalog, and displays the second page of the STAT menu (for calculating single-sample statistics).
PLOT	Makes the selected entry the current statistical matrix, leaves the catalog, and displays the third page of the STAT menu (for plotting data).
2-VAR	Makes the selected entry the current statistical matrix, leaves the catalog, and displays the fourth page of the STAT menu (for calculating paired-sample statistics).
EDIT	Places the selected entry in the MatrixWriter environment for editing. When you're finished editing, press ENTER to save the changes, or press ATTN to abort the edit without changing the matrix.
→STK	Copies the matrix to the stack.

STAT Catalog Operations (continued)

VIEW	Lets you view the contents of the entry. If the entry is a subdirectory, switches to that subdirectory.
ORDER	Moves the selected matrix to top of the catalog.
PURG	Purges the entry (and its corresponding variable).
NXT	Selects the next page of STAT Catalog operations.
◀ PREV	Selects the previous page of STAT Catalog operations.
▲	Moves the catalog pointer up one level. When prefixed with ◀, moves the catalog pointer up one page (◀ PgUp in the following keyboard illustration); when prefixed with ▶, moves the catalog pointer to the top of the catalog (▶ ▲ in the following keyboard illustration).
▼	Moves the catalog pointer down one level. When prefixed with ◀, moves the catalog pointer down one page (◀ PgDn in the following keyboard illustration); when prefixed with ▶, moves the catalog pointer to the bottom of the catalog (▶ ▼ in the following keyboard illustration).
ENTER	Executes →STK (copies matrix to stack). If the entry is a subdirectory, switches to that subdirectory, thus giving you access to any matrices there.
◀ UP	Switches to the parent directory.
▶ HOME	Switches to the *HOME* directory.
ATTN	Exits the catalog.

The redefined keyboard looks like this:



Exiting the Catalog

In general, press [ATTN] to exit the catalog and return to the STAT menu. Also, executing PLOT, 1-VAR, or 2-VAR automatically exits the catalog.

Calculating Single-Variable Statistics

Sample Statistics

The second page of the STAT menu contains commands for calculating single-sample statistics. Each command returns a vector containing m numbers, where m is the number of columns in the matrix. (If $m = 1$, where each data point consists of only one number, the commands return one number.) For example, if you have a 3-column matrix in ΣDAT and press MEAN, a 3-element vector containing the mean of each column is returned to level 1.

Single-Sample Statistics Commands

Keys	Programmable Command	Description
◀ STAT (page 2):		
TOT	TOT	Total.
MEAN	MEAN	Mean (average).
SDEV	SDEV	Sample standard deviation.
MAXΣ	MAXΣ	Maximum value.
MINΣ	MINΣ	Minimum value.
BINS	BINS	Calculates frequencies using the independent-variable column (XCOL) of ΣDAT. Takes as its arguments the minimum x-value (level 3), the width of each bin in user units (level 2), and the number of bins, n (level 1); returns an $n \times 1$ matrix of frequencies (level 2) and a two-element vector of frequencies outside the specified range (level 1).

When you execute these operations, the status message relating to the last data entered is erased. You can press ◀ REVIEW to redisplay it.

Population Statistics

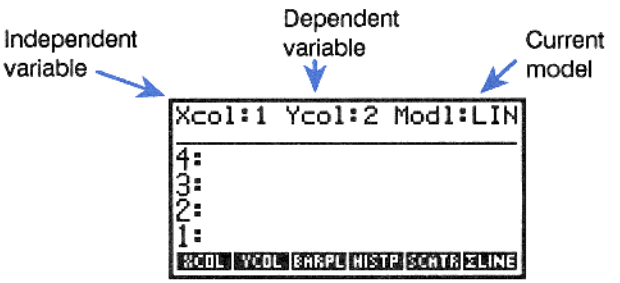
SDEV (standard deviation — page 2 of the STAT menu) and COV (covariance — page 4 of the STAT menu) calculate *sample statistics* for data that represents a sample of the population. If the contents of Σ*DAT* represent the entire population, you can calculate the *population statistics* using these steps:

1. Calculate the mean of the data (MEAN).
2. Execute Σ+ to append the mean data point to Σ*DAT*.
3. Then use SDEV and COV to calculate the population statistics.
4. Remove the mean data point from Σ*DAT* using Σ− (◀ Σ+).

Paired-Sample Statistics

The third and fourth pages of the STAT menu contain commands for computing paired-sample statistics.

When the third or fourth page of the STAT menu is displayed, the status message at the top of the display indicates the column designations for the independent (x) and dependent (y) variables and the current model.



Paired-Sample Statistics Commands

Keys	Programmable Command	Description
↰ STAT (pages 3 and 4):		
XCOL	XCOL	Takes a column number as its argument, and designates that column as the independent variable by storing it in the first position in Σ*PAR*. (↱ XCOL returns the XCOL column number to level 1.)
YCOL	YCOL	Takes a column number as its argument, and designates that column as the dependent variable by storing it in the second position in Σ*PAR*. (↱ YCOL returns the YCOL column number to level 1.)
ΣLINE	ΣLINE	Returns the expression representing the best fit line according to the current model.
LR	LR	Using the current model, computes the linear regression for the selected independent and dependent variables, and returns the intercept (level 2) and slope (level 1). Also, stores the intercept and slope values in Σ*PAR*, positions 3 and 4.
PREDX	PREDX	Takes as its argument a value for the dependent variable, and computes a predicted value for the independent variable. (LR must be executed at some point before PREDX.)

Paired-Sample Statistics Commands (continued)

Keys	Programmable Command	Description
PREDY	PREDY	Takes as its argument a value for the independent variable, and computes a predicted value for the dependent variable. (LR must have been executed at some point before PREDY.)
CORR	CORR	Correlation (computed according to the current model).
COV	COV	Sample covariance (computed according to the current model).
MODL		Displays the menu for selecting a model. Selection is stored in Σ*PAR*, position 5.

When you execute these operations, the status message (x, y, and model) is erased. You can press ↰ REVIEW to redisplay it.

To calculate paired-sample statistics:

1. Select the proper columns for your independent and dependent variables (XCOL , YCOL).
2. Select the desired model from the MODL menu (MODL). Do one of the following:
   - Select a particular model: LIN (linear), LOG (logarithmic), EXP (exponential), or PWR (power).
   - Compute the best model (BEST). The HP 48 selects the model for which the correlation has the largest absolute value. (If any data is negative or zero, LIN is selected.)
3. Optional: Press SCATR to draw a scatter plot of the data.
4. Use the fourth page of the STAT menu to calculate the paired-sample statistics. *You must calculate the linear-regression statistics* (LR) *before calculating predicted values* (PREDX or PREDY).

The example on page 364 demonstrates calculating paired-sample statistics.

The HP 48 uses the reserved variable Σ*PAR* when it calculates paired-sample statistics. Σ*PAR* contains a list of parameters, which can be changed by executing XCOL, YCOL, LR, and MODL, as described in the preceding table.

Plotting

The third page of the STAT menu contains commands for plotting single- and paired-sample statistics.

When the third page of the STAT menu is displayed, the status message at the top of the display indicates the column designations for the independent (*x*) and dependent (*y*) variables and the current model.

Plotting Commands

Keys	Programmable Command	Description
↰ STAT (page 3):		
XCOL	XCOL	Takes a column number as its argument, and designates that column as the independent variable.
YCOL	YCOL	Takes a column number as its argument, and designates that column as the dependent variable.
BARPL	BARPLOT	Draws a bar chart using the *x*-column. Autoscaled.
HISTP	HISTPLOT	Draws a frequency histogram using the *x*-column. Autoscaled.

Plotting Commands (continued)

Keys	Programmable Command	Description
SCATR	SCATRPLOT	Plots the (*x*,*y*) points using the designated *x*- and *y*-columns, and optionally draws the best line using the current model. Autoscaled.

When you execute some of these operations, the status message (*x*, *y*, and model) is erased. You can press [↰][REVIEW] to review the status information. Hold down the [REVIEW] key to prolong the status display.

Plotting Bar Charts

BARPLOT (BARPL) plots a bar chart of the specified column in Σ*DAT*. You specify the column using XCOL (page 3 of the STAT menu). If no column is specified, the first column in Σ*DAT* is used. Data can be positive or negative, resulting in bars above or below the *x*-axis.

Example. Records from a gas station show the following relationship between the monthly percentage changes in gasoline price and amount sold over a 4-month period:

Month	Price % Change	Sales % Change
1	+3.5	−1.2
2	+9.3	−2.6
3	−6.5	+6.1
4	+2	−0.4

Enter the data using the MatrixWriter application, and then plot bar charts for the percentage change in price and the percentage change in sales.

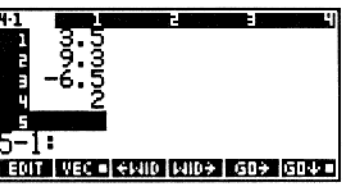
Start the MatrixWriter application.

[→] [MATRIX]



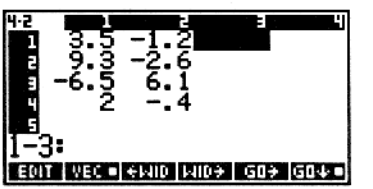
Enter the price data.

GO↓ 3.5 [SPC] 9.3 [SPC]
6.5 [+/-] [SPC] 2 [ENTER]



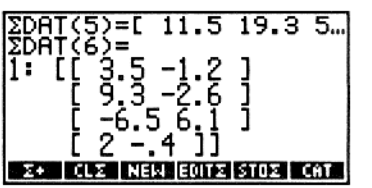
Enter the sales data.

[▶] 1.2 [+/-] [SPC] 2.6 [+/-] [SPC]
6.1 [SPC] .4 [+/-] [ENTER]



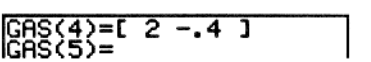
Enter the matrix onto the stack and start the Statistics application.

[ENTER] [←] [STAT]



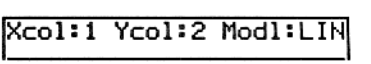
Name the matrix and make it the current data.

NEW GAS [ENTER]



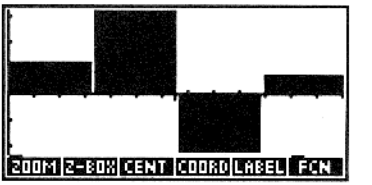
Select the column for percentage change in price (the first column in the statistical matrix).

[NXT] [NXT] 1 XCOL



Draw the bar chart for percentage change in price.

BARPL



Select the column for percentage change in sales (the second column in the matrix) and draw a bar chart for it.

ATTN 2 XCOL BARPL



When you plot statistical data, you actually move out of the Statistics application into the Graphics environment. Press ATTN to return to the Statistics application. (For more information on the Graphics environment, see page 301.)

Plotting Histograms

Once the statistical data is in Σ*DAT*, there are two approaches to plotting frequency histograms:

- HISTPLOT (HISTP on page 3 of the STAT menu) simply plots a histogram, showing the *relative* frequencies.
- BINS (BINS on page 2 of the STAT menu), in conjunction with BARPLOT, shows you *numerical* frequencies and then lets you plot the histogram. (BINS allows you more plotting control than HISTPLOT, but is a little more complicated to use.)

Plotting Histograms Using HISTPLOT. With the frequency data in Σ*DAT*, simply press HISTP to see your plot. The default number of bins is 13. (The RES command in the Plot application allows you to change the number of bins by changing their width — for more information, see page 321.)

Viewing Bins before They're Plotted. The BINS command allows you to see numerical frequencies before they're plotted. The command takes three arguments:

- In level 3, the minimum x-value to use (the lower bound of the range).
- In level 2, the width of each bin expressed as a positive real number.
- In level 1, the number of bins.

The output of BINS is:

- In level 2, an $n \times 1$ "bins" matrix, where n is the number of bins. The value of each element is the frequency of data in that bin.
- In level 1, a two-element "excess" vector containing the number of data points less than the minimum x-value, and the number of data points greater than the maximum x-value.

If you want to plot the histogram after viewing the bins:

1. Drop the vector in level 1 from the stack ([←][DROP]).
2. Save the one-dimensional matrix into Σ*DAT* by pressing STOΣ.
3. Press BARPL.

Summation Statistics

The fifth page of the STAT menu contains commands for calculating summation statistics. Use XCOL and YCOL (on the third page of the STAT menu) to designate *x* and *y*.

Summation Statistics Commands

Keys	Programmable Command	Description
↰ STAT (page 5):		
ΣX	ΣX	Returns the sum of the entries in the *x* (independent) column of Σ*DAT*.
ΣY	ΣY	Returns the sum of the entries in the *y* (dependent) column of Σ*DAT*.
ΣX^2	ΣX^2	Returns the sum of the square of the *x*-column entries of Σ*DAT*.
ΣY^2	ΣY^2	Returns the sum of the square of the *y*-column entries of Σ*DAT*.
ΣX*Y	ΣX*Y	Returns the sum of the products of corresponding *x* and *y* columns in Σ*DAT*.
NΣ	NΣ	Returns the number of rows in Σ*DAT*.

Test Statistics

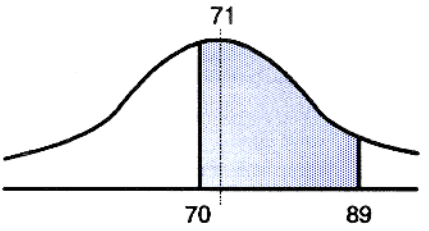
The PROB (probability) menu (MTH PROB) contains commands for calculating combinations, permutations, factorials, random numbers, and *upper-tail probabilities* of various test statistics. Upper-tail probabilities are covered here; the other topics are covered in chapter 9, "Common Math Functions."

Test Statistics Commands

Keys	Programmable Command	Description
MTH PROB (page 2):		
UTPC	UTPC	Upper-tail Chi square distribution: 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: 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: 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: 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*.

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

Example: Probabilities from a Normal Distribution. 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?



First, calculate the probability that a student chosen at random obtained a score greater than 70. (The standard deviation is squared to give the variance.)

MTH PROB NXT
71 ENTER
11 ↰ x^2
70 UTPN

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

Now, do the same calculation for a score of 89.

↱ LAST ARG ↰ DROP
89 UTPN

```
2:          .536217586697
1:          .050881752476
UTPC UTPF UTPN UTPT
```

Subtract the two values.

−

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

The calculation shows that 49% of the students scored between 70 and 89.

22

Algebra



The operations described in this chapter let you manipulate an algebraic expression or equation much as you would on a piece of paper. These operations let you:

- Solve symbolically for a variable:
  - ISOL isolates a single occurrence of a variable.
  - QUAD solves a quadratic equation for a variable.
- Collect and reorder variables and expand subexpressions:
  - The COLCT and EXPAN commands execute broad rearrangements by searching for a fixed set of patterns in an expression or equation, and applying an appropriate algebraic rule to every occurrence of each pattern.
  - The Rules operations let you perform step-by-step rearrangement of an expression or equation, tailoring the rearrangement to your need. The EquationWriter application serves as a platform from which to select the Rules operations.

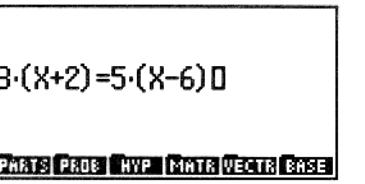
In this chapter, the term *algebraic* is used to mean algebraic expression or equation. The terms *expression* and *equation* are used only where the distinction between these two forms of an algebraic object is important.

Example: Solving for a Variable. Solve for x in the equation:

$$3(x + 2) = 5(x - 6)$$

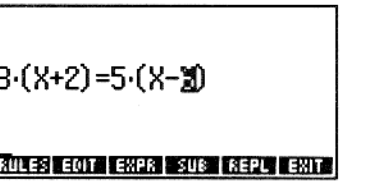
Select the EquationWriter application and key in the equation.

◀ EQUATION
3 ◀ ( ) X + 2 ▶
◀ =
5 ◀ ( ) X − 6 ▶



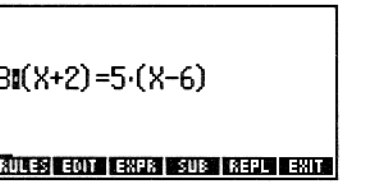
Activate the Selection menu and selection cursor.

◀



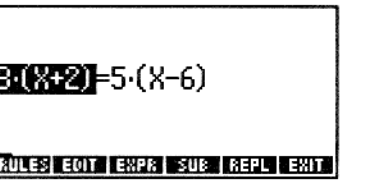
Move the selection cursor to the · sign on the left side of the equation.

↷ ◀ ▶



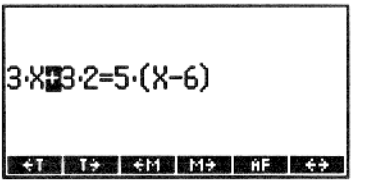
Highlight the subexpression defined by ·. This shows you what part of the equation will be affected by the subsequent execution of an algebraic rearrangement operation.

EXPR



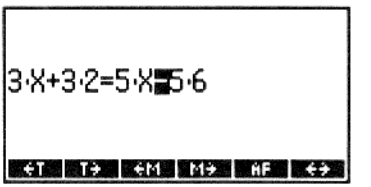
Select the RULES menu for this subexpression and distribute the 3 over (X+2).

RULES D→



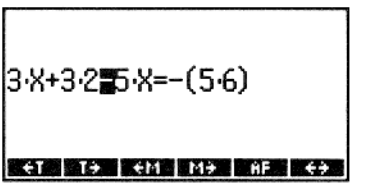
Now move the selection cursor to the · on the right side of the equation and distribute again.

▶ (6 times) RULES D→



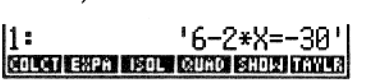
Move the cursor to the = sign and then move the term 5·X to the left side of the equation.

◀ (4 times) RULES ←T



Now that both terms in X are on the same side of the equation, return the equation to the stack, select the ALGEBRA menu, and collect like terms.

ENTER ↰ALGEBRA
COLCT



Now solve for X.

↑ X ISOL



Symbolic Solutions

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. The commands described in this section let you solve for a variable that appears once in an algebraic or for a variable in a quadratic equation.

Commands for Symbolic Solutions

Keys	Programmable Command	Description
↰ [ALGEBRA]:		
ISOL	ISOL	For an algebraic in level 2, isolates the first occurrence of the variable in level 1.
QUAD	QUAD	Solves the quadratic in level 2 for the specified variable in level 1.
SHOW	SHOW	Shows the algebraic in level 2 with all implicit references to the variable in level 1 made explicit.

Isolating a Variable

The ISOL command isolates a single occurrence of a variable in an algebraic—it returns an equation of the form:

'*variable*=*expression*'

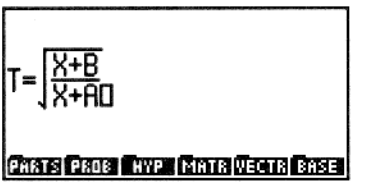
that represents a symbolic solution of the algebraic. To execute the ISOL command, place the algebraic in level 2 and the variable to be isolated in level 1.

Example: The ISOL Command. Use ISOL to isolate A in the equation:

$$T = \sqrt{\frac{X + B}{X + A}}$$

Key in the equation.

[↰] [EQUATION] T [↰] [=]
[√x] [▲] X [+] B [▶]
X [+] A



Enter the equation. Enter the name of the variable to be isolated and execute ISOL.

[ENTER]
['] A [ENTER]
[↰] [ALGEBRA] ISOL



Executing ISOL with Expressions. When level 2 contains an expression (an algebraic without an =), the expression is treated as an equation of the form '*expression*=0'. For example, isolating x in the expression:

```
'A*(B+2*X)-C'
```

returns:

```
'X=(C/A-B)/2'
```

Functions Not Allowed with ISOL. The variable to be isolated can be the argument of any function for which the HP 48 provides an inverse (defined in this manual as an *analytic* function). For example, you can isolate X in algebraics containing TAN(X) or LN(X) because TAN and LN have inverses (ATAN and EXP). However, you cannot isolate X in algebraics containing IP(X). The operations index identifies the HP 48 analytic functions.

Solving Quadratic Equations

The QUAD command solves any algebraic that is up to second order in the unknown variable. The command is named for its ability to solve second order (quadratic) algebraics, but you can also use QUAD to solve first order (linear) algebraics. (If you supply an equation that is *not* first or second order in the variable to be solved for, QUAD transforms the equation into a second order polynomial *approximation* and then solves that quadratic.)

To execute QUAD, place the algebraic in level 2 and the variable for which you are solving in level 1. Like ISOL, QUAD returns an equation of the form:

'*variable=expression*'

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. Otherwise, QUAD evaluates them.

Example 1: The QUAD Command. Solve for x in the expression:

$$x^2 - x - 6$$

This example assumes that variable X does not exist in the current directory.

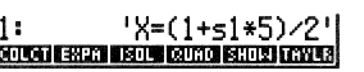
Enter the expression and the name of the variable.

' X y^x 2 − X − 6 ENTER
' X ENTER



Execute QUAD.

◀ ALGEBRA QUAD



QUAD returns an expression containing the variable *s1*, which represents an arbitrary + or − sign. (*s1* is discussed in detail in the following section.) Copy the expression. Then evaluate it for *s1* = 1. (To key in *s1*, press α ◀, then press SIN, and then press 1.)

ENTER
1 ' s1 STO
EVAL



Now evaluate the expression for $s1 = -1$

1 [+/-] ['] s1 [STO]
[↶] [SWAP]
[EVAL]



The two roots are $+3$ and -2.

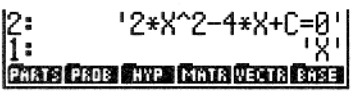
Example 2: Solving an Equation with More Than One Variable. Solve for x in the equation:

$$2x^2 - 4x + c = 0$$

This example assumes that variable X does not exist in the current directory.

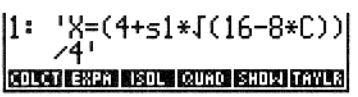
Purge C, if necessary. Then, enter the equation and the variable name X.

['] C [↶] [PURGE]
['] 2 [×] X [y^x] 2
[−] 4 [×] X [+] C [↶] [=] 0
[ENTER] ['] X [ENTER]



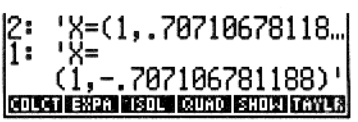
Execute QUAD to obtain a result containing C.

[↶] [ALGEBRA] QUAD



Copy the expression. Calculate the roots when $c = 3$.

[ENTER]
3 ['] C [STO]
1 ['] s1 [STO] [EVAL]
1 [+/-] ['] s1 [STO]
[↶] [SWAP] [EVAL]



The roots are $1 \pm 0.7071i$.

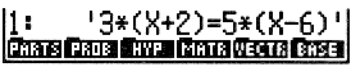
Example 3: Solving a First Order Equation. The introductory example in this chapter (page 387) used ISOL to solve for x in the equation:

$$3(x + 2) = 5(x - 6)$$

Use QUAD to solve for x.

Key in the equation.

[ ] 3 [×] [←] [( )] X [+] 2 [▶]
[←] [=] 5 [×] [←] [( )] X [−] 6
[ENTER]



Solve for x.

[ ] X QUAD



Using QUAD Or ISOL. Comparing the previous example with the introductory example in this chapter shows that using QUAD can be more efficient than ISOL for solving for a variable. The advantage of using QUAD is that you don't have to rearrange the algebraic to a form in which the unknown variable occurs only once. However, QUAD provides exact solutions only for first or second order polynomials — you must use ISOL to obtain an exact solution when:

- The unknown variable is third order or higher.
- The unknown variable occurs as an argument to a non-linear function like SIN.

General and Principal Solutions

HP 48 functions always return one result — a *principal* solution. For example, $\sqrt{4}$ always returns +2, and ASIN(.5) always returns 30° or 0.524 radians. Since you may want other results when you isolate a variable or solve a quadratic equation, the HP 48 returns a *general solution* for ISOL and QUAD. A general solution contains either or both of the following variables:

- *s1* represents an arbitrary + or − sign. You can evaluate the expression by storing either +1 or −1 into *s1*. Additional arbitrary signs in the result are indicated by *s2*, *s3*, etc.
- Variable *n1* represents an arbitrary integer — 0, 1, 2, etc. Additional arbitrary integers are represented by *n2*, *n3*, etc.

To specify that ISOL and QUAD return a principal solution, set flag −1. When you specify principal solutions, arbitrary signs are always chosen to be +1 and arbitrary integers are always chosen to be 0.

Example: General and Principal Solutions. Use ISOL to isolate x in the equation:

$$y = \sin x^2$$

First, enter the equation. Then copy it. Set Radians mode. Then, enter the variable to be isolated and execute ISOL.

' Y ◀ = SIN
X y^x 2 ENTER ENTER
◀ RAD (if necessary)
' X ENTER
◀ ALGEBRA ISOL



ISOL returns a general solution with arbitrary sign *s1* and arbitrary integer *n1*. Now return a principal solution for the same expression. First set flag −1. Then swap the copy of the original equation to level 1, key in the variable and execute ISOL.

1 +/- ▶ MODES NXT SF
◀ SWAP
' X ◀ ALGEBRA ISOL



ISOL returns a principal solution with arbitrary sign *s1* = +1 and arbitrary integer *n* = 0, so the expression evaluates to `√ASIN(Y)`.

Clear flag −1 so that the HP 48 returns general solutions.

Showing Hidden Variables

You may want to solve for a variable that is stored in a different variable. The SHOW command rewrites an algebraic to explicitly show every occurrence of a specified variable. For example, if `A` contains `'X+1'` then `'A*B' 'X' SHOW` returns `'(X+1)*B'`.

You can supply two or more names to SHOW in a list argument. In this case, SHOW evaluates all names in the level 2 expression that are *not* in the list so that the result expression contains only the listed names. This is a useful tool for speeding up plotting of an equation.

Rearranging Terms

The ISOL command lets you solve for a variable that appears just once in an algebraic. However, if the variable for which you want to solve appears more than once, you must first combine the multiple occurrences. In addition, it may be necessary, before collecting multiple occurrences, to expand subexpressions or in some other way reorder variables. The operations described in this section let you expand subexpressions and collect and reorder variables.

A *subexpression* consists of a function and its arguments. The function that defines a subexpression is called the *top-level* function for that subexpression. For example, in the expression `'A+B*C/D'`, * is the top level function in the subexpression `'B*C'`, / is the top-level function in the subexpression `'B*C/D'`, and + is the top level function in the subexpression `'A+B*C/D'`.

Collecting Terms and Expanding Subexpressions

Keys	Programmable Command	Description
↰ ALGEBRA:		
COLCT	COLCT	Simplifies the algebraic in level 1 by collecting like terms.
EXPA	EXPAN	Rewrites the algebraic in level 1 by expanding subexpressions that contain products and powers.

Collecting Terms

COLCT simplifies an algebraic by "collecting" like terms. Specifically, COLCT:

- 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.

Expanding Products and Powers

EXPAN rewrites an algebraic by expanding products and powers. Specifically, EXPAN:

- Distributes multiplication and division over addition. For example, `'A*(B+C)' EXPAN` returns `'A*B+A*C'`.
- Expands powers over sums. For example, `'A^(B+C)' EXPAN` returns `'A^B*A^C'`.
- Expands positive power integers. For example, `'X^5' EXPAN` returns `'X*X^4'`. The square of a sum, for example, `'(X+Y)^2'` expands to `'X^2+2*X*Y+Y^2'`.

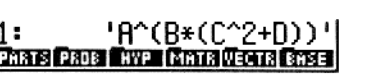
EXPAN does not carry out all possible expansions of an algebraic in a single execution. Instead, EXPAN works down through the subexpression hierarchy, stopping in each branch of the hierarchy when it finds a subexpression that can be expanded. It first examines the top-level subexpression (the top level subexpression is the algebraic itself); if that is suitable for expansion, it is expanded and EXPAN stops. If not, EXPAN examines each of the second-level subexpressions. Any of those that are suitable are expanded. The remaining second-level subexpressions are then examined. This process continues down through the hierarchy until an expansion halts further searching down each branch.

Example: Complete Expansion. Expand the expression:

```
'A^(B*(C^2+D))'
```

Key in and enter the expression.

' A y^x ◀ ( ) B × ◀ ( )
C y^x 2 + D
ENTER



Expand the expression. The top-level function is the left ^. Since its associated subexpression (the complete expression itself) cannot be expanded, the second-level function * is examined. One of the arguments in its associated subexpression is a sum (C^2+D), so the product is distributed.

◀ ALGEBRA EXPA



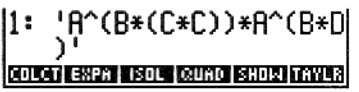
Expand the expression again. The top level function is still the left ^, but now its argument is a sum, so the power is expanded over the sum.

EXPA



One more expansion is possible. The top level function is now the middle *. Since it cannot be expanded, the second-level functions, the outside ^'s, are examined. They cannot be expanded, so the third-level function, the middle ^, is examined. Its argument is a positive integer, so the power is expanded.

EXPA



The expression is now completely expanded. Optionally, you can now collect like terms.

COLCT



The Rules Transformations

The Rules transformations are algebraic-rearrangement operations that are narrower in their scope than EXPAN and COLCT. The Rules transformations let you, in small steps, direct the path of an algebraic rearrangement.

To apply a Rules transformation to an algebraic, you must first put that algebraic in the EquationWriter application. (Either build the algebraic in the EquationWriter application or put the algebraic in level 1 of the stack and press ▼.) Then, in a three-step process:

1. Use the *Selection environment* to specify the subexpression that you want to rearrange.

2. Select the RULES menu. The relevant Rules transformations for that subexpression are displayed.
3. Execute the desired transformation.

In this section, the definition of "subexpression" in the previous section is expanded to include individual objects. For example, in the expression 'A+B', the Selection environment lets you specify A as a subexpression.

The Selection Environment—Specifying a Subexpression

After you put the algebraic you want to rearrange in the EquationWriter application, press ◀. This activates the Selection environment, displaying the Selection menu and *selection cursor*. The object in the equation that is currently highlighted is the *specified object*. The subexpression defined by the specified object is the *specified subexpression*.

Operations in the Selection Menu

RULES	Selects a menu of relevant rearrangement transformations for the specified subexpression.
EDIT	Returns the specified subexpression to the command line for editing (see "Editing a Subexpression" on page 244 in chapter 16 for more information).
EXPR	Highlights the specified subexpression. (+/- works just like EXPR and also can be executed when a transformation menu is displayed.)
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.

Operations in the Selection Menu (continued)

EXIT	Exits the Selection environment, restoring the normal 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.

Move the selection cursor from object to object in the expression or equation with the cursor keys. At any time you can press EXPR to highlight the subexpression defined by the specified object. This lets you identify exactly which subexpression will be affected by a subsequent Rules transformation.

Selecting the Rules Transformations

After you specify the subexpression you want to work with, press RULES to select the menu of Rules transformations that applies to that subexpression. You can press RULES with either the object or the associated subexpression highlighted.

Executing a Rules Transformation

Press the corresponding menu key to execute a transformation. After the transformation is executed, the selection cursor highlights the new top level object. In addition, the relevant transformation menu for that new object is displayed.

Note that not necessarily all transformations in a menu are applicable to the specified subexpression. If a transformation is not applicable, the HP 48 beeps when you press the corresponding menu key.

Exiting a RULES Menu

There are four ways to exit a RULES menu. Press:

- Any cursor key to restore the Selection menu. The selection cursor moves to the next object in the indicated direction if possible.
- [−] to restore the Selection menu without moving the selection cursor.
- [ENTER] to exit the EquationWriter application and put the algebraic on the stack.
- [ATTN] to exit the EquationWriter application and abandon the algebraic.

Rules Examples



Note

The following tables show objects or expressions in their *command-line form*, before and after execution of a Rules transformation. Remember, however, that you execute a Rules transformation *in the EquationWriter application*, and that the new expression remains in the EquationWriter application. You can return the new expression to the command line by pressing [ENTER].

The tables do *not* include all the patterns for which a transformation is applicable.

Universal Transformations. The following seven transformations apply to any subexpression. These transformations appear as the last seven entries of every RULES menu.

DNEG (Double-negate).

Before	After
A	-A

DINV (Double-invert).

Before	After
A	INV(INV(A))

*1 (Multiply by 1).

Before	After
A A+B/1	A*1 A+B

^1 (Raise to the power 1).

Before	After
A	A^1

/1 (Divide by 1).

Before	After
A A+B*1	A/1 A+B

`+1−1` (Add 1 and subtract 1).

Before	After
A	A+1−1

`COLCT` (Collect). `COLCT` executes a limited form of the COLCT command in the ALGEBRA menu. It works only on the subexpression defined by the specified object and it leaves the coefficients of collected terms as sums or differences.

Before	After
(2+3)*X	5*X
2*X+3*X	(2+3)*X

(The COLCT command in the ALGEBRA menu returns `5*X` in both cases.)

Moving Terms. `←T` and `T→` are used to move a *term* over its "nearest neighbor" to the left or right. A term is any one of the following:

- An argument of + or − (a summand).
- An argument of * or / (a factor).
- An argument of =.

Note in the following examples that these two operations ignore parentheses. You can make these operations respect parentheses by executing `*1` to make the parenthetical subexpression a term.

←T (Move-term-left). ←T moves the nearest neighbor to the right of the specified function over the nearest neighbor to the left of the function.

Before	After
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). →T moves the nearest neighbor to the left of the specified function over the nearest neighbor to the right of the function.

Before	After
A+B=(D+E)	A=-B+(D+E)
A*B=(X+Y)	A=INV(B)*(X+Y)

Building and Moving Parentheses.

(()) (Parenthesize-neighbors). (()) parenthesizes the nearest neighbors of + or *. The operation has no effect if the specified function is the first (or only) function in the expression, since these parentheses are already present, but hidden.

Before	After
A+B+C+D	A+(B+C)+D

(← expands the subexpression associated with the specified function to include the next term to the left. Note that this operation, and the following three operations, may result in a matched pair of parentheses disappearing.

Before	After
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.

Before	After
A+(B*C)+D+E	A+(B+C*D)+E

Commutation, Association, and Distribution.

←→ (Commute). ←→ commutes the arguments of the specified function.

Before	After
A*B	B*A
INV(A)*B	B/A

←A (Associate-left).

Before	After
A+(B+C)	A+B+C
A*(B/C)	A*B/C
A^(B*C)	A^B^C

A→ (Associate-right).

Before	After
(A+B)+C	A+(B+C)
(A*B)/C	A*(B/C)
(A^B)^C	A^(B*C)

→() (Distribute-prefix-function).

Before	After
-(A+B)	-A-B
INV(A/B)	INV(A)*B
IM(A*B)	RE(A)*IM(B)+IM(A)*RE(B)

←D (Distribute-left).

Before	After
(A+B)*C	A*C+B*C
(A/B)^C	A^C/B^C

D→ (Distribute-right).

Before	After
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). ←M 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. ←M also merges sums where only one argument is a product.

Before	After
(A*B)*(A*C)	A*(B+C)
EXP(A)*EXP(B)	EXP(A+B)
A+A*B	A*(1+B)

M→ (Merge-factors-right). M→ merges arguments of +, −, *, and /, where the arguments have a common factor. The → indicates that the right-hand factors are common. M→ also merges sums where only one argument is a product.

Before	After
(A*C)*(B*C)	(A+B)*C
A*B+1*B	(A+1)*B

−() (Double-negate and distribute). Execution of −() is equivalent to execution of DNEG followed by execution of →() on the resulting inner negation.

Before	After
A+B	−(−A−B)
LOG(INV(A))	−LOG(A)

1/() (Double-invert and distribute). Execution of 1/() is equivalent to execution of DINV followed by execution of →() on the resulting inner inversion.

Before	After
A*B	INV(INV(A)/B)
EXP(A)	INV(EXP(−A))

Rearrangement of Exponentials.

L* (Replace log-of-power with product-of-log).

Before	After
LOG(A^B)	LOG(A)*B

L( ) (Replace product-of-log with log-of-power).

Before	After
LN(A)*B	LN(A^B)

E^ (Replace power-product with power-of-power).

Before	After
ALOG(A*B)	ALOG(A)^B

E( ) (Replace power-of-power with power-product).

Before	After
EXP(A)^B	EXP(A*B)

Adding Fractions.

AF (Add fractions). AF combines terms over a common denominator. (If the denominator is already common between two fractions, use M+.)

Before	After
A+(B/C)	(A*C+B)/C
(A/B)-C	(A-B*C)/B

Expansion of Trigonometric Functions.

→DEF (Expand-trigonometric-definition). →DEF replaces trigonometric, hyperbolic, inverse trigonometric, and inverse hyperbolic functions with their definitions in terms of EXP and LN.

(The following examples assume Radians mode.)

Before	After
COS(X)	(EXP(X*i)+EXP(-(X*i)))/2
ASINH(U)	-LN(√(1+U^2)-U)

TRG* (Expand as product-of-trigonometric-functions). TRG* expands trigonometric functions of sums and differences.

Before	After
SIN(X+Y)	SIN(X)*COS(Y)+COS(X)*SIN(Y)

Automatic Multiple Execution of Rules Transformations. Prefixing the following transformation keys with ↱ causes that transformation to execute repeatedly until no further change occurs:

- D→ and ←D.
- A→ and ←A.
- M→ and ←M.
- T→ and ←T.
- →) and (←.

Before	Operation	After
(A+B+C)*D	↱ ←D	A*D+B*D+C*D
A*(B+C+D)	↱ ←A	A+B+C+D
A*B*C*B+D*B	↱ M→	(A+C+D)*B
A+B+C+D=E	↱ T→	B+C+D=E−A
A+(B+C)+D+E	↱ →)	A+(B+C+D+E)

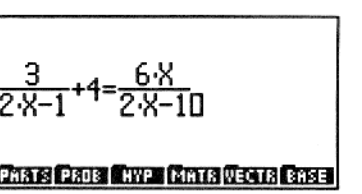
Summary Examples

Example 1. Solve for x in the equation:

$$\frac{3}{2x - 1} + 4 = \frac{6x}{2x - 1}$$

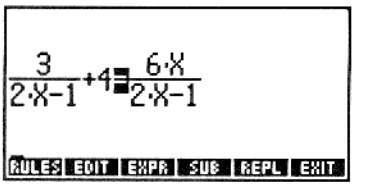
Key in the equation.

↰ EQUATION
3 ÷ 2X − 1 ▶ + 4 ↰ =
6X ÷ 2X − 1



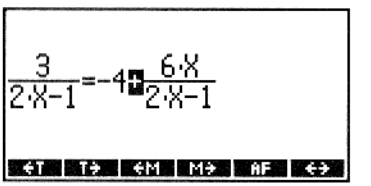
Activate the Selection menu and selection cursor. Move the cursor to the = sign.

◀ ◀



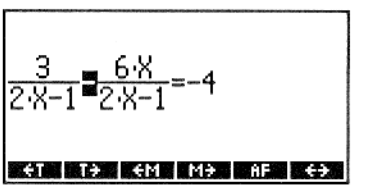
Move 4 to the right side of the equation.

RULES T→



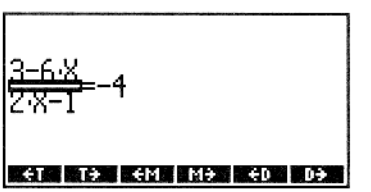
Move the quotient to the left side of the equation.

←T ←T



Now merge the two terms on the left over their common denominator.

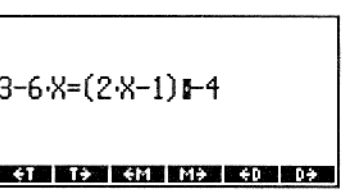
M→



Move the cursor to the = sign, then move the denominator to the right side of the equation.

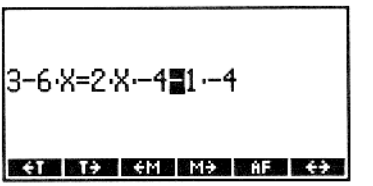
▶

RULES T→



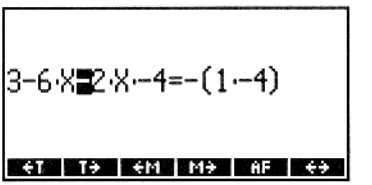
Distribute `-4` over `(2*X-1)`

`←D`



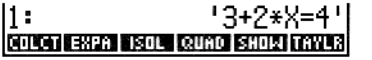
Move the cursor to the = sign, then move the term in X to the left side of the equation.

◀ (7 times)
`RULES` `←T`



Return the equation to the stack, select the ALGEBRA menu, and collect like terms.

ENTER ↰ ALGEBRA
`COLCT`



Solve for X.

▯ X `ISOL`

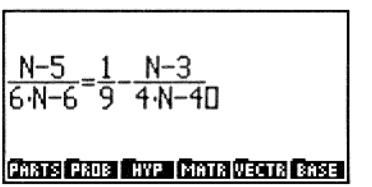


Example 2. Solve for n in the equation:

$$\frac{n-5}{6n-6} = \frac{1}{9} - \frac{n-3}{4n-4}$$

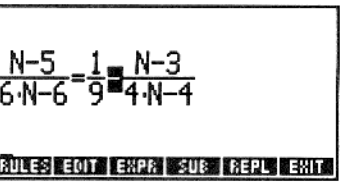
Key in the equation.

↰ EQUATION
▲ N − 5 ▶ 6N − 6 ▶
↰ − 1 ÷ 9 ▶ −
▲ N − 3 ▶ 4N − 4



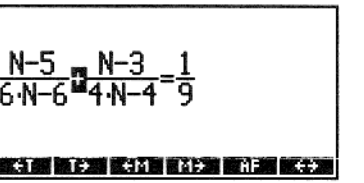
Activate the Selection menu and selection cursor. Move the cursor to the − sign between the two right-hand terms.

◀ ◀



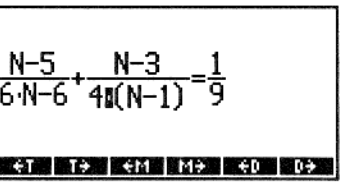
Move the rightmost term to the left side of the equation.

RULES ←T ←T



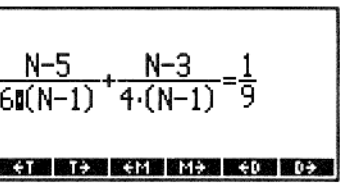
Move the cursor to the − sign in the denominator of that term and merge factors left.

▶ ▼
RULES ←M



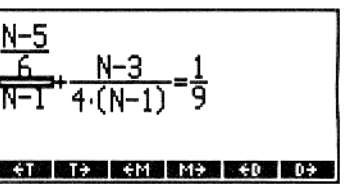
Move the cursor to the − sign in the denominator of the first term on the left side and merge factors left.

◀ ◀ ◀ ▼
RULES ←M



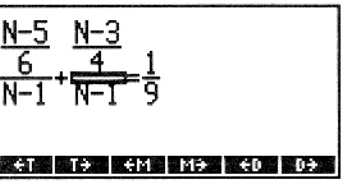
Move the cursor to the divide bar of that term and associate left.

▲ RULES NXT ←A



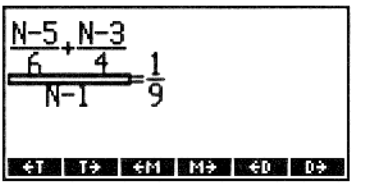
Move the cursor to the divide bar of the second term and associate left.

▶ ▶
RULES NXT →A



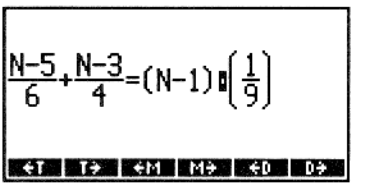
Move the cursor to the + sign between the two terms and merge factors right.

◀ RULES M→



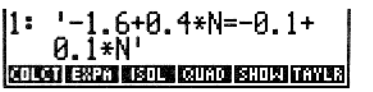
Move the cursor to the = sign and move term right.

▶ RULES T→



Return the equation to the stack. Set the display mode to 1 Fix (to see the result of the subsequent COLCT operation more easily). Expand terms and then collect like terms.

ENTER
↰ MODES 1 FIX
↰ ALGEBRA
EXPA COLCT



Solve for *N*.

' N QUAD



Return the display mode to Standard by pressing ↰ MODES STD.

User-Defined Transformations

If the built-in set of Rules transformations do not rearrange an algebraic in the form you desire, you can use ↑MATCH and ↓MATCH (◀ ALGEBRA NXT ↑MATCH or ↓MATCH) to specify a "customized" transformation. These commands search for a specified pattern in an algebraic and replace all occurrences of that pattern with a new pattern. If a replacement is made, the new expression is returned to level 2 and 1 (true) is returned to level 1. If a replacement is not made, the original expression is returned to level 2 and 0 (false) is returned to level 1. ↑MATCH starts its search at the lowest level subexpression(s) and works up; ↓MATCH starts with the complete algebraic and works down.

↑MATCH and ↓MATCH take the following arguments from the stack:

- From level 2, the algebraic to rewrite.
- From level 1, a list of the form { *'pattern'* *'replacement'* *'conditional'* }, where *'pattern'* is the subexpression for which to search, *'replacement'* is the new subexpression, and *'conditional'* is an *optional* specification of an additional condition that must be met before the replacement is made.

To facilitate the specification of the pattern, the *'pattern'* and *'replacement'* expressions can contain "wildcard" names that will match *any* subexpression. The wildcards in the *'replacement'* expression are replaced by the subexpressions they matched in the original expression. A wildcard variable is specified by prefixing any variable name with the & character (α ◀ ENTER).

Example: A User-Defined Transformation. An extension of the half-angle formula for sine is:

$$\sin(2z) = 2\sin(z)\cos(z)$$

There is no built-in Rules transformation for this formula, so create a program *HALF* that executes the transformation using ↓MATCH.

Program:	Comments:
«	
⟨ 'SIN(2*&wc)' '2*SIN(&wc)*COS(&wc)' ⟩ ↓MATCH	Puts the algebraic to rewrite on the stack, puts the list argument on the stack, and executes ↓MATCH. To key in &, press α ◀ ENTER.
»	
ENTER ' HALF STO	Puts the program on the stack and stores it in *HALF*.

Now execute *HALF* to transform the expression `'SIN(2*(X+1))'`.

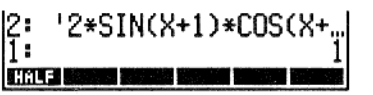
Enter the expression to transform.

' SIN 2 × ◀ ( ) X + 1 ENTER



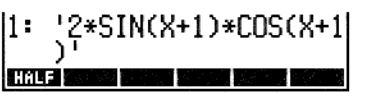
Select the VAR menu and execute *HALF*.

VAR HALF



Swap the expression into level 1 to see the complete expression.

◀ SWAP



The | (Where) Function

The | function (◀ ALGEBRA NXT |), read as "where," binds numeric values to variables that occur in a partially evaluated algebraic. Its purpose is to provide a means for stepwise evaluation of integrals and user-defined functions. For example, in chapter 23, "Calculus," you'll see how evaluation of an integral returns a symbolic result of the form:

'expr | var=upper limit−expr | var=lower limit

Here *expr* is the integrated expression, still in symbolic form, and *var* is the variable of integration. Evaluating again substitutes the limits of integration.

As another example, consider the user-defined function *DRV* created by executing:

```
'DRV(X)=∂X(X^2)' DEFINE
```

Evaluating *DRV*(2) returns the partially evaluated result:

```
'∂X(X)*2*X^(2-1)|(X=2)'
```

X is a local variable and exists only while the user-defined function *DRV* is being executed, so the | function here serves the purpose of letting the HP 48 show stepwise differentiation while "prolonging the life" of the (local) variable of differentiation. Evaluating again returns the final answer `4`.

| can also be executed with two arguments on the stack: an expression in level 2 and a list in level 1 of the form:

$$\{\ name_1\ expr_1\ \ldots name_n\ expr_n\ \}$$

For example, executing `'A+B' { A 'C+D' B 7 } |` returns the expression `'C+D+7'`.

23

Calculus



The HP 48 calculus commands let you do the following with algebraic expressions:

- Step-by-step and complete differentiation.
- Summation of series.
- Taylor's polynomials.
- Symbolic and numerical integration.

A Calculus Example. For the expression:

```
'SIN(X)'
```

calculate:

- The derivative of the expression.
- The derivative of the fifth-order Taylor's polynomial of the expression.

Then evaluate both expressions for $X = 0.5$, and compare the results.

(This example assumes that variable X does not exist in the current directory.)

Select Radians mode, and key in the derivative of the expression, using the EquationWriter application.

[←] [RAD] (if necessary)
[←] [EQUATION]
[→] [∂] X [▶] [SIN] X



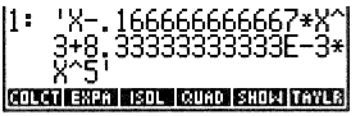
Put the expression on the stack, and evaluate it.

[ENTER] [EVAL] [EVAL]



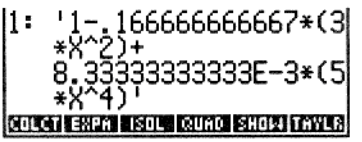
Now enter the original expression, the polynomial variable (X), and the order of the polynomial. Then find the Taylor's polynomial.

[ ] [SIN] X [ENTER]
X [ENTER]
5 [←] [ALGEBRA] TAYLR
[EVAL]



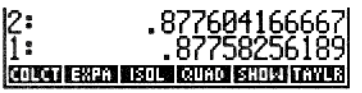
Find the derivative of the Taylor's polynomial with respect to X.

X [ENTER] [→] [∂]



Evaluate both expressions for $X = 0.5$

.5 X [STO]
[EVAL] [←] [SWAP] [EVAL]



The Taylor's polynomial is accurate to three decimal places.

Differentiation

The HP 48 can execute either *stepwise* or *fully-evaluated* differentiation of algebraic expressions.

Stepwise Differentiation

When ∂ takes its arguments in *algebraic* syntax, it differentiates an expression step-by-step. Keyed directly into the command line, the expression has the syntax:

'∂*var*(*expression*)'

where *var* is the variable of differentiation and *expression* is the expression to be differentiated.

Using the EquationWriter Application to Key In a Derivative. The EquationWriter application lets you key in a derivative in a graphical form that's easy to read and understand. Page 233 in chapter 16 describes the rules for keying in a derivative using the EquationWriter application.

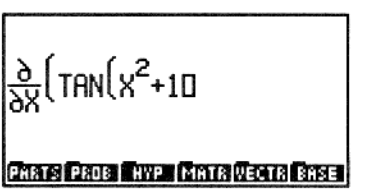
Example: Stepwise Differentiation. Calculate step-by-step the expression:

$$\frac{d}{dx}\tan(x^2 + 1)$$

(This example assumes that variable *X* does not exist in the current directory.)

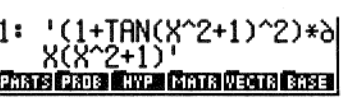
Set the angle mode to Radians. Select the EquationWriter application and key in the derivative.

◀ RAD (if necessary)
◀ EQUATION
▶ ∂ X ▶
TAN X y^x 2 ▶ + 1



Evaluate the expression.

EVAL



The result still contains a derivative, illustrating the chain rule of differentiation:

$$\frac{d}{dx}\tan(x^2+1) =$$

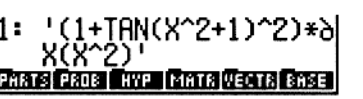
$$\frac{d}{d(x^2+1)}\tan(x^2+1)\times\frac{d}{dx}(x^2+1) =$$

$$(1+\tan^2(x^2+1))\times\frac{d}{dx}(x^2+1)$$

The derivative of the tangent function has been evaluated.

Next, evaluate the derivative of $x^2 + 1$.

EVAL



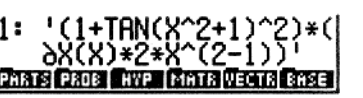
The result represents the derivative of a sum:

$$\frac{d}{dx}(x^2+1) = \frac{d}{dx}x^2 + \frac{d}{dx}1$$

The derivative of 1 is 0, so the term disappears.

Next, evaluate the derivative of x^2.

EVAL



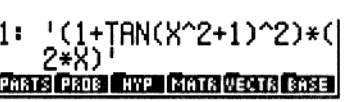
The result again reflects the chain rule:

$$\frac{d}{dx}x^2 = \frac{d}{dx}(x)^2\times\frac{d}{dx}(x)$$

The derivative of x^2 has been evaluated.

Evaluate the final derivative.

EVAL



Complete Differentiation

To differentiate an expression completely in one step, execute the ∂ command with two arguments:

- In level 2, the expression to be differentiated.
- In level 1, the variable of differentiation.

Example: Complete Differentiation. Calculate in one step the expression:

$$\frac{d}{dx}\tan(x^2 + 1)$$

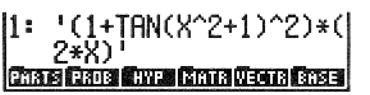
Enter the expression. Enter the variable of differentiation.

↑ TAN X y^x 2 + 1
ENTER
↑ X ENTER



Differentiate the expression.

→ ∂



Differentiation of User-Defined Functions

User-defined functions are differentiable. (See the example on page 150 in chapter 10, "User-Defined Functions.")

Advanced Topic: User-Defined Derivatives

If ∂ is applied to an HP 48 function for which a built-in derivative is not available, ∂ returns a new function whose name is "der" followed by the original function name. The new function will have arguments that are the arguments of the original function, plus the arguments' derivatives. For example, the HP 48 definition of % does not include a derivative. If you evaluate '∂Z(%(X,Y))', you obtain:

'der%(X,Y,∂Z(X),∂Z(Y))'

Each argument to the % function results in two arguments to the der% function. In this example, the X argument results in X and ∂Z(X) arguments, and the Y argument results in Y and ∂Z(Y) arguments.

You can further differentiate by creating a user-defined function to represent the derivative. Here is a derivative for %:

```
« → x y dx dy '(x*dy+y*dx)/100' » 'der%' STO
```

With this definition you can obtain a correct derivative for the % function. For example:

```
'%(X,2*X)' 'X' ∂ COLCT
```

returns '.04*X'.

Similarly, if ∂ is applied to 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. For example, differentiating the formal user function 'f(x1,x2,x3)' with respect to x returns:

```
'derf(x1,x2,x3,∂x(x1),∂x(x2),∂x(x3))'
```

Summations

The Σ function lets you calculate the value of a finite series. Keyed directly into the command line, the algebraic syntax for a summation is:

'Σ(*index=initial value, final value, summand*)'

Using the EquationWriter Application to Key In a Summation. The EquationWriter application lets you key in a summation in a graphical form that's easy to read and understand. Page 235 in chapter 16 describes the rules for keying in a summation using the EquationWriter application.

Example 1: Calculating the Sum of a Finite Series. Calculate:

$$\sum_{n=1}^{50} \frac{(-1)^n n}{2^n}$$

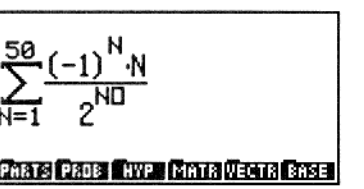
Select the EquationWriter application and key in the Σ sign. Key in the summation index and its initial value. Key in the final value.

◀ EQUATION
▶ Σ
N ▶ 1 ▶
50 ▶



Key in the summand.

◀ ( ) − 1 ▶
y^x N ▶ N
÷ 2 y^x N



Calculate the sum.

EVAL



The result is returned to level 1.

Example 2: Evaluating a Series to Suggest Convergence. Consider the geometric series:

$$\sum_{n=1}^{\infty} r^{n-1}$$

Part 1. Evaluate the series to suggest if it converges or diverges for $r = 0.5$.

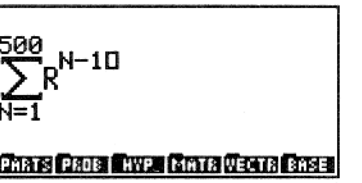
Set system flag −21 so that numbers larger than MAXR (*maximum real number*) cause an overflow error. Then, select the EquationWriter application, and key in the summation sign and the summation index at its initial value.

21 [+/−] [→] [MODES] [NXT] [SF]
[←] [EQUATION]
[→] [Σ]
N [▶] 1 [▶]



Key in the final value for the summation index. Since the HP 48 can't represent infinity, substitute a large number. Key in the summand.

500 [▶]
R [y^x] N [−] 1



Enter the expression, then copy it twice; you will use it again in this example. Store the value 0.5 in *R* and calculate the sum. (The calculation takes about 20 seconds.)

[ENTER] [ENTER] [ENTER]
.5 ['] R [STO]
[EVAL]



Now change the final value of the index to 1000 and calculate the sum. (The calculation takes about 45 seconds.)

[▶] [←] [EDIT]
[▶] (7 times) [DEL] 10 [ENTER]
[EVAL]

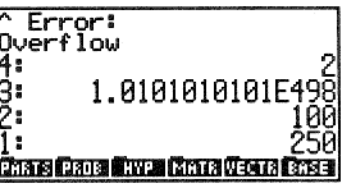


The calculations suggest that the series converges to 2.

Part 2. Evaluate the series to suggest if it converges or diverges for $r = 100$.

Return the remaining copy of the expression to level 1, store 100 in *R*, and calculate the sum.

▲
▲ ▲ ROLL ATTN
100 ' R STO
EVAL



An overflow error occurs, suggesting that the series diverges.

Using the Stack for Calculating Summations. The previous sections showed you how to execute a summation calculation by evaluating an algebraic expression. You can also calculate a summation by executing the Σ function with the following arguments:

- In level 4, the summation index.
- In level 3, the initial value of the summation index.
- In level 2, the final value of the summation index.
- In level 1, the summand.

Taylor's Polynomial Approximations

The TAYLR command (↰ ALGEBRA TAYLR) lets you compute a Taylor's polynomial for an algebraic expression. The expression is evaluated at $x = 0$ (known as a Maclaurin series). TAYLR takes three arguments from the stack:

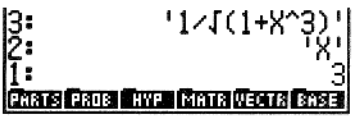
1. From level 3, the expression.
2. From level 2, the series variable.
3. From level 1, the order of the polynomial.

Example: Calculating a Taylor's Polynomial. Calculate the 3rd-order Taylor's polynomial for:

$$\frac{1}{\sqrt{1+x^3}}$$

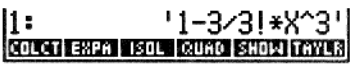
Enter the expression, the polynomial variable, and the order of the polynomial.

[ ] 1 ÷ [√x] [←] [( )]
1 [+] X [y^x] 3
[ENTER]
[ ] X [ENTER]
3 [ENTER]



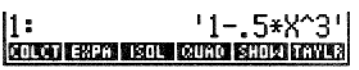
Select the ALGEBRA menu, key in the order of the polynomial, then execute the approximation. (The calculation takes about 30 seconds.)

[←] [ALGEBRA]
TAYLR



Evaluate again to complete the calculation.

[EVAL]



Translating the Point of Evaluation. If you are interested in the behavior of a function in a particular region away from zero, its Taylor's polynomial will be more useful if you translate the point of evaluation to that region. 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. Although TAYLR always evaluates the function and its derivatives at zero, you can effectively translate the point of evaluation away from zero by changing variables in the expression. For example, suppose the function is an expression in X, and you want the Taylor's polynomial at $X = 2$. To translate the point of evaluation by changing variables:

1. Purge Y.
2. Store `'Y+2'` in `X`.
3. Evaluate the original function to change the variable from X to Y.
4. Find the Taylor's polynomial at $Y = 0$.

Now translate the function back to X:

1. Purge X.
2. Store `'X-2'` in `Y`.
3. Evaluate the new function to change Y to X.

Integration

The HP 48 can execute *symbolic* integration of many expressions with known antiderivatives. If the HP 48 fails to produce an answer using symbolic techniques, you can *estimate* an answer by executing *numerical* integration.

Symbolic Integration

Symbolic integration means calculating an integral by finding a known antiderivative and then substituting specified limits of integration. Keyed directly into the command line, the algebraic syntax for an integral is:

$$\int(\textit{lower limit, upper limit, integrand, var})$$

where *var* is the variable of integration.

Using the EquationWriter Application to Key In an Integral. The EquationWriter application lets you key in an integral in a graphical form that's easy to read and understand. Page 234 in chapter 16 describes the rules for keying in an integral using the EquationWriter application.

Evaluation of an integral in algebraic syntax returns a result to level 1:

- If the result is a *closed-form* expression (if there is no integral sign in the result), a symbolic integration was successful.
- If the result still contains an integral sign, you can try rearranging the expression and evaluating again. If rearranging fails to produce a closed form result, you must estimate the answer with numerical integration.

A closed-form result has the form:

$$'\textit{result}\,|\,\textit{var}=b-\textit{result}\,|\,\textit{var}=a'$$

where *result* is the closed-form integral, *var* is the variable of integration, *b* is the upper limit, and *a* is the lower limit. (The | (where) function is discussed in chapter 22, "Algebra," on page 416.)

Press [EVAL] again to substitute the limits of integration into the variable of integration. This completes the procedure.

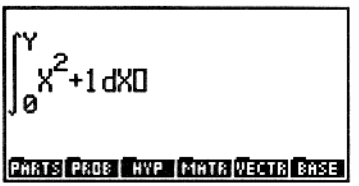
Example: Symbolic Integration. Calculate:

$$\int_0^y (x^2+1)\,dx$$

(This example assumes variable *Y* does not exist in the current directory.)

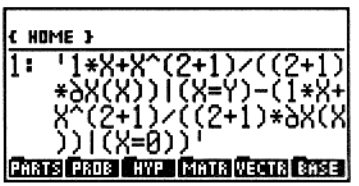
Select the EquationWriter application and key in the ∫ sign and the limits. Key in the integrand and the variable of integration.

[←] [EQUATION]
[→] [∫]
0 [▶] Y [▶]
X [y^x] 2 [▶] [+] 1
[▶] X



Evaluate the expression.

[EVAL]



The result is closed form. Now evaluate again to substitute the limits into the variable of integration.

[EVAL]



How the HP 48 Does Symbolic Integration. The HP 48 does symbolic integration by *pattern matching*. The HP 48 can integrate:

- All the built-in functions whose antiderivatives are expressible in terms of other built-in functions — for example, SIN is integrable since its antiderivative COS is a built-in function. The arguments for these functions must be linear.
- Sums, differences, and negations of built-in functions whose antiderivatives are expressible in terms of other built-in functions — for example, `'SIN(X)-COS(X)'`.
- Derivatives of all the built-in functions — for example, `'INV(1+X^2)'` is integrable because it is the derivative of the built-in function ATAN.

- Polynomials whose base term is linear—for example, `'X^3+X^2-2*X+6'` is integrable since `X` is a linear term. `'(X^2-6)^3+(X^2-6)^2'` is not integrable since `X^2-6` is not linear.
- Selected patterns composed of functions whose antiderivatives are expressible in terms of other built-in functions—for example, `'1/(COS(X)*SIN(X))'` returns `'LN(TAN(X))'`.

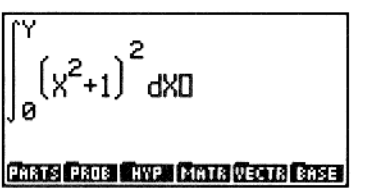
Example: Symbolic Integration. Calculate:

$$\int_0^y (x^2+1)^2 dx$$

(This example assumes variable *Y* does not exist in the current directory.)

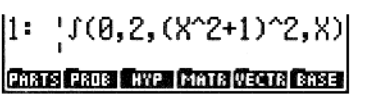
Select the EquationWriter application, then key in the integral sign, the limits, the integrand, and the variable of integration.

◀ EQUATION
∂ ∫
0 ▶ Y ▶
◀ ( ) X y^x 2 ▶ + 1
▶ y^x 2 ▶ ▶ X



Calculate the integral.

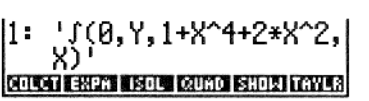
EVAL



The HP 48 does not make progress because the term `(X^2+1)` is not linear.

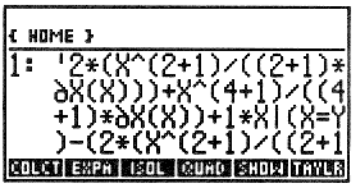
Try rearranging the expression by expanding and collecting.

◀ ALGEBRA
EXPA EXPA EXPA
COLCT



Now evaluate the rearranged expression.

EVAL



Substitute the limits of integration to complete the procedure.

EVAL



Taylor's Polynomial Approximation of the Integrand

The TAYLR command can be used to approximate in polynomial form expressions that are otherwise not integrable.

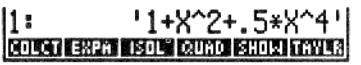
Example: Taylor's Polynomial Approximation of the Integrand. Calculate:

$$\int_0^y e^{x^2} dx$$

The expression e^{x^2} is not integrable by any of the methods thus far described in this chapter. However, you can calculate a Taylor's polynomial for this expression and then integrate the polynomial. For this example, calculate the 4th-order polynomial.

Enter the expression. Enter the series variable. Select the ALGEBRA menu, key in the order of the polynomial, then execute the calculation. The calculation takes about 15 seconds. Then complete the evaluation.

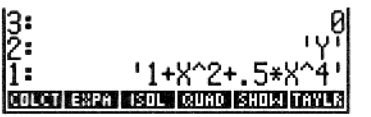
' ◀ e^x X y^x 2
ENTER
' X ENTER
4 ◀ ALGEBRA
TAYLR EVAL



Since the integrand is already on the stack, use stack arguments to calculate the integral. (The instructions for using stack arguments for integration are discussed on page 436.)

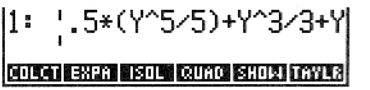
Enter the lower and upper limits and move the integrand to level 1.

0 ENTER Y ENTER
▲
▲ ▲ ROLL
ATTN



Enter the variable of integration and integrate the expression. Then complete the evaluation.

' X ENTER
→ ∫
EVAL



The approximation of the integral is returned to level 1. Note that this approximation becomes less accurate as *Y* increases in value.

Numerical Integration

Numerical integration lets you approximate a definite integral when symbolic integration cannot generate a closed-form result. Numerical integration employs an iterative numerical procedure to obtain the approximation.

To execute numerical integration:

1. Specify the accuracy factor for the integrand. The accuracy factor determines the acceptable tolerance between the final iterations of the numerical procedure. Except in rare cases, this factor is the percent error in the result. The display format specifies the accuracy factor:
   - Press ← MODES.
   - Set the display mode to *n* Fix. For example, to specify an accuracy factor of 0.0001 (0.01%), press 4 FIX.
2. Enter the integral.
3. Press → →NUM.

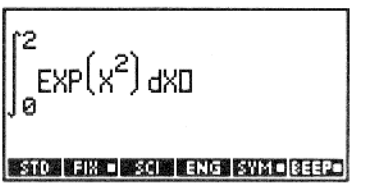
Example: Numerical Integration. Use numerical integration to calculate:

$$\int_0^2 e^{x^2} dx$$

Specify an accuracy factor of 0.0001.

Specify the accuracy factor. Select the EquationWriter application, then key in the integral sign and the limits of integration. Key in the integrand and the variable of integration.

◀ MODES 4 FIX
◀ EQUATION
▶ ∫ 0 ▶ 2 ▶
◀ e^x X y^x 2 ▶
▶ ▶ X



Calculate the numerical approximation. (The HP 48 takes about 25 seconds to execute the calculation.)

▶ →NUM



(In the previous example, you approximated an answer for the same integral by calculating a Taylor's polynomial for the integrand. Evaluation of that integral for $Y = 2$ returns the inaccurate result 5.53.)

The Accuracy Factor and the Uncertainty of Integration. Numerical 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, the more the sample points, the greater the accuracy. There are two reasons why you might want to limit the accuracy of the integral:

1. The length of time to calculate the integral increases as the number of sample points increases.
2. There are inherent inaccuracies in each calculated value of $f(x)$:
   - *Empirically-derived* constants in $f(x)$ may be inaccurate. For example, if $f(x)$ contains empirically-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 $f(x)$.

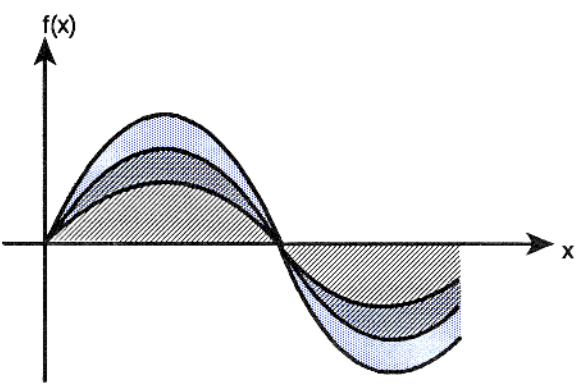
To indirectly limit the accuracy of the integral, you specify the *accuracy factor* of the *function*, defined as:

$$\textit{accuracy factor} \geq \left| \frac{\text{true value of } f(x) - \text{computed value of } f(x)}{\text{computed value of } f(x)} \right|$$

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:

$$\textit{uncertainty of integration} \leq \textit{accuracy factor} \times \int |f(x)|\, dx$$



The striped area is the value of the integral. The blue-shaded area is the value of the uncertainty of integration. It is the weighted sum of the errors of each computation of $f(x)$. You can see that at any point x, the uncertainty of integration is proportional to $f(x)$.

The numerical integration algorithm uses an iterative method, doubling the number of sample points in each successive iteration. At the end of

each iteration, it calculates both the integral and the uncertainty of integration. It then compares the value of the integral calculated during that iteration with the values calculated during the previous two iterations. If the difference between any one of these three values and the other two is less than the uncertainty of integration, 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*.

It is extremely unlikely that the errors in each of three successive calculations of the integral — that is, the differences between the actual integral and the calculated values — would all be larger than the disparity among the approximations themselves. *Consequently, the error in the final value will almost certainly be less than the uncertainty of integration.*

Example: The Accuracy Factor and the Error of Integration. Certain problems in communications theory require calculating an integral (sometimes called the *sine* integral) of the form:

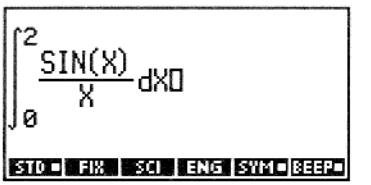
$$\mathrm{Si}(t) = \int_0^t \frac{\sin x}{x}\,dx$$

Find Si(2 degrees).

Since the function $f(x) = \sin x/x$ is a purely mathematical expression containing no empirically-derived constants, the only constraint on the accuracy of the function is the round-off error introduced by the calculator. It is, therefore, at least analytically reasonable to specify an accuracy factor of 1×10^{-11}.

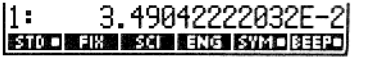
Set the angle mode to Degrees. Set the display mode to Standard. Select the EquationWriter application. Key in the integral sign and limits of integration. Key in the integrand and variable of integration.

[↰] [RAD] (if necessary)
[↰] [MODES] STD
[↰] [EQUATION]
[↱] [∫] 0 [▶] 2 [▶]
[SIN] X [▶] [÷] X [▶] [▶] X



Put the integral on the stack and copy it for later use. Calculate the integral.

[ENTER] [ENTER]
[↱] [→NUM]



Check the uncertainty of integration.

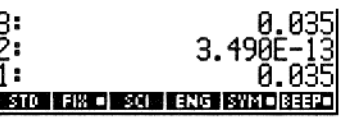
VAR IERR



The uncertainty of integration is significant only with respect to the last digit of the integral. The calculation takes about 5 seconds. If you can accept a less accurate answer, you can shorten the calculation time. Try an accuracy factor of 0.001.

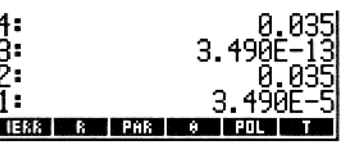
Set the display mode to 3 Fix, move the integral to level 1, then integrate.

← MODES 3 FIX
▲ ▲ ▲ ROLL ATTN
→ →NUM



Check the uncertainty of integration.

VAR IERR



The uncertainty of integration is much larger now. However, it is still relatively small compared to the value of the integral, and the calculation takes only one second.

Using the Stack for Integration

To do symbolic integration on the stack, execute the ∫ command with the following arguments:

- In level 4, the lower limit.
- In level 3, the upper limit.
- In level 2, the integrand.
- In level 1, the variable of integration.

To do numerical integration on the stack, first set Numerical Results mode (set flag −3), then follow the same procedure as above.

24

Time, Alarms, and Date Arithmetic



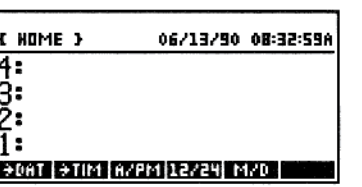
The operations in the Time application let you:

- Set and adjust the calculator clock.
- Set and review alarms.
- Execute date and time arithmetic.

A Time Example. Your manager has just assigned you a project and wants to meet with you in 30 days for a report on your progress. First set the current date and time (December 10, 1990, 10:45 AM). Then calculate the date that your progress report is due.

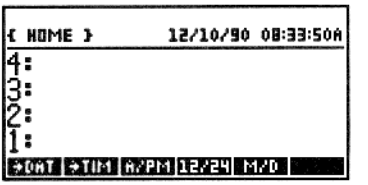
Set the display mode to 6 Fix. Select the TIME SET menu.

◀ MODES 6 FIX
◀ TIME SET



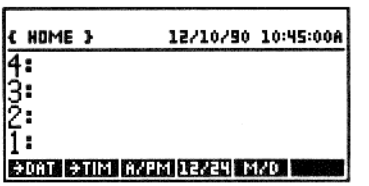
Set the current date.

12.101990 →DAT



Set the current time.

10.45 →TIM



To calculate the progress-report due-date, first return the current date to level 1.

◀ TIME NXT DATE



Key in the number of days and calculate the date.

30
DATE+

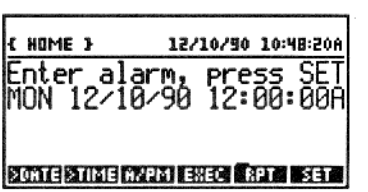


Your progress report is due 1.091991 (January 9, 1991).

Now set an alarm for 9:00 AM on January 9.

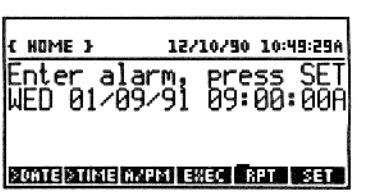
Select the TIME ALRM menu.

◀ TIME ALRM



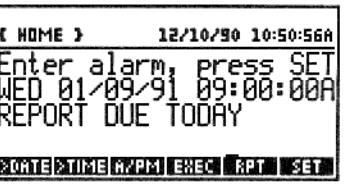
Set the alarm date and time. (The alarm date is already in level 1.)

>DATE
9 >TIME



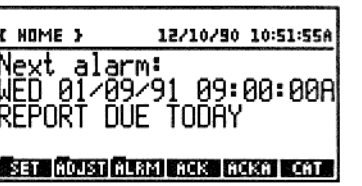
Enter an alarm message.

[→][ ] REPORT DUE TODAY
EXEC



Store the alarm in the system alarm list.

SET



The Structure of the Time Application

The first page of the TIME menu contains keys for doing basic Time operations: setting or adjusting the clock, and setting, acknowledging or reviewing alarms. The second and third pages contain date- and time-arithmetic commands.

Press [←][TIME] to display the first page of the TIME menu. In addition to the TIME menu, the HP 48 displays the current date and time in the status area, and the next alarm due, if there is one. You can press [←][REVIEW] at any time to redisplay the next due alarm.

Normally, the date and time are displayed only when you've selected the Time application. To display the date and time at all times, press [←][MODES] [NXT] CLK. Press CLK■ again to turn the date and time display back off.

First Page of the TIME Menu

Keys	Programmable Command	Description
↰ TIME:		
SET		Selects the SET menu for setting the calculator date and clock.
ADJST		Selects the ADJST menu for adjusting the calculator clock.
ALRM		Selects the ALRM menu for entering an alarm. The ALRM menu also contains commands for using alarms in programs.
ACK	ACK	Acknowledges the oldest past-due alarm.
ACKA	ACKALL	Acknowledges all past-due alarms.
CAT		Selects the Alarm Catalog for reviewing and editing existing alarms.

Setting the Date and Time

To set the date and time, select the TIME menu (↰ TIME) and press SET. This activates the TIME SET menu.

The TIME SET Menu

Keys	Programmable Command	Description
↩ TIME SET :		
→DAT	→DATE	Sets the number in level 1 as the current date.
→TIM	→TIME	Sets the number in level 1 as the current time.
A/PM		Switches the clock setting between AM and PM.
12/24		Switches between 12-hour and 24-hour format.
M/D		Switches between month/day/year and day.month.year format.

Setting the Date

To set the date:

1. Note the current date format. If the date contains slashes (for example, 11/21/90), the current date format is month/day/year. If the date contains periods (for example, 21.11.90), the current date format is day.month.year.
2. Key in the current date as a seven- or eight-digit number, using the current date format. For example, January 5, 1991 would be 1.051991 in month/day/year format (*MM.DDYYYY*) or 5.011991 in day.month.year format (*DD.MMYYYY*). You don't need to specify the year if it's the same as the year currently displayed by the calculator.
3. Press →DAT to set the date.

The range of allowable dates is from January 1, 1989 to December 31, 2088.

Setting the Time

To set the time:

1. Note the current time format—A or P after the time indicates 12-hour format.
2. Using the current format, key in the correct time as a number of the form *HH.MMSS*. For example, 1:16:47 PM would be keyed in as 1.1647 (in 12-hour format) or 13.1647 (in 24-hour format). (Note that when the calculator is set to 12-hour format, a time supplied in 24-hour format is acceptable.)
3. Press →TIM to set the clock.
4. For 12-hour format only: If necessary, press A/PM to switch between AM and PM.

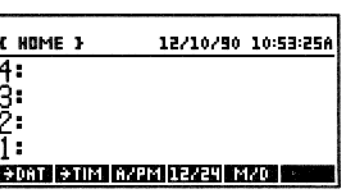
Changing the Date and Time Formats

To switch between month/day/year and day.month.year format, press M/D. To switch between 12- and 24-hour format, press 12/24.

Example: Setting the Time and Date. Set the time and date to 10:08 AM, April 20, 1990. Use 12-hour and month/day/year formats (press 12/24 and M/D if necessary).

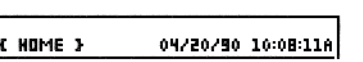
Select the TIME SET menu.

◀ TIME SET



Set the time (press A/PM if necessary to switch to AM) and date.

10.08 →TIM
4.201990 →DAT



Note the new time and date in the status area.

Adjusting the Time

To adjust the time, select the TIME menu and press ADJST. This activates the TIME ADJST menu.

The TIME ADJST Menu

Keys	Programmable Command	Description
↩ TIME ADJST:		
HR+		Increments the time by one hour.
HR−		Decrements the time by one hour.
MIN+		Increments the time by one minute.
MIN−		Decrements the time by one minute.
SEC+		Increments the time by one second.
SEC−		Decrements the time by one second.
CLKA	CLKADJ	Adds (or subtracts) the specified number of *clock ticks* to the time, where 8192 clock ticks equals 1 second. CLKADJ is used to programmatically change the calculator clock.

Setting Alarms

The HP 48 lets you set two types of alarms, *appointment* alarms and *control* alarms. These two types of alarms are distinguished by their *execution action*:

- When an appointment alarm comes due, it displays the (optional) message that you supplied as a string when you set the alarm. It also sounds a sequence of beeps for about 15 seconds or until you acknowledge it by pressing a key.

- When a control alarm comes due, it executes the object (typically a program) that you supplied when you set the alarm.

When you set an alarm, it is listed in the Alarm Catalog. The Alarm Catalog lets you review and edit any existing alarm.

To set an alarm, select the TIME menu and press ALRM.

The First Page of the TIME ALRM Menu

◀ TIME ALRM:		
>DATE		Sets the number in level 1 as the alarm date.
>TIME		Sets the number in level 1 as the alarm time.
A/PM		Switches the alarm time between AM and PM.
EXEC		Stores the object in level 1 as the alarm execution action. If the object is a string, the alarm is treated as an appointment alarm, displaying the contents of the string as the alarm message. If the object is *not* a string, the alarm is a control alarm, and the object is executed when the alarm comes due. (▶ EXEC recalls the current object to the stack.)
RPT		Selects the RPT menu for setting a repeat interval.
SET		Sets the alarm currently being constructed, and saves it in the system alarm list.

Appointment Alarms

To set an appointment alarm:

1. Key in the alarm date in the current date format, then press →DATE. (You don't need to specify the date if it's the same as the current date.)
2. Key in the alarm time in the current time format, then press →TIME.
3. Optional: Key in the alarm message (a string), then press EXEC.
4. Optional: Set the repeat interval (described in the next section).
5. Press SET to set the alarm and save the alarm in the system alarm list. The HP 48 automatically returns to the main TIME menu and redisplays the next due alarm.

Repeating an Alarm

You can set an alarm to repeat at a specified interval using the ALRM RPT menu. The commands in the ALRM RPT menu take a real number *n* from level 1. After the command is executed, the TIME ALRM menu is automatically redisplayed.

The RPT Menu

Keys	Programmable Command	Description
↶ TIME ALRM RPT:		
WEEK		Sets the repeat interval to *n* weeks.
DAY		Sets the repeat interval to *n* days.
HOUR		Sets the repeat interval to *n* hours.
MIN		Sets the repeat interval to *n* minutes.
SEC		Sets the repeat interval to *n* seconds.
NONE		Cancels the repeat interval and returns to the TIME ALRM menu.

Example: Setting a Repeating Alarm. Set an alarm for a weekly staff meeting on Fridays at 10:30 AM, beginning November 1, 1991.

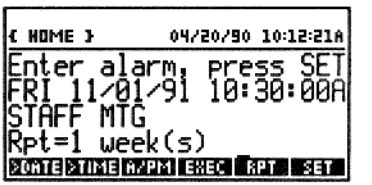
Select the TIME ALRM menu and set the alarm time, date, and message.

[←][TIME] ALRM
10.30 >TIME
11.011991 >DATE
[→][▪▪]
STAFF MTG EXEC



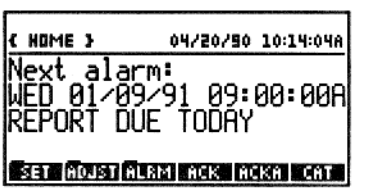
Select the RPT menu and set the repeat interval to one week.

RPT
1 WEEK



The ALRM menu is redisplayed. Set the alarm.

SET



The next due alarm is redisplayed.

Acknowledging an Appointment Alarm

When an appointment alarm comes due, the beeper sounds at short intervals for about 15 seconds and the alarm message is displayed. To acknowledge a current alarm, press any key while the beeper is sounding. When the alarm is acknowledged:

- The beep stops.
- The (•) annunciator is cleared from the display.
- The alarm message is cleared after a short delay.
- If the alarm is a repeating alarm, it is rescheduled.

Unacknowledged Appointment Alarms

If an appointment alarm is not acknowledged within 15 seconds, the beeper stops, the message is cleared from the display, and the alarm becomes "past due." The (•)) annunciator remains on to inform you of the past-due alarm. To acknowledge past due alarms:

1. Press [←][TIME]. The oldest past due alarm is shown in the display (date, time, and message).
2. Press ACK to acknowledge the alarm and clear it from the display and the system alarm list.
3. If more than one alarm is past due, the next oldest past due alarm is now displayed. Press ACK again to clear the alarm. When no alarms are past due, the (•)) annunciator turns off (assuming no other sources keep it on) and the next-due alarm is displayed.

Optionally, press ACKA to acknowledge *all* past due alarms with one keystroke.

Saving Acknowledged Nonrepeating Appointment Alarms. Normally, acknowledged nonrepeating appointment alarms are deleted from the system alarm list. To *save* acknowledged nonrepeating appointment alarms, set system flag −44.

Note that accumulating a large list of past due alarms (greater than 20) may affect calculator operations, so it's a good idea to manage the alarm list size.

Deleting Unacknowledged Repeating Appointment Alarms. Normally, unacknowledged repeating appointment alarms are rescheduled. To cause this type of alarm to be deleted when it becomes past due, set system flag −43.

Turning the Beeper Off. To prevent the audible tones from sounding when an appointment alarm becomes due, set flag −57.

Setting a Control Alarm

A *control* alarm *executes an object* (typically a program or a name that contains a program) at the specified date and time instead of displaying a message. To set a control alarm:

1. Set the alarm date and time using the same procedure as for normal alarms.
2. Enter the object to be executed, then press EXEC.
3. Press SET to set the control alarm.

When a control alarm comes due, a copy of the *alarm index* is returned to level 1 and 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. The alarm index is used by programmable alarm commands (discussed on page 453).

Acknowledging and Saving Control Alarms. A control alarm that comes due is *always* considered to be acknowledged. A control alarm (nonrepeating or repeating) that becomes due is *always* saved in the system alarm list. Therefore, flags −43 and −44 have no effect on control alarms.

Recovery from Short-Interval Repeating Alarms

It is 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 used to make the HP 48 take measurements at short intervals.

You can recover from this situation by pressing the ON and 4 keys simultaneously. This sets a state in the calculator that cancels the rescheduling of the next due alarm (presumably the short-interval repeat alarm). In turn, when that alarm comes due, or when the next key is pressed, the "no-reschedule" state of the calculator is cancelled so that future alarms are not affected.

Note the following:

- Since a key press cancels the "no-reschedule" state, you should wait until the alarm comes due before pressing any keys.
- If you want to restart the short-interval repeat alarm at a future time, you will need to edit it to reenable its rescheduling. Simply setting a new starting time accomplishes this.

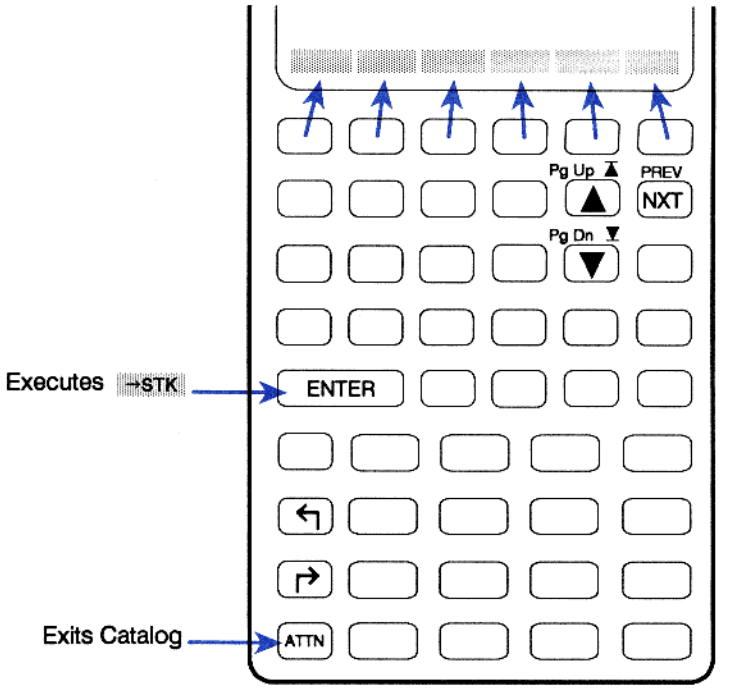
Reviewing and Editing Alarms

To review the current alarm schedule and modify existing alarms, select the Alarm Catalog: press [←][TIME] CAT or [→][TIME]. The system alarm list is displayed with the ▶ character pointing to the next alarm due. (If there are no alarms in the list, the HP 48 displays the message `Empty catalog`.)

The Alarm Catalog is a special environment where the keyboard is redefined and limited to special operations. You cannot go to another menu without exiting the catalog. In addition, you do not have access to the stack when you're in the Alarm Catalog. [▼] moves the ▶ pointer down the list (forward in time) and [▲] moves the pointer up the list (backward in time). The operations in the Alarm Catalog work on the selected entry.

Operations in the Alarm Catalog

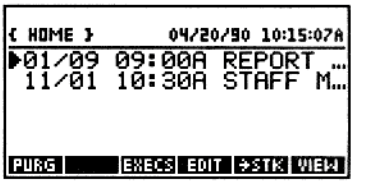
PURG	Deletes the selected alarm from the alarm list.
EXECS	Switches between displaying the time and date of each alarm entry and displaying the alarm execution action only.
EDIT	Removes the selected alarm from the system alarm list for editing and exits the catalog.
→STK	Copies the selected alarm to the stack.
VIEW	Views all information about the selected alarm.
▲	Moves the catalog pointer up one level. When prefixed with ◀, moves the catalog pointer up one page (◀ PgUp in the following keyboard illustration); when prefixed with ▶, moves the catalog pointer to the top of the catalog (▶ ▲ in the following keyboard illustration).
▼	Moves the catalog pointer down one level. When prefixed with ◀, moves the catalog pointer down one page (◀ PgDn in the following keyboard illustration); when prefixed with ▶, moves the catalog pointer to the bottom of the catalog (▶ ▼ in the following keyboard illustration).
ENTER	Copies the selected entry to the stack (same as →STK).
ATTN	Exits the catalog.



Example: Editing an Alarm. In the previous example, you set an alarm for a 10:30 staff meeting on Mondays. The meeting has been changed to 9:30 on the same day. Modify the alarm to reflect this change.

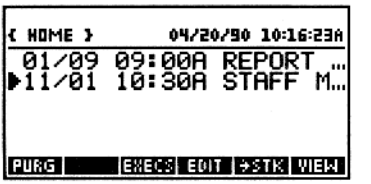
Select the alarm catalog in the Time application.

◀ TIME CAT



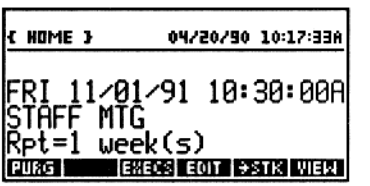
Press ▼ until ▶ points to the correct alarm. (The position of the alarm in the alarm list will vary depending on the specific dates you have used in preceding examples.)

▼ ...



View the alarm to check if this is the one you want to edit.

VIEW



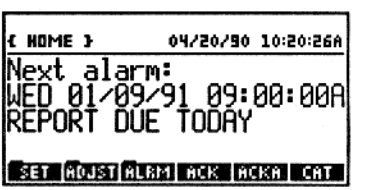
Edit the alarm.

EDIT



Set the new alarm time.

9.30 >TIME SET



The next due alarm is redisplayed.

Note that when you press EDIT, the selected alarm is *removed* from the alarm list and is not returned there until you press SET. It is, however, saved in a reserved variable *ALRMDAT* until you press SET.

Using Alarms in Programs

The following commands are for programmable alarm control. An alarm is specified on the stack by a list of the form:

{ *date time action repeat* }

where *date* and *time* are the alarm date and time, *action* is the execution action, and *repeat* is the repeat interval in *clock ticks* (1 clock tick is 1/8192 second).

Programmable Alarm Commands

Keys	Programmable Command	Description
↩ TIME ALRM (page 2):		
STOAL	STOALARM	Stores the alarm in level 1 into the system alarm list and returns its alarm index *n* to level 1. The argument for STOAL can take any one of the following four forms: *time*; { *date time* }; { *date time action* }; { *date time action repeat* }. If only the real number *time* is specified, the alarm date is the current date by default.
RCLAL	RCLALARM	Takes an alarm index *n* from level 1, and returns the corresponding alarm to level 1.
DELAL	DELALARM	Takes an alarm index *n* from level 1 and deletes the corresponding alarm from the system alarm list. If *n* = 0, deletes *all* alarms from the system alarm list.

Programmable Alarm Commands (continued)

Keys	Programmable Command	Description
FINDA	FINDALARM	Returns the alarm index *n* of the first alarm that comes due after the time specified in level 1 as follows: if the level 1 argument is a list of the form { *date time* }, returns the first alarm due after that date and time; if the level 1 argument is a real number *date*, returns the first alarm due after midnight on that date; if the level 1 argument is 0, returns the first *past due* alarm.

Date Arithmetic

Four commands on page two of the TIME menu execute date arithmetic.

The Date Arithmetic Commands

Keys	Programmable Command	Description
↰ TIME (page 2):		
DATE+	DATE+	Returns a past or future date in number form (DD.MMYYYY or MM.DDYYYY), given a date in level 2 and the number of days in level 1.

The Date Arithmetic Commands (continued)

Keys	Programmable Command	Description
DDAYS	DDAYS	(Delta days.) Returns the number of days between the dates in levels 2 and 1.
DATE	DATE	Returns the current date in number form (DD.MMYYYY or MM.DDYYYY).
TSTR	TSTR	(Timestring.) Returns any valid date and time in string form, given the date in number form in level 2, and the time in number form in level 1.

Example: Determining a Future Date. On July 15, 1991, you purchased a 120-day option on a piece of land. Determine the expiration date.

Select page two of the TIME menu.

[←] [TIME] [NXT]



Enter the known date. Key in the number of days, then calculate the expiration date.

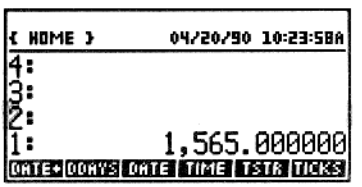
7.151991 [ENTER]
120 DATE+



Example: Calculating the Number of Days Between Two Dates. Find the number of days between April 20, 1982 and August 2, 1986.

Select page two of the TIME menu and enter the first date. Key in the second date and calculate the number of days.

[←] [TIME] [NXT]
4.201982 [ENTER]
8.021986 DDAYS



Time Arithmetic

The commands on page three of the TIME menu, and the TICKS command on page 2, let you execute time arithmetic.

HMS refers to *hours-minutes-seconds* format: *H.MMSSs* where:

- *H* is zero or more digits representing the number of hours.
- *MM* are two digits representing the number of minutes.
- *SS* are two digits representing the number of seconds.
- *s* is zero or more digits representing the decimal fraction part of seconds.

(Note that *H* can also represent *degrees* in angle calculations—see "Angle Conversion Functions" on page 142 in chapter 9.)

The Time Arithmetic Commands

Keys	Programmable Command	Description
↰ TIME (pages 2 and 3):		
TIME	TIME	Returns the current time in number form.
TICKS	TICKS	Returns the system time as a binary integer in units of 1/8192 second.
→HMS	→HMS	Converts a real number representing decimal hours to HMS format.
HMS→	HMS→	Converts a real number in HMS format to its decimal form.

The Time Arithmetic Commands (continued)

Keys	Programmable Command	Description
HMS+	HMS+	Adds two numbers in HMS format, returning the sum in HMS format.
HMS−	HMS−	Subtracts two numbers in HMS format, returning the difference in HMS format.

Example: Decimal to HMS Conversion. Convert 5.27 hours to its HMS equivalent.

Select the third page of the TIME menu.

◀ TIME NXT NXT



Key in the decimal time and execute the conversion.

5.27 →HMS



The answer is interpreted as 5 hours, 16 minutes, 12 seconds.

Example: HMS-Format Addition. Add 5 hours 50 minutes to 4 hours 30 minutes.

Select the third page of the TIME menu and add the two times.

◀ TIME NXT NXT
5.5 ENTER
4.3 HMS+



The answer is interpreted as 10 hours, 20 minutes.

The TICKS Command. The TICKS command enables elapsed time computations in programs. Program *FIBT* on page 551 demonstrates the use of the TICKS command.

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 index, and "Answers to Common Questions" in appendix A. If you can't find an answer in the manual, you can contact the Calculator Support department:

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

(503) 757-2004
8:00 a.m. to 3:00 p.m. Pacific time
Monday through Friday

For Service. If your calculator doesn't seem to work properly, refer to appendix A for diagnostic instructions and information on obtaining service. If you are in the United States and your calculator requires service, mail it to the Corvallis Service Center:

Hewlett-Packard
Corvallis Service Center
1030 N.E. Circle Blvd.
Corvallis, OR 97330, U.S.A.
(503) 757-2002

If you are outside the United States, refer to appendix A for information on locating the nearest service center.

HP Calculator Bulletin Board System. The Bulletin Board provides for the exchange of software and information between HP calculator users, developers, and distributors. It operates at 300/1200/2400 baud, full duplex, no parity, 8 bits, 1 stop bit. The telephone number is (503) 750-4448. The Bulletin Board is a free service — you pay for only the long-distance telephone charge.

Contents

Page 20 How to Use This Manual

Part 1: Building Blocks

Page 24 1: Getting Started
45 2: The Keyboard and Display
60 3: The Stack and Command Line
80 4: Objects
100 5: Calculator Memory
105 6: Variables and the VAR Menu
118 7: Directories
125 8: More About Algebraic Objects

Part 2: Hand Tools

Page 132 9: Common Math Functions
150 10: User-Defined Functions
156 11: Complex Numbers
169 12: Vectors
185 13: Unit Management
207 14: Binary Arithmetic
212 15: Customizing the Calculator

Part 3: Power Tools

Page 226 16: The EquationWriter Application
250 17: The HP Solve Application
283 18: Basic Plotting and Function Analysis
317 19: More About Plotting and Graphics Objects
345 20: Arrays
364 21: Statistics
386 22: Algebra
418 23: Calculus
437 24: Time, Alarms, and Date Arithmetic

HEWLETT PACKARD

Reorder Number
00048-90003

00048-90077 English
Printed in Canada 7/90