

MASTERING THE HP 48G/GX

A Step-By-Step, Easy to Read Introduction to
Operating and Programming the HP 48G/GX

Thomas Adams



Mastering the HP48G/GX

*A Step-by-Step, Easy-to-Read Introduction to
Operating and Programming the HP48G/GX*

Thomas Adams
Michigan State University



KENDALL/HUNT PUBLISHING COMPANY
4050 Westmark Drive Dubuque, Iowa 52002

Information in this book is presented without any kind of representation or warranty. No liability is assumed by the author resulting from the direct or indirect use of any material or information in this book. All information, instructions and examples are presented solely and exclusively for the purpose of illustration and explanation, not for application.

Information about Instructor's Materials is available from:

Thomas Adams, Ph.D.
Department of Physiology
Michigan State University
East Lansing, MI 48824-1101

Copyright © 1994 by Thomas Adams

ISBN 0-8403-9534-5

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the copyright owner.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Table of Contents

Table of Exercises	iv
Preface	vii
Chapter 1 In the Beginning	1
Section 1.1. Keyboard Designations	2
Section 1.2. Backing Out of an Operation	3
Section 1.3. Writing Words and Numbers	4
Section 1.3.1. Generating ALPHA Characters	4
Section 1.3.2. Generating Special Characters	6
Section 1.3.3. Selecting How Numbers are Displayed	7
Section 1.4. Setting the Clock, Calendar and Alarms	9
Chapter 2 Basic Operations	11
Section 2.1. Number Control	11
Section 2.2. The Importance of STACK Position	14
Section 2.3. Tools for STACK Control	16
Section 2.3.1. Exercises for LS STACK Operations	17
Section 2.3.2. Exercises for RS STACK Operations	19
Section 2.4. Calculations of Percent	19
Section 2.5. Variables as Numbers and <i>Vice Versa</i>	21
Section 2.6. Variables with Units	22
Section 2.7. Unit Conversions	25
Section 2.8. Customized Menus	29
Section 2.9. Arithmetic with Complex Units	32
Chapter 3 Writing and Solving Equations	35
Section 3.1. Writing an Equation and Solving it	35
Section 3.2. Cleaning-up VAR Menus	41
Section 3.3. Making Changes in Stored Equations	42

Section 3.4. An Alternative Way to Solve an Equation.	43
Section 3.5. Solving Equations Algebraically	45
Section 3.6. The LS DEF Operation	47
Section 3.7. A Special Feature For Special Cases	51
Section 3.8. Taking Score and Looking Ahead.	54
Chapter 4 Basic Statistics	56
Section 4.1. Evaluating a Single Column of Numbers	56
Section 4.2. Editing Stored Data	60
Section 4.3. Using Two Columns of Numbers	60
Section 4.3.1. Basic Calculations	60
Section 4.3.2. Curve-Fitting.	63
Section 4.3.3. Making Predictions	69
Section 4.4. A Mystery Solved with the HP48G	70
Section 4.5. Directories: Basic Concepts	74
Section 4.6. Directory Structure and Construction	75
Section 4.7. More Progress	78
Chapter 5 Basic Programming	79
Section 5.1. Getting Started	79
Section 5.1.1. A Program Map	79
Section 5.1.2. Menus for Program Writing	82
Section 5.2. Storing Equations and Data as Variables.	82
Section 5.3. Using Local Variables.	85
Section 5.4. Combining STACK and VAR Data	89
Section 5.5. Constructing Data Requests.	92
Section 5.6. Finding Program Errors	99
Chapter 6 Programming Strategies	101
Section 6.1. Programming with RPN.	101
Section 6.2. Other Ways to Enter Data.	103
Section 6.3. Decision Structures	108
Section 6.4. Loop Structures	119

Chapter 7 Flags	127
Section 7.1. The Basic Concept	127
Section 7.2. Where FLAGS Are And What They Do	128
Section 7.3. "User-defined FLAGS"	129
Section 7.4. Complex Choices with FLAGS	132
Section 7.5. Keeping Track	137
Section 7.6. Controlled FLAG Clearing	139
Section 7.7. Wrapping It Up	142
 Chapter 8 Sample Problems	 144
Population Growth	145
Relative Humidity	149
Checks	153
Plant Density	157
Useful Statistics	172
Automatic Curve-Fitting	174
 Chapter 9 File Transfer and Printing	 178
Section 9.1. File Transfer Operations	179
Section 9.1.1. Installing the Software	179
Section 9.1.2. File Transfer from the HP48G to a Tabletop Computer	180
Section 9.1.3. File Transfer from a Tabletop Computer to the HP48G	182
Section 9.1.4. File Transfer Between HP48Gs	183
Section 9.1.5. File Transfer Between HP48S/SX and HP48G/GX	184
Section 9.1.6. File Transfer Between HP48S/SXs.	186
Section 9.2. Printing HP48 View Window Displays	187
Section 9.2.1. Obtaining Prints of the HP48G/GX View Window	187
Section 9.2.2. Obtaining Prints of the HP48S/SX View Window	187
 Index	 189

Table of Exercises

Chapter 2 Basic Operations

Exercise 2.1. Basic "Reverse Polish Notation" (RPN) Operations	12
Exercise 2.2. A Complex Calculation Using the STACK	16
Exercise 2.3. Clearing STACK Lines	18
Exercise 2.4. Reordering The STACK	18
Exercise 2.5. Duplicating and Counting the STACK.	18
Exercise 2.6. RS STACK Operations	19
Exercise 2.7. Percent Calculations	20
Exercise 2.8. Sample Unit Conversions: Using Built-in Units	26
Exercise 2.9. Sample Unit Conversion: Mixing Built-in Units	28
Exercise 2.10. Complex Units Conversions	29
Exercise 2.11. Creating and Using Customized Menus.	31
Exercise 2.12. Calculations with Complex Units.	32

Chapter 3 Writing and Solving Equations

Exercise 3.1. Writing and Storing a Simple Equation	36
Exercise 3.2. Writing and Storing a Complex Equation	37
Exercise 3.3. Equation Solutions with RS SOLVE	38
Exercise 3.4. Equation Solutions with LS SOLVE	39
Exercise 3.5. Amending an Existing Equation Using EDIT	42
Exercise 3.6. Writing a Program	44
Exercise 3.7. Solving an Equation Algebraically.	46
Exercise 3.8. An Alternative for Algebraic Solutions	46
Exercise 3.9. Equation Display and Storage	47
Exercise 3.10. Using LS DEF.	48
Exercise 3.11. Calculating Mound Temperature	53

Chapter 4 Basic Statistics

Exercise 4.1. Calculations Using RS STAT	58
Exercise 4.2. Calculations Using LS STAT	59
Exercise 4.3. Array Functions with RS STAT	61
Exercise 4.4. Array functions with LS STAT	62
Exercise 4.5. Curve-fitting with RS STAT	65
Exercise 4.6. Curve-fitting with LS STAT	67
Exercise 4.7. Predictions	69
Exercise 4.8. Curve-fitting With Equation Solutions	70
Exercise 4.9. Subdirectory Functions	76

Chapter 5 Basic Programming

Exercise 5.1. Examination Score	83
Exercise 5.2. Examination Score - A Better Idea	85
Exercise 5.3. Course Score I	87
Exercise 5.4. Course Score II	89
Exercise 5.5. Area of a Rectangle	92
Exercise 5.6. Calculating Volumes	95
Exercise 5.7. Calculating Volume - The Next Step	96
Exercise 5.8. VOL - Final Version	98
Exercise 5.9. Single-step Execution of a Program	100

Chapter 6 Programming Strategies

Exercise 6.1. Programming with RPN	102
Exercise 6.2. Data Entry with ENTER and INPUT	104
Exercise 6.3. Automatic Program Execution	105
Exercise 6.4. Programming Options	107
Exercise 6.5. Controlled Choices	108
Exercise 6.6. Addition with Conditional Branching	111
Exercise 6.7. Tests with Double Conditionals	113
Exercise 6.8. Tests with Many Conditions	117
Exercise 6.9. Controlled Looping	120
Exercise 6.10. "DO/UNTIL"	121
Exercise 6.11. Keeping in the Budget	123
Exercise 6.12. Guess Again	124

Chapter 7 Flags

Exercise 7.1. FLAG Test and Manipulation: The Basics	128
Exercise 7.2. FLAGS: A Simple Application	130
Exercise 7.3. Weight Predictions.	132
Exercise 7.4. Controlled FLAG Review	137
Exercise 7.5. Programmed Clearing	140

Preface

Why this book?

This book is written for those who want to take advantage of the newest and most powerful calculator/computer technology to improve math and programming skills and to apply them to practical problems. There is no better machine than the HP48G/GX for these purposes. It is powerful, inexpensive and with a little practice, intuitive and easy to use. Even so, many people find it difficult and frustrating to operate in the beginning. For some, basic math skills are poorly remembered, or perhaps inadequately learned in the first place. Others find it difficult to read and apply instructions that come with the HP48G/GX. All too many abandon their efforts because they cannot span the gap between existing abilities and those required to operate the HP48G/GX. This book provides a bridge to get beginners from the level of basic skills to the point of mastery of the instrument.

The book begins by introducing the keyboard of the HP48G/GX. It describes next how to construct alphanumeric symbols, then how to use and control the machine's memory for basic mathematical operations. As an intermediate stage, It describes how to use the instrument's powerful features for unit conversion, numerical and symbolic solutions of equations, how to compute basic statistics, how to perform curve-fitting tasks, and how to interpret these data. Final chapters describe how to write first simple, then more complex computer programs to solve equations and make decisions. The last chapter shows how to transfer files between machines in the HP48 family and between these instruments and IBM-compatible computers.

Step-by-step instruction is an important feature of this book. Exercises in each chapter give solutions to simple problems at first, then present progressively more complicated strategies for more complex and difficult tasks. There is a lot to be said for starting at the very beginning of the book and at each of its chapters and sections, even though the problems presented at first look easy and familiar. In this way, there is less chance of missing an important lesson in the sequence of incremental instruction.

Most readers will find the arithmetic and mathematical examples in this book ridiculously easy. Certainly, no one needs an instrument as powerful as the HP48G/GX to add a couple of numbers or calculate the area of a rectangle. But there is a reason for this simplicity. Calculations are purposely simple so that computing and programming strategies are more easily seen. Readers are encouraged to develop applications as complicated as needed for their own purposes once the basic computing and programming skills are learned.

It is the nature of learning to use technology like that of the HP48G/GX that mistakes will be made. Keystroke sequences sometimes will be incorrect and required characters will not always appear in the view window as easily as expected. There are many hints,

reminders and suggestions in each exercise to help get over the rough spots. Even with this help, though, errors are inevitable. Each mistake from which one learns a lesson in using the HP48G/GX, though, is a solid step forward in mastering a powerful and useful technology.

Readers will get the most from them if each exercise and demonstration is keyed into their own machines, and if they construct problems of their own using the same basic skills. Little-by-little, with no more than about a thirty minute daily investment, the power of the HP48G/GX will soon reveal itself. Even though the instrument is complex and might even be intimidating at first, a little practice soon makes it easy to use.

Why Learn about the HP48G/GX?

The benefits from learning how to use the HP48G/GX are enormous. No longer is there a problem in converting even the most intricate and unfamiliar sets of units. No longer are there obstacles for numerical or symbolic solutions of even the most complex equations. No longer do problems involving integration or differentiation pose difficulties and no longer is it difficult to write a computer program to meet one's own needs. The hardest part of learning how to acquire these skills with the HP48G/GX are predictably at the very beginning. Once the ice is broken, however, momentum and confidence soon build.

How is the HP48G/GX Similar to the HP48S/SX?

Superficial appearance is quite similar, but there are many differences under the keyboard. The HP48G/GX comes with many more built-in solutions for equations, a larger memory (512K ROM; 128K RAM), and more powerful graphing capabilities, among many other improved, simplified and expanded features. Both models, though, use the same programming languages, solve equations in about the same way, and require similar keystrokes for most functions. Those who are proficient with the HP48S/SX will be pleasantly surprised to discover they already know most of the things that have to be done to use the HP48G/GX, even though a few basic keystroke sequences will have to be relearned. View window displays for performing many operations are greatly improved, and are easier to read and use.

Chapter 1

In the Beginning...

Anyone who takes the HP48G out of the box for the first time must be impressed with the many functions shown on the keyboard. What some of them do is obvious. There's little doubt, for example, what the keys in the number pad do, or what the most likely actions are for the ENTER key, the ON key and a few others. What most of the other keys do, however, is much less obvious. The general impression for many people could well be one of confusion and possibly even intimidation - there is such a complex array. What does EQ LIB mean? What does ARG mean? What does << >> do? How come most of the keys have white letters on them, but also have colored labels above them? What does all this mean? Suppose I damage the machine by pressing the wrong button at the wrong time?

The purpose of this chapter is to introduce how the keyboard is arranged, how its functions are organized and how to use some of the computer's basic operations. It will also show that even though many functions are activated from the keyboard itself, however complex and complete it might appear, the strength of the HP48G lies in its many more operations not shown on the face of the instrument. Some will come from activating functions in menus, others from using those in subdirectories.

As you become more familiar with the instrument, you'll construct menus, directories and subdirectories to supplement those that are built-in. You'll be able to customize an already powerful machine into one that is even more useful for your own specific applications. Even so, that's not much consolation at the beginning when just looking at the keyboard is confusing. It'll make a lot more sense by the time you get to the end of this chapter. Using the keyboard will soon become second-nature as you progress from chapter-to-chapter, despite whatever confusion there is at the beginning.

Whether you use just the functions that come with the computer or make others of your own, there's no way short of gross abuse to damage the HP48G by pressing its buttons, no matter in what sequence. You may not get the calculation you expected, but you won't hurt the instrument. In fact, trial-and-error experiments are a good way to learn how this complex machine operates. It's a safe-enough adventure. Section 1.2. gives some useful hints about how to back out of any operations you may find yourself trapped in when you test different key operations.

Section 1.1. Keyboard Designations

The first step in using this book to learn about the HP48G is to understand the way in which its keys are identified. For example, the white letters embossed on each black key indicates its **primary function**. The action performed by the purple-colored letters is defined as the key's **left-shift function** designated in this text as "LS". The action performed by the green-colored letters is defined as the key's **right-shift function**, designated here as "RS". For example, for the first key in the first row of black keys, the primary function of the key is MTH, LS RAD indicates its left-shifted function, and RS POLAR indicates its right-shifted function. The first few sections of this book will also describe the location of some keys by count. For example, the phrase, "...the 4th key, 3rd row..." helps find the key to press for a square root function. A small left- or right-pointing arrow is displayed in the upper left corner of the view window whenever a shifted function is about to be activated. It's a useful reminder.

But there is another function for the keys in the upper half of the keyboard - they have a letter of the alphabet associated with them, as show to the lower right of each key. These letters are placed in the view window by using the alpha (α) function (primary function of the 1st key, 5th row). Figure 1.1. later in this chapter describes how this key is used in conjunction with others to construct alphanumeric and special symbols you'll need in calculations and programs.

So much for the black keys - how about the white ones at the very top of the keyboard just under the view window? They are used to select functions that will appear in the six blue rectangles at the bottom of the view window. Many different menus and "pages" of menus will be shown there from time-to-time. Using these six white keys is the way to activate much of the power of this computer.

For a simple demonstration of what the white keys do, turn the computer on (press the ON key at the lower left of the keyboard), then press the VAR key (4th key, 1st row). All the menu rectangles in the view window are blank. Press RS UNITS (shifted function of the "6" key). Each rectangle now designates a different subdirectory within UNITS, as coded with the small flag at the upper left of each rectangle. Access to each of these six subdirectories is controlled by pressing one of the corresponding six white keys. For an example, press the first white key (marked "A") to select the subdirectory entitled, LENG (for units of length). Units in this subdirectory of meter (M), centimeter (CM), millimeter (MM), yard (YD), foot (FT) and inches (IN) have now replaced the subdirectory titles. Key the number "4", then the "C" key to write "4 mm" in the view window. Erase this entry by pressing the key with the left-pointing white arrow (5th key in the 4th row). The entry you created is erased, but the menu selections remain. Press VAR to clear them.

The six white keys have another function besides selecting subdirectories and menu

options that appear at the bottom of the view window. They are used to write the alphabetical symbols A to F after the α key is pressed. Much more about this later in this chapter.

If you found the window displays hard to read when you completed this example, it's easy to adjust display contrast to be just right for you. Hold down the ON key while you press the + ("plus") key (lower right of keyboard) to increase contrast, or the - ("minus") key to decrease it.

Section 1.2. Backing Out of an Operation

Few people are so clever they will be able to execute the operations they need in the HP48G without ever making the mistake of pressing a wrong key or getting into a wrong subdirectory. Sometimes correcting such mistakes is easy, sometimes it's not so obvious how to do it. A couple of examples, though, will help.

With the computer turned on, press RS SOLVE (the right-shifted function of the "7" key). Once the SOLVE screen is displayed, pressing almost any black keys results in a warning beep indicating this is an inappropriate choice. There are two simple ways to get back where you started. Either press the white key marked "E" to select the menu option CANCL (for "cancel"), or press the ON key (marked "CANCEL" in white letters below the key). You're at the HOME menu again with a blank display.

As another example, press RS EQ LIB (right-shifted function of the "3" key). Although there is no CANCL function among the menu options, there is a choice of QUIT by pressing the "F" white Key. Pressing ON would have worked just as well. As a last example, press RS PLOT (the right-shifted function of the "8" key) to see the PLOT screen. Even though there is no CANCL or QUIT function in the menu bar, pressing the ON key gets you back HOME. One or more presses of the ON key will return the display to a cleared state.

Even though you're back to the HOME display, there are now the letters "PPAR" in the "A" position of the menu options. Pressing VAR doesn't clear it. It will have to be "purged". First, activate the left-shifted function of the + (plus) key by pressing LS { }. This places the symbols "{}" in the view window with a flashing arrow between them. Press the "A" white key to select the displayed menu option and to write "{PPAR}" in the view window. Purge this variable by first pressing ENTER, then use the left-shifted function of the EEX key (3rd key, 4th row) by pressing LS PURG. The view is now as it was when you started.

The ON key obviously controls several operations. It turns the computer ON, of course, it is also used to adjust display contrast, it cancels operations and its right-shifted function, RS OFF, turns the machine off. It is an important key for controlling operations during program execution and it is useful for some printing functions. More later.

Section 1.3. Writing Words and Numbers

Manual input and output of the HP48G, as it is with all other computers too, will be in words, numbers, mathematical symbols or similar notations used either singly or most often in some unique combination. Learning how to construct messages using these symbols is a good place to start putting the computer to your use. It's also a good introduction to how the keyboard is arranged and to how gain access to alphanumeric and special symbols. There are a lot of them, they are useful and they're easy to construct.

The next couple of sections, introduce the use of ALPHA symbols and special characters, most of which are not referenced on the keyboard. A few simple and easy-to-learn steps, though, will show how to take full advantage of them all.

Section 1.3.1. Generating ALPHA Characters

A useful feature of the HP48G is its ability to use alphabetic and number symbols in computations, in writing equations and in programming. Selecting number symbols is easy. Just use the number pad at the center bottom of the keyboard. Learning how to control the ALPHA features of the machine is easy too and it's time well-spent, but it takes a little more practice. Instructions for generating ALPHA symbols are summarized in Figure 1.1.

Figure 1.1 Guide for Selecting Alpha Characters	
Pressing	Produces
ALPHA (once only)	Upper case ALPHA symbol for next key press
ALPHA. ALPHA	Continuous generation of upper case ALPHA symbols until ALPHA or ENTER is pressed
ALPHA, LS	Lower case ALPHA symbol for next key press
ALPHA, RS	Special character for next key press
ALPHA, LS, ALPHA, ALPHA, ALPHA	Continuous generation of lower case ALPHA symbols until ALPHA or ENTER is pressed
ALPHA, ALPHA, LS	Lower case ALPHA symbol for next key press; upper case symbols for additional key presses
ALPHA, LS, ALPHA, ALPHA, ALPHA, LS	Upper case ALPHA symbol for next key press; lower case symbols for additional key presses. Press LS for next ALPHA symbol in upper case

The easiest and most direct ALPHA operation is to generate an upper case letter. Press the ALPHA key (note the "α" symbol at the top of the display), then press any key that has an alphabetic symbol at its lower right. The selected ALPHA symbol appears in an upper case format at the lower left of the view window. Pressing the same key again will not write another ALPHA symbol. But repeating the process of pressing the ALPHA key then choosing another alphabetic symbol adds a new ALPHA entry to the view window. A little experimentation soon shows that number symbols can be mixed with ALPHA symbols in any order. It will show also that each symbol in the string is erased one at a time by pressing the key with the left-pointing arrow (the ERASE function). After ENTER is pressed, all can be erased by LS DROP if only one line has been generated, or by multiple LS DROP operations or just LS CLEAR if there is more than one line in the view window. None can be eliminated just by turning the computer off. It has a continuous memory for all data entries whether they are alphabetic or numeric.

There are two ways to construct a string of ALPHA symbols, as you might want to in writing a word, without having to press the ALPHA key each time. One way is to hold the ALPHA key down as you select each alphabetic symbol. The other way is to press the key twice, as described in Figure 1.1. The ALPHA annunciator remains displayed at the top of the view window and each key press produces an alphabetic symbol until the ALPHA key is pressed again. Erasing a single symbol in a series is easily done with the ERASE key, but LS CLEAR no longer eliminates the entire string. It produces a special symbol (an exclamation mark) when ALPHA is still on. More later about how special symbols are controlled, but for now, how can the ALPHA string you've written be erased? One way is to press the ALPHA key (the annunciator disappears) and LS CLEAR works as before. Another way is to press ENTER (the ALPHA annunciator vanishes) and LS CLEAR works again. Also, pressing CANCEL clears the display.

As shown in Figure 1.1., single lower case letters are written by first pressing ALPHA, then LS. Pressing next the 4th key in the 2nd row then, for example, puts the symbol "p" in the view window. Creating a string of lower case letters without having to press ALPHA, LS each time requires the keystrokes, ALPHA, LS, ALPHA, ALPHA, ALPHA. The first three keystrokes give the instruction to write letters in lower case, the second two set the computer so it continues to write ALPHA characters. Pressing ALPHA or ENTER returns the computer to its original data entry mode. Sequences of upper and lower case letters can be mixed in any order. Start by pressing ALPHA, ALPHA, then press the SIN key (1st key, 3rd row) to generate "S". Pressing the LS key, then the SIN key once more produces "s". If you have keyed the sequence ALPHA, LS, ALPHA, ALPHA, ALPHA to write a string of lower case letters, pressing LS produces an upper case symbol. If you had just pressed ALPHA, ALPHA, LS produces a lower case symbol.



Figure 1.2. Characters 0-63

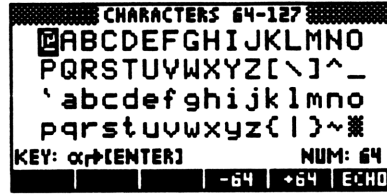


Figure 1.3. Characters 64-127

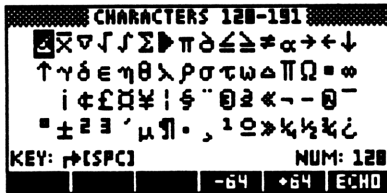


Figure 1.4. Characters 128-191

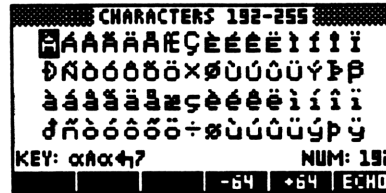


Figure 1.5. Characters 192-255

Section 1.3.2. Generating Special Characters

In addition to using standard alphanumeric symbols in your programs and equation solutions, the HP48G presents a wide range of special characters and symbols. They are shown in Figures 1.2. to 1.5. in the form of the screen displays. Produce one of them by pressing RS CHARS (the right-shifted function of the 2nd key, 1st row). There are several features of these displays that will help you select the character you want, place it in the view window, or scroll among the four displays for these special characters. View each of the other displays by pressing the white key marked "D" for the previous set, or the white key marked "E" to show the next set in the series.

As soon as you bring one of the these displays to the view window, one character will be highlighted automatically. In Figure 1.5, for example, it's the capital letter "A" with a small circle at its top. Use the cursor control keys (5th key, 1st row; 4th, 5th and 6th keys, 2nd row) to move the cursor to highlight any other character. Two pieces of information change each time you move the cursor. Keystroke instructions for generating the highlighted character are listed at the lower right of the view window next to "Key:", and the character number is listed at the lower right of the view window. Write the

selected character to the view window by selecting ECHO (press the white key marked "F"). To see the highlighted character to the view window and to clear the character menus, press NXT (6th key, 1st row), then the white key marked "E" to select CANCL, or just press CANCEL (the ON key). Pressing NXT brings the "next page" of menu options to the bottom of the view window. Some displays will have many "pages". This one just has two of them. Press NXT again to see the first one.

Section 1.3.3. Selecting How Numbers are Displayed

There are two decisions to make about how numbers are displayed in the view window or how they are printed. One is "*How many digits to the right of the decimal are needed*" and the other is "*What's the best format for display*". The decision about how many digits are displayed affects what is seen in the window, but it doesn't affect the accuracy with which a number is used in a calculation. For example, the number "1234.56789" can be displayed as such or rounded to a decimal as "1235.", seen as "1234.6", as "1234.57", as "1.235E3" or in a variety of other ways you can select. All calculations, however, will use "1234.56789" no matter what display option is selected. Calculations will use all numbers up to 12 digits to the right of the decimal.

It is an easy job to set the HP48G for exactly how you want to see numbers displayed and how you want the view window to be configured with other features. Press RS MODES (3rd key, 1st row) to bring the CALCULATOR MODES menu into display (Figure 1.6). This menu allows selecting the format for digit

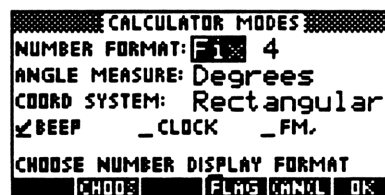


Figure 1.6. Modes Menu

display to either "standard", "fixed", "scientific" or "engineering", and controls how many digits will be shown to the right of the decimal. It also allow customization for angle measurements (degrees, radians or grads), coordinate system (rectangular, polar or spherical), enabling the "beep", displaying a "ticking clock", and whether fraction marks for a string of digits will be a comma (common in the United States) or a decimal (common in Europe).

The CHARACTER MODES display shown in Figure 1.6 has the cursor highlighting "Fix". To show other format styles, select CHOOS (for "choose") by pressing the white key marked "B". A new menu appears with one line highlighted (Figure 1.7). Up or down cursor keys control highlighting a different choice. Pressing ENTER returns the original display configured for whatever format was selected. How many digits will be shown to the right of the decimal is deter-

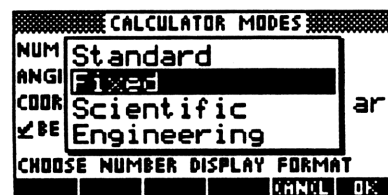


Figure 1.7. Format Menu

mined by highlighting the number to the right of the format selection, pressing CHOOS, highlighting a number and pressing ENTER. An alternative is to highlight the number next to the format choice, key any whole number from zero to 11, then press ENTER. The view window is cleared and the computer configured to the choices by selecting OK (white key marked "F"). When the format for number display is selected to be "standard", no choice can be made for the number of digits to the right of the decimal, because this choice shows all digits. Figure 1.8 compares number displays in different formats.

Figure 1.8 Number Displays			
Standard	Fix	Scient.	Engin.
0.01	0.01	1.00E-2	10.0E-3
0.12	0.12	1.20E-1	120.E-3
1.23	1.23	1.23E0	1.23E0
12.34	12.34	1.23E1	12.3E0
123.45	123.45	123E2	123.E0
1234.56	1234.56	1.23E3	1.23E3
12345.67	12345.67	1.23E4	12.3E3
123456.78	123456.78	1.23E5	123.E3
1234567.89	1234567.89	1.23E6	1.23E6
12345678.90	12345678.90	1.23E7	12.3E6
123456789.0	123456789.00	1.23E8	123.E6

To configure angle measure, coordinate system and other features, bring the display in Figure 1.6 to the view window (by RS MODES), highlight the feature to be changed using the cursor control keys, Key CHOOS to see options, then press ENTER and finally OK to record the selections. A short phrase just above the rectangular menu option boxes is a useful hint about the highlighted feature. "Page 2" of the CHARACTER MODES menu (selected by pressing NXT) provides the options to reset either a specific value or to return to the default settings for all choices. This page also allows canceling the current operation.

Section 1.4. Setting the Clock, Calendar and Alarms

The HP48G's clock can be used to control many functions. One of its most common is just to display the current time and date. Like buying a new watch, though, the computer's clock has to be set correctly at least once. It may also have to be adjusted once in a while if second-to-second accuracy is required. Clock, calendar and alarm functions are set and adjusted in similar ways using menus displayed by RS TIME. Figure 1.9 shows the opening menu.

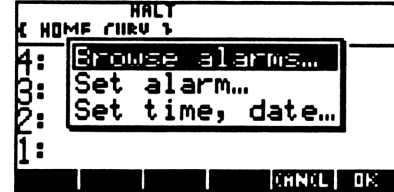


Figure 1.9. RS TIME

Highlighting "Set time, date..." and pressing ENTER presents the menu in Figure 1.10 with the cursor initially set for changing the clock hour, as indicated by the brief reminder at the lower left of the view window. Pressing CHOOS highlights a number for the correct hour. Pressing ENTER selects it and returns the view window to the previous display. Keying the number from the keyboard and pressing ENTER accomplishes the same task. NXT presents "page 2" with options to RESET, CANCL and other functions. Select the menu option OK on either "page" 1 or 2 to save the choices you've configured. Abandoning changes is easy with either CANCL or CANCEL. Displaying the date and time, as described in Section 1.3.3., is selected with options in the RS MODES menu (Figure 1.6).

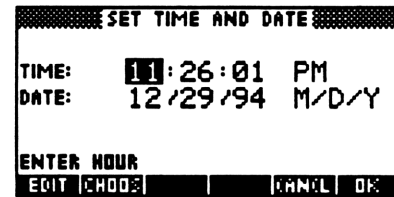


Figure 1.10. Set Time and date...

Setting one or more alarms begins by using the cursor controls to highlight the middle option in the menu shown in Figure 1.9, then pressing ENTER. If a message display is required, press α , α first, then use the ALPHA functions of appropriate keys to write a word or short phrase. Use the cursor controls to highlight the hour, minutes, seconds and date for the alarm to sound, and indicate whether the alarm will sound at successive times, and if so, at what intervals. Indicating times and intervals for alarms is made by highlighting "REPEAT", then pressing CHOOS. Select a number for the repeat sequence, press ENTER, then move the cursor to the right, and press CHOOS. Highlight the interval for the alarm and press ENTER. Pressing OK saves the settings you've constructed.

Setting one or more alarms begins by using the cursor controls to highlight the middle option in the menu shown in Figure 1.9, then pressing ENTER. If a message display is required, press α , α first, then use the ALPHA functions of appropriate keys to write a word or short phrase. Use the cursor controls to highlight the hour, minutes, seconds and date for the alarm to sound, and indicate whether the alarm will sound at successive times, and if so, at what intervals. Indicating times and intervals for alarms is made by highlighting "REPEAT", then pressing CHOOS. Select a number for the repeat sequence, press ENTER, then move the cursor to the right, and press CHOOS. Highlight the interval for the alarm and press ENTER. Pressing OK saves the settings you've constructed.

If an alarm is set to activate only once, pressing CANCEL terminates it. An alarm set to repeat must be erased with a purge function. To erase an alarm setting, press RS TIME, and with "Browse alarms..." highlighted, press ENTER. Use the white key marked "C" to PURG the highlighted alarm designation. OK returns the view window to the HOME display.

The next chapter gets down to business by explaining how arithmetic operations are performed and how units are designated and converted.

Chapter 2

Basic Operations

Chapter 1 described how to generate alphabetic and numeric symbols of one kind or another. How to use them in actual calculations will soon be clear and will become habit with just a little practice. The simplest calculations to start with are those involving the basic arithmetic operations of addition, subtraction, multiplication and division. The HP48G computer, along with others made by the same company, employs a special logic for performing these tasks when they are used as a single operation, when they are involved in complex strings of calculations and when they are used in programs. Its logic is that of RPN ("Reverse Polish Notation") which is based on a system developed by a Polish mathematician, Lukasiewicz, in the early 1920's, long before anyone envisioned such a sophisticated and compact computer as the HP48G. It's easy to learn, as you see in the next couple of pages, and offers many advantages in the direct and step-by-step way it deals with relationships between numbers.

Section 2.1. Number Control

When you first turn your computer on and press LS CLEAR, you see the name of the directory ("HOME") at the top left of the view window, and the numbers 1 to 4 displayed in a column at the far left (Figure 2.1). These numbers designate positions in the STACK. More numbered STACK locations will be generated when you ENTER more than 4 sets of numbers, as you'll see in some of the exercises to come, but these 4 will be enough to show some important basic operations. It is well worth the time to spend the next few minutes to understand how the STACK is organized and to see how numbers are positioned in it. This will show the basis not only for how all calculations are performed with the HP48G, but also it will also describe the organization of logic used in programs.

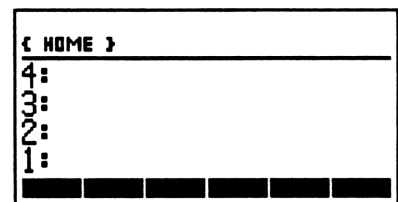


Figure 2.1. View Window

The HP48G computer operates on numbers in the STACK to add, subtract, multiply and divide them in a different way than most people are used to. Ask almost anyone to add, for example, the numbers 10 and 17. Good chance you'll hear something like, "Ten

plus seventeen equals twenty seven", a statement describing a sequence of 5 steps. Step 1: identifying the first number, Step 2: identifying what arithmetic operation will take place, Step 3: identifying the second number, Step 4: making an "equals" statement, and Step 5: stating the answer. A common strategy we are all used to using.

The HP48G performs arithmetic in a more direct and simpler way by using RPN. The first and second numbers are identified, then the arithmetic operation is stated. That's all. The answer is immediately available. A complex chain of calculations is performed in a similar way, operating on numbers a pair at a time. A simple example is shown in Exercise 2.1. Figure 2.2. shows the STACK display at each of the steps in this exercise. As you go through this exercise, notice how each number assumes different positions in the STACK.

Exercise 2.1. Basic "Reverse Polish Notation" (RPN) Operations

Problem Statement: Using only the STACK registers, calculate:

$$X = \frac{(30)(6) - (10+17)}{5}$$

Solution:

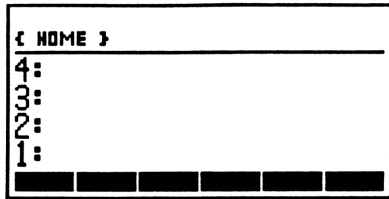
<u>Step</u>	<u>Press</u>
1.	30, ENTER, 6, *, 10, ENTER, 17, +, -, 5, ÷ (read: "30.60")

Working from an empty STACK (Step 1 in Figure 2.2.), keying "30" (Step 2) places it to the left of the view window at line 1. ENTERing it (Step 3) displays it to the right at line 1. Keying "6" raises the STACK to put "30" at line 2. Designating the arithmetic function of multiplication drops the stack and places the correct answer "180" at line 1. Keying "10" raises the STACK and ENTERing it places it to the right at line 1. Keying "17" raises the STACK again and Step 9 adds the numbers in lines 1 and 2.

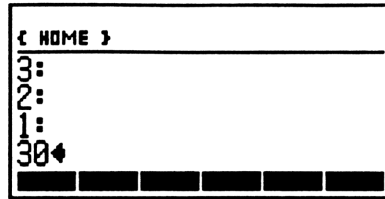
The numbers "180" and "27" are now in STACK positions 1 and 2 where Step 10 subtracts them. Keying "5" raises the STACK, with Step 12 providing the final answer at line 1.

Each time a new number was keyed, all previously displayed numbers were raised in the display. When an arithmetic operation was performed, the display was dropped one line with the answer shown at the end of each step in the first line of the view window. Each number was entered one-at-a-time into a STACK which went up or down depending on the nature of each one-at-a-time operation.

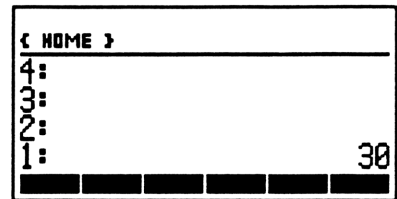
Figure 2.2 Display for Exercise 2.1.



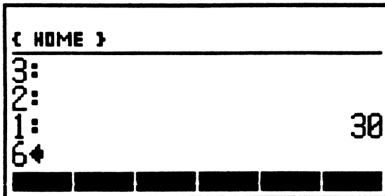
Step 1: Clear View



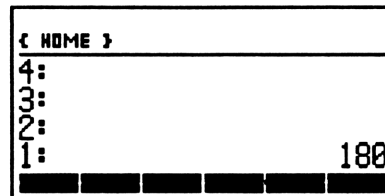
Step 2: key 30



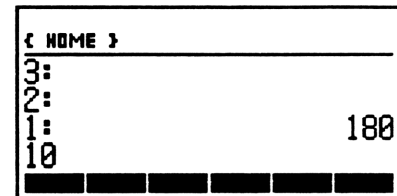
Step 3: ENTER



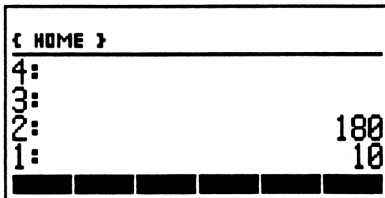
Step 4: key 6



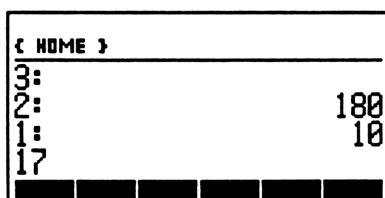
Step 5: Multiply



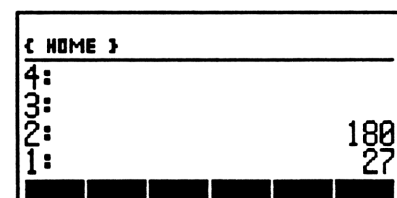
Step 6: key 10



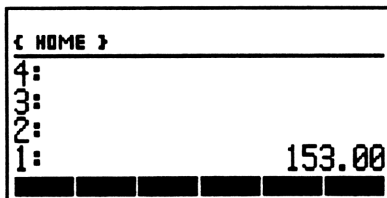
Step 7: ENTER



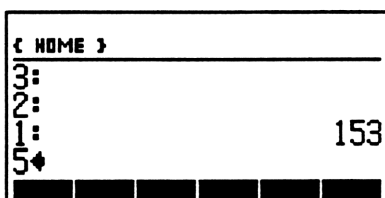
Step 8: key 17



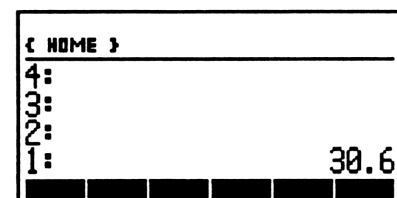
Step 9: Add



Step 10: Subtract



Step 11: key 5



Step 12: Divide

Many more numbers (up to the limit of the computer's memory) can be loaded into the STACK. Not all of them, of course, can be displayed in the view window at the same time. There is only room for four of them, but all of them remain in the STACK nonetheless. For an example, starting with a blank HOME display, key, then ENTER each of the numbers "100", "200", "300", "400", "500" and "600". Each of the first 4 numbers appeared in successive STACK positions and was shown in the view window. As soon as the digit "5" was pressed for keying "500", however, "100" disappeared from the top of the view window.

The number "200" vanished also with keying the last number in the sequence.

"100" and "200" remain in STACK locations 6 and 5, respectively, off-screen. To see them, press the UP cursor control (5th key, 1st row) until the screen scrolls down to show the first number entry. If pressed again, a tone indicates the cursor is at the top of the STACK. Pressing the DOWN cursor control (5th key, 2nd row) enough times until a tone is sounded returns the STACK to its original position. Pressing the ERASE key eliminates each number as it appears in the first line of the STACK. Pressing it several times eventually clears the STACK. Pressing CANCEL, then LS CLEAR would have cleared the STACK and the view window all at once.

You probably noticed that a menu automatically appeared in the view window as soon as you pressed the UP cursor control. Its options permit manipulating and viewing STACK contents in many ways. These will be explained in the context of specific applications later in the book.

Section 2.2. The Importance of STACK Position

Entering two or more numbers raised the STACK in Exercise 2.1. and performing an arithmetic operation lowered it. Not all functions affect the level or the order of STACK contents in these ways. Any number at line 1 in the STACK, however, will be changed in some way almost no matter what operation is performed.

To make the point with an example, key "1234" and press ENTER. Calculating the sine function of this number by pressing SIN changes the number at line 1, of course, but it doesn't alter the orientation of the STACK at any level. Had the cosine (COS) or tangent (TAN) functions of the number been calculated, had its reciprocal been computed with 1/X (6th key, 3rd row), or its square root taken, or had its sign been changed with +/- (2nd key, 4th row), the STACK would have responded in similar ways. Each operation would take place, but only the contents of line 1 would change.

Analogous operations affect the appearance of alphabetic symbols in line 1 in the STACK although, of course, arithmetic operations cannot be performed on them. For an example, press ALPHA twice, type the word "CAT", then press ENTER. The symbols 'CAT' appear at line 1. Keying a number (try "123") and pressing ENTER puts "123" at line 1 and raises the stack. The symbols 'CAT' now are listed at line 2. Erasing "123" with the ERASE key, or by keying LS DROP (5th key, 4th row), brings 'CAT' back to line 1. Pressing the SIN key constructs 'SIN(CAT)' as an ALPHA statement which can be further modified to read '-SIN(CAT)' if the "+/-" key is pressed, and changed again with a press of the "1/X" key to appear as 'INV(-SIN(CAT))'. Function keys (like "1/X", SIN, +/-, etc.) construct corresponding ALPHA amendments to non-numerical symbols appearing at line 1. Each would have performed its arithmetic function, of course, had there been a number at line 1.

Although it may be a little confusing at first, automatic changes either in the

orientation of the STACK or in a number at line 1 provide great power for making calculations, especially complicated ones. The number at line 1 and the one at line 2 combine in the way you designate when you add, subtract, multiply or divide them. The order of the numbers at these two lines is unimportant for addition and multiplication, since, for example, 5 plus 10 gives the same sum as does 10 plus 5, and the product of 12 times 30 is equal to that of 30 times 12. The order of numbers at lines 1 and 2 is critical for subtraction and division because, for example, 24 minus 16 doesn't give the same answer as does 16 minus 24, and 150 divided by 30 is not equivalent to 30 divided by 150.

Setting up or changing the STACK to position numbers where you want them for a calculation is easy. Starting with a clear HOME screen, to divide 150 by 30, for example, key 150, press ENTER then key 30 and press the division key to see "5" at line 1. Were 150 at line 2 and 30 at line 1 when 30 is to be divided by 150, pressing LS SWAP (6th key, 2nd row) reverses the contents of lines 1 and 2 and pressing the division key yields the desired calculation. Review Exercise 2.2. and Figure 2.3. to see how the contents of the STACK change in working through a complex calculation.

Before beginning Exercise 2.2, it will be useful to know that numbers at different levels in the STACK will have designations other than those of just the numbered levels shown at the left of the view window (Figure 2.1). Level 1 is also called the X STACK REGISTER and level 2 is also called the Y STACK REGISTER. Several functions performed directly from the keyboard require knowing this nomenclature. It will be important also when these functions are written into programs. For example, the primary function of the last key in the 3rd row ($1/X$) calculates the reciprocal of whatever number is at level 1 (the X STACK REGISTER), and the left-shifted function of this key raises to a natural log base whatever number is at level 1. Similarly, the primary function of the 4th key in the same row calculates the square root of the number at level 1. Its left-shifted operation squares the number there. Its right-shifted operation calculates the X'th root of Y, that is the number at level 1 provides the designation for the root and that at level 2 (the Y STACK REGISTER) is the number for which the designated root is determined. The primary function for the 5th key in the 3rd row (y^x) raises the number at level 2 to the power of the number at level 1, and its left-shifted function raises to a base ten the number at level 1.

There are a couple of places where there might be confusion about the X and Y designated positions of the STACK levels 1 and 2, respectively. The ALPHA function of the last key in the 3rd row ($1/X$), for example, is "X". This is an ALPHA character and has no implicit reference to the X STACK REGISTER. Neither does the "X" on the 5th key in the 6th row. It is, of course, the multiplication key. Its symbol has no reference to either an ALPHA character or to the X STACK REGISTER.

Now, on with Exercise 2.2. It is constructed to review basic RPN operations and to show how the X and Y designations of numbers at levels 1 and 2 in the view window

are important for using correctly keys that have these definitions. With not much more practice than given in the exercise, these different key definitions will soon become habit and commonplace.

Exercise 2.2. A Complex Calculation Using the STACK

Problem Statement: Solve the following equation using only the STACK:

$$X = \frac{\left(\frac{31.2}{9.6}\right)^{1.32}(\text{LOG } 4.9+(3.2)^2)}{(2.01)^3}$$

Solution:

<u>Step</u>	<u>Press</u>
1.	31.2, ENTER, 9.6, ÷, 1.32, y ^x , 4.9, ENTER, RS LOG, 3.2, ENTER, LS, x ² , +, *, 2.01, ENTER, 3, y ^x , ÷ (read: "6.38")

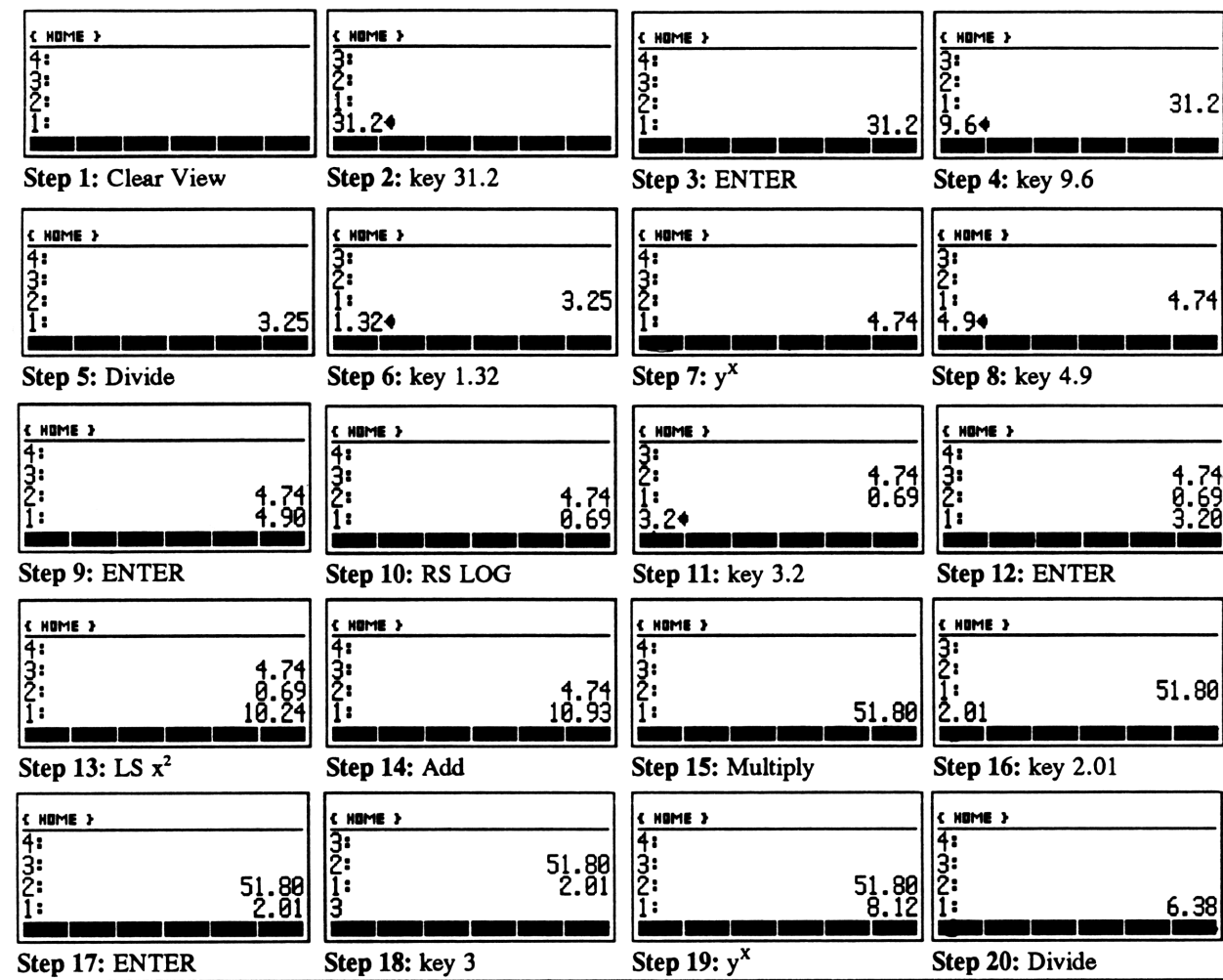
Section 2.3. Tools for STACK Control

Even the simplest examples presented so far make it clear that where numbers appear in the STACK is vitally important to the accuracy of a calculation. The sequence with which data are manually entered and the pattern of arithmetic operations are just two ways the STACK contents are ordered. There are many built-in functions for similar controls. They are important to manual operation of the computer, but they are essential in controlling the STACK during machine operations controlled by a program.

Access to the STACK comes in two ways. One is through the RS STACK operation (5th key, 1st row), and the other is by using LS STACK (not shown on keyboard; key LS then press the upward pointing cursor control key, the 5th key, 1st row). Each provides different operations related to the STACK, as described in Section 2.3.1. and Section 2.3.2. Some are useful for repositioning, duplicating and deleting numbers in the STACK. These will be described first. Others will be more usefully described later in the context of programming examples.

Before examining these operations, it will be useful to know about RS UNDO (right-shifted function of the 3rd key, 2nd row). It's a quick and easy way to recover from the results of unwanted operations. For an easy example, first press LS CLEAR, then key 1, ENTER, 2, ENTER, 3, ENTER, 4, ENTER. This "loads the stack". LS CLEAR, of course, removes all entries and "clears the stack". If this operation is judged to be a mistake, RS UNDO recovers from the error.

Figure 2.3. STACK Display for Exercise 2.2.



Section 2.3.1. Exercises for LS STACK Operations

The STACK contents can be cleared, reordered, duplicated and counted, as illustrated in Exercises 2.3. to 2.5., each of which begins with a cleared STACK.

Exercise 2.3. Clearing STACK Lines

<u>Step</u>	<u>Press</u>
1.	1, ENTER, 2, ENTER, 3, ENTER, 4, ENTER
2.	LS, DROP (erases number at Line 1. 4, ENTER returns original order.)
3.	LS, STACK, NXT, DROP2 (erases numbers at Lines 1 and 2. 3, ENTER, 4, ENTER returns original order)
4.	3 (for example), DRPN (erases numbers at Lines 1,2 and 3. 2, ENTER, 3, ENTER. 4, ENTER returns original order)
5.	LS CLEAR (erases entire STACK contents.)

Exercise 2.4. Reordering The STACK

<u>Step</u>	<u>Press</u>
1.	10, ENTER, 20, ENTER, 30, ENTER, 40, ENTER
2.	LS, SWAP (exchanges ("swaps") numbers at Lines 1 and 2. LS, SWAP returns original order)
3.	LS STACK, ROT (number at Line 3 is "rotated" to Line 1. Press ROT twice for original order)
4.	4 (for example), ROLL (number at Line 4 is "rolled" to Line 1. 4, ROLL 3 times for original order)
5.	3 (for example), ROLLD (number at Line 1 is taken to Line 3. 3, ROLL for original order), LS CLEAR erases STACK contents

Exercise 2.5. Duplicating and Counting the STACK

<u>Step</u>	<u>Press</u>
1.	100, ENTER, 200, ENTER, 300, ENTER, 400, ENTER
2.	LS, STACK, NXT, DUP (duplicates number at Line 1. LS, DROP returns original order)
3.	DUP2 (duplicates numbers at Lines 1 and 2. DROP2 returns original order)
4.	3 (for example), DUPN (duplicates numbers at Lines 1, 2 and 3. 3, DRPN returns original order)
5.	NXT, OVER (duplicates number at Line 2. LS, DROP returns original order)
6.	4 (for example), PICK (duplicates number at Line 4. LS, DROP returns original order)
7.	DEPTH (places the number of STACK entries at Line 1), LS CLEAR erases STACK contents

Section 2.3.2. Exercises for RS STACK Operations

In contrast to LS STACK operations, those involving RS STACK institute an "interactive STACK" and redefines the keyboard. Exercise 2.6. gives the basics. It makes sense that something has to be in the STACK for its contents to be arranged, so Exercise 2.6. starts by entering data into each of the first four STACK positions. This exercise uses keyboard controls in the first and second row for moving the cursor up, down, left or right, designated here as UC, DC, LC, and RC respectively.

Exercise 2.6. RS STACK Operations

<u>Step</u>	<u>Press</u>
1.	1, ENTER, 2, ENTER, 3, ENTER, 4, ENTER
2.	RS, STACK, ECHO, ECHO, ENTER, ENTER (copies the contents of the current level of the STACK, as indicated by the cursor position, to line 1; LS DROP, LS DROP returns original order)
3.	RS, STACK, UC, UC, ECHO, ENTER, ENTER (copies the contents of the current level (line 3) to line 1; LS DROP returns original order)
4.	3 (for example), PICK (copies contents of the numbered line to line 1; LS DROP returns original order)
5.	RS, STACK, UC, UC, LS, PREV KEEP (clears all STACK levels above the cursor; CANCEL, LS CLEAR erases remaining STACK contents)

Section 2.4. Calculations of Percent

Calculations involving percents can be made, of course, following the general procedures outlined in Exercises 2.1. and 2.2. just using the keyboard, but there is an easier way to do them. Menus available for you to make these calculations with fewer keystrokes are listed under the MTH directory. Figure 2.4. lists these menus.

Any calculation of percent requires 2 numbers. For example, asking, "*What is 5.83% of 12.92?*", or "*What is the percent change when 129.36 goes to 208.11?*", or "*What percent is 12.92 of 16.48?*" all inquire about a relationship between 2 numbers. Similarly, two numbers must be in the STACK when you ask your HP48G to calculate a percent, a percent change or to determine what percent one number is of another. Exercise 2.7. shows how to make these common and useful calculations. Before beginning the exercise, refer to Figure 2.4. to see that the menus "%", "%CH" and "%T" are listed in "page" 1 of the REAL subdirectory under the MTH main directory.

Figure 2.4. The MTH, REAL Menus

	Menu Keys					
page	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
1	VECTR*	MATR*	LIST*	HYP*	REAL*	BASE*
2	PROB*	FFT*	CMPL*	CONS*		

The "*" indicates a directory whose menus for PARTS are:

***REAL**

1	%	%CH	%T	MIN	MAX	MOD
2	ABS	SIGN	MANT	XPON	IP	FP
3	RND	TRNC	FLOOR	CEIL	D→R	R→D

Exercise 2.7. Percent Calculations

Part I. Problem Statements

What is:

- I.A. 9.68% of 14.37?
- I.B. 51.78% of 10.92?

Solutions (Part I):

	<u>Step</u>	<u>Press</u>
I.A.	1.	14.37, ENTER, 9.68, MTH, REAL, % (read: "1.39")
I.B.	1.	10.92, ENTER, 51.78, MTH, REAL, % (read: "5.65")

Part II. Problem Statements

What is the percent change when:

- II.A. 138.42 increases to 961.38?
- II.B. 1,427.11 decreases to 401.01?

Solutions (Part II):

	<u>Step</u>	<u>Press</u>
II.A.	1.	138.42, ENTER, 961.39, MTH, REAL, %CH (read: "594.55")
II.B.	1.	1427.11, ENTER, 401.01, MTH, REAL, %CH (read: "-71.90")

Part III. Problem Statements

What percent is:

III.A. 107 out of 392?

III.B. 1.5×10^{-2} out of 4.9×10^{-1}

Solutions (Part III):

	<u>Step</u>	<u>Press</u>
III.A.	1.	392, ENTER, 107, MTH, REAL, %T (read: "27.30")
III.B.	1.	4.9, EEX, 1, +/-, ENTER, 1.5, EEX, 2, +/-, MTH, REAL, %T (read: "3.06")

Section 2.5. Variables as Numbers and *Vice Versa*

It is common to need the same number more than once either in a single calculation or in a series of calculations. Even many inexpensive calculators allow you to store a number, then recall and use it whenever there is need. This saves both the number of keystrokes and the time, let alone the aggravation, to make a calculation. In general, the more complex the expression you store in this way, the more time and effort saved. The HP48G allows you to store more than just numbers, though. This section of the chapter introduces the advantages the HP48G gives you in being able to store complex variables, as well as just simple numbers, under names you construct. The process is easy to learn and handy to use.

Imagine that you need the number "1234.56" for a series of calculations. Starting with the computer at the HOME directory and FIXed to display 2 digits to the right of the decimal, making a calculation using 1234.56 is started by keying 1234.56, then pressing ENTER, keying another number, then pressing the "+" key to complete addition, for example. The number 1234.56 could be keyed again in subsequent calculations involving subtraction, multiplication and/or division. That would work well enough, but keying and entering 1234.56 each time is laborious and unnecessary.

An easier way to use a number repetitively is to store it, then recall it when it is

needed. The key VAR (4th key, 1st row) makes this process simple. Using it first requires identifying the number you need for the variable, then indicating a name for it, then storing it under that name. The number will be recalled whenever you ask for the variable by name or by just pressing the appropriately named menu key displayed by VAR.

For example, imagine that the number "1234.56" is going to be used many times in a series of calculations and it will be stored under the name "N" (for number). Key "1234.56" and press ENTER. Define the variable name by first using the apostrophe (') key (1st key, 2nd row), then press the ALPHA key, then press the "N" key (2nd key, 2nd row). Pressing STO (2nd key, 2nd row) stores 1234.56 under the variable name "N".

The menu of stored variables (in this example for the HOME directory) is seen anytime by just pressing VAR. Recalling 1234.56 to line 1 requires pressing the white key corresponding to the position in the menu where N appears. If is in the left-most position of the menu, pressing the white key marked "A" brings the numerical equivalent of the variable "N" into line 1 of the computer's display where it can be acted upon arithmetically. But you don't need to display the menus with VAR to get the number stored as "N". Just press the ' key, then ALPHA, then "N", and finally RS RCL (2nd key, 2nd row) to get 1234.56. Once it's at line 1, you can do anything you want to with it. If it is to be used often, the process for constructing a VAR menu for "N" makes a lot more sense than having to key it each time.

When the number 1234.56 has outlived its usefulness as a stored variable, it is easily purged from memory using the LS PURG function (3rd key, 4th row). To erase the computer's memory for 1234.56, press the ' key, then ALPHA, then N, then LS PURG. Pressing VAR to display the menu of stored variables shows that "N" is gone.

The VAR option of the HP48G computer gives many more valuable options than just storing and recalling a single number. When you've become more familiar with using VAR, it will allow you to save entire equations for future use, construct directories and subdirectories for data and equation storage, use VAR menus interactively within and between programs, among many other operations. These features will be explained later.

Section 2.6. Variables with Units

Using VAR menus for storing numbers is valuable. Additional power is available by using them to store variables with corresponding units. One advantage is that variables stored in this way can have unit conversions made on them automatically during arithmetic operations. This is a powerful feature and is well worth learning. Menus displayed under the directory RS UNITS allow manipulating a large variety of units in different ways. A simple example will show the basics. But first, an introduction to the directories and subdirectories and their menus displayed by RS UNITS.

There are many sets of units among which you can make conversions, as you will

see in Section 2.7. The easiest way to make them is to use the menu shown in Figure 2.5. You may have to hold onto your hat when you make the first couple of passes through the RS UNITS menus. They are complete and most valuable, but may be a little overwhelming at first. Exercises later in this chapter along with a little practice on your own will soon show that the RS UNITS menus are handy and not all that hard to use.

The directories and menus displayed with RS UNITS are so complex they deserve special explanation. Press RS UNITS to see the first "page" of directories, as shown in the first line of Figure 2.5. Press NXT twice more to see the next 2 "pages" of directories and press it again to redisplay "page" 1. It is clear that each option in each display shows a directory because each has a tab on its upper left corner.

Each of these directories contains sets of units appropriate for the physical measurement it designates. For example, the first directory (white key marked "A") on the first "page" of RS UNITS contains units for the measurements of length (LENG). Press A to see menu in its first "page" of M (for meter; position A), CM (for centimeter; position B), mm (for millimeter; position C), etc. Pressing NXT displays "page" 2 of the menu for this directory. The original display of directories is recaptured, of course, with RS UNITS.

One of the functions for the menu shown in Figure 2.5. is to amend a number at line 1 in the view window with an appropriate set of units. You may have already deduced how this might be done. To test your expectation: How would you construct the statement "55 mph" (miles per hour) in the view window? (Press: 55, ENTER, RS, UNITS, E (for SPEED), E (for MPH).) How would you show "10 acre" at line 1 in the view window? (Press: 10, ENTER, RS, UNITS, B (for AREA), NXT, F (for ACRE).) And a last one - How would you show "5.3 MOL"? (Press: 5.3, ENTER, RS UNITS, F (for MASS), NXT, NXT, B (for MOL).) As long as there are no "customized menus" stored in the HP48G, pressing CST at any time, of course, clears the menu display and LS CLR clears the display. By the end of this chapter, pressing CST will display the customized menu constructed in Exercise 2.11. More about that later.

So what? Is all of this a trick just to be able to display units with numbers? If it were, it certainly wouldn't be worth it in most people's opinion. But, there is much more to it. For one thing, each designated number with its set of units can be stored as a menu in VAR, much the same as you stored a number "N" a few pages ago in Section 2.5.

For an example, create and store the variable "1234.56" with units of seconds under the name "T" (for time). Here's how. Key 1234.56, press ENTER. To add units of time to the number which is now at line 1 in the view window, press: RS UNITS, D, E to add the units of seconds (s). Store the variable under "T" by first pressing ', then ALPHA, T, then STO.

Figure 2.5. The RS Units Menu

page	Menu Keys					
	A	B	C	D	E	F
1	LENG*	AREA*	VOL*	TIME*	SPEED*	MASS*
2	FORCE*	ENRG*	POWR*	PRESS*	TEMP*	ELEC*
3	ANGL*	LIGHT*	RAD*	VISC*		

The "*" indicates a directory whose menus are:

*LENG						
1	M	CM	MM	YD	FT	IN
2	MPC	PC	LYR	AU	KM	MI
3	NMI	MIUS	CHAIN	RD	FATH	FTUS
4	MIL	μ	A	FERMI		
*AREA						
1	M ²	CM ²	B	YD ²	FT ²	IN ²
2	KM ²	HA	A	MI ²	MIUS ²	ACRE
*VOL						
1	M ³	ST	CM ³	YD ³	FT ³	IN ³
2	L	GALU	GALC	GAL	QT	PT
3	ML	CU	OZFL	OZUK	TBSP	TSP
4	BBL	BU	PK	FBM		
*TIME						
1	YR	D	H	MIN	S	HZ
*SPEED						
1	M/S	CM/S	FT/S	KPH	MPH	KNOT
2	C	GA				
*MASS						
1	KG	G	LB	OZ	SLUG	LBT
2	TON	TONU	T	OZT	CT	GRAIN
3	U	MOL				
*FORCE						
1	N	DYN	GF	KIP	LBF	PDL
*ENRG						
1	J	ERG	KCAL	CAL	BTU	FT*LB
2	THER	MEV	EV			
*POWR						
1	W	HP				
*PRESS						
1	PA	ATM	BAR	PSI	TORR	MMH
2	INHG	INH20				
*TEMP						
1	°C	°F	K	°R		
*ELEC						
1	V	A	C	Ω	F	W
2	FDY	H	MHO	S	T	WB
*ANGL						
1	°	R	GRAD	ARCM	ARCS	SR
*LIGHT						
1	FC	FLAM	LX	PH	SB	LM
2	CD	LAM				
*RAD						
1	GY	RAD	REM	SV	BQ	CI
2	R					
*VISC						
1	P	ST				

You can see the variable named "T" at the far left of the VAR menu by pressing VAR. Each newly defined variable takes the left-most position at the bottom of the view window and the others are shifted to the right. Recalling the numerical value stored as "T" requires simply pressing ', then ALPHA, T, RS RCL, or by just pressing the white key designated A (if the menu "T" is at the far left) while viewing the VAR menu. If the menu option "T" is at some other location, press one of the keys A to F corresponding to its position. The number appears now with its unit designation. But that's not all. The best is yet to come.

Numbers with units can be arithmetically operated upon whether they reside in the view window at lines 1 and 2 (a standard RPN configuration for arithmetic operations, as shown in Figure 2.2.) or if they are stored in separate VAR menus. For example, adding 78.9 minutes to 1234.56 seconds (which is presumably still at line 1) requires keying 78.9, ENTER, RS UNITS, D, D. Pressing "+" adds these 2 numbers with their units to display "99.48 min". A little experimentation shows how numbers with units of length (LENG), AREA, volume (VOL), SPEED, MASS, FORCE, energy (ENGR), power (PWR), pressure (PRESS), temperature (TEMP), electrical units (ELEC), LIGHT and radiation (RAD) operate in similar mixed and matched ways. Arithmetic operations on units, of course, requires they are internally consistent. For example, you couldn't add 5.5 knots to 12.4 BTU's. Were you to try, a warning tone and the displayed "Inconsistent Units" tells you of the error.

A little experimentation also shows, for example, how a value of 2.54 cm can be added to that of 1 inch to provide an indicated sum of 2 inches. It will show also how a speed of 48 cm/sec (constructed by pressing: 48, ENTER, RS UNITS, LENG, B, RS UNITS, TIME, RS E) can be subtracted from that of 24 knots (constructed by pressing: 24, ENTER, RS UNITS, SPEED, F) then LS SWAP, - to calculate a difference of 1186.67 cm/sec, and how a weight of 70 kg can be added to 150 lbs to give a total of 304.32 lbs. The UNITS feature of the HP48G computer is more than a convenience for displaying appropriate units for a number. It is a valuable bonus for those who need a fast and accurate way to combine and convert internally consistent units, as described in the next section. The next section describes how to make yet another valuable use of the RS UNITS menus, that of unit conversions.

Section 2.7. Unit Conversions

It may just be the simple tasks of converting a temperature value in degrees Fahrenheit to one of degrees Celsius, converting a velocity expressed in miles per hour to one using units of kilometers per minute or seeing how a pressure value in inches of water equals one expressed in units of pounds per square inch. It could be the more complicated challenge of seeing how a thermal conductivity value expressed in BTU per hour per foot per degree Fahrenheit is represented when expressed in milliwatts per centimeter per

degree Kelvin. For any case, the processes of unit conversion by paper-and-pencil, even with a calculator, usually are not quick or easy for anyone. Unless there is a longstanding experience with particular sets of units, most of us depend on one or more reference books and have to take a lot more time and care than we'd like to get final values in the units we need. The HP48G comes to the rescue. Exercise 2.8. summarizes the steps in the next few examples.

Exercise 2.8. Sample Unit Conversions: Using Built-in Units**Part I. Temperature**

Problem Statement: Convert 37.84 degrees Celsius to degrees Fahrenheit, then to degrees Kelvin.

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 37.84, ENTER (read: "37.84") |
| 2. | RS, UNITS, NXT, TEMP, °C (read: "37.84°C") |
| 3. | LS, °F (read: "100.11°F") |
| 4. | LS, K (read: "310.99K") |

Part II. Speed

Problem Statement: Convert 36.54 miles per hour to meters per second.

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | 36.54, ENTER (read: "36.54") |
| 2. | RS, UNITS, SPEED, MPH (read: "36.54 mph") |
| 3. | LS, M/S (read: "16.33 m/s") |

Conversions among temperature scales, as shown in Part I of Exercise 2.8., is as simple as it gets. For the first example, 37.84 degrees Celsius is readily converted to 100.11 degrees Fahrenheit by first keying the number sequence then pressing ENTER. Press RS UNITS, then NXT, select TEMP on that menu "page" and press "°C" to give your number its units. Changing this value to °F is simply a matter of pressing first LS then the white key marked B. Changing it to °K requires no more than pressing LS then the white key marked C. Converting among units of speed (Part II, Exercise 2.8.),

follows about the same few keystroke patterns as does the temperature conversion, but, of course, the specific menu selections are different. The next few examples will show that conversions among other units is not much more difficult.

How to make conversions between sets of units which are not shown in the UNITS menus is summarized in Exercise 2.9. It requires becoming familiar with a menu displayed by the keystrokes LS UNITS. This menu is shown in Figure 2.6.

Figure 2.6. The LS UNITS Menu

	Menu Keys					
<u>page</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
1	CONV	UBASE	UVAL	UFACT	→UNIT	

Examples in Exercise 2.9. reveal some of the HP48G's valuable features for unit conversion using the LS UNITS menu. Keying the instructions to place "1.89 gal" in the view window are generally similar to those in used in Exercise 2.8. Keying the remainder of the phrase to display "1.89 gal/min" uses a couple of new procedures. As shown at Step 3 in Part I, constructing the denominator of the units statement requires pressing RS before pressing the appropriate menu key to select the units. A similar strategy is used in Step 3 of Part II to construct ".05 N/m²".

The other new technique is the conversion of the statement in line 2 of the STACK to the units listed at line 1. Two features of this technique are important. First, the number keyed at Step 4 (for both Part I and Part II) has no numerical value and will not be used in the conversion. The number "1" was arbitrarily chosen for these examples. Any other number would have served the same purpose. Second, the menu CONV needs to be selected from the LS UNITS directory.

The conversion of a value expressed, for example, in BTU per hour per foot per degree Fahrenheit to one in units of milliwatts per centimeter per degree Kelvin requires a few more keystrokes than the examples in Exercise 2.8., but the basic procedure is about the same. The number and its current set of units is written in line 1 of the view window using appropriate RS UNIT menus (Figure 2.5.). Next, the desired set of units (lead by an arbitrarily chosen number) is keyed followed by LS UNITS, CONV.

Only values with corresponding units, of course, can be converted in this manner. In the example, 1.89 "volume per time unit" was converted to a different "volume per time unit" set. Similarly in Part II of Exercise 2.8., 0.05 "force per area" units of Newtons per square meter were converted to a matched "force per area" set of units, dynes per square inch.

Exercise 2.9. Sample Unit Conversion: Mixing Built-in Units**Part I. Volume flow rate**

Problem Statement: Convert the value of 1.89 gallons per minute to liters per second.

Solution:

<u>Step</u>	<u>Press</u>
1.	1.89, ENTER (read: "1.89")
2.	RS, UNITS, VOL, NXT, GAL (read: "1.89 gal")
3.	RS, UNITS, TIME, RS, MIN (read: "1.89 gal/min")
4.	1, RS, UNITS, VOL, NXT, L (read: "1 l")
5.	RS, UNITS, TIME, RS, S (read: "1 l/s")
6.	LS, UNITS, CONV (read: "0.12 l/s")

Part II. Force per unit area

Problem Statement: Convert the value of 0.05 Newtons per square meter to dynes per square inch.

Solution:

<u>Step</u>	<u>Press</u>
1.	.05, ENTER (read: ".05")
2.	RS, UNITS, NXT, FORCE, N (read: ".05 N")
3.	RS, UNITS, AREA, RS, M ² (read: ".05 N/m ² ")
4.	1, RS, UNITS, NXT, FORCE, DYN (read: "1 dyn")
5.	RS, UNITS, AREA, RS, IN ² (read: "dyne/in ² ")
6.	LS, UNITS, CONV (read: "3.23 dyn/in ² ")

For the example in Exercise 2.9., Part I, how is 1.89 gallons per min (GAL/MIN) expressed in liters per second (L/S)? Starting at the HOME menu, key 1.89 and press ENTER. Press RS UNITS, select VOL, press NXT and select GAL. To construct this volume into a volume flow rate, press RS UNITS, select TIME, press RS and select MIN from the displayed menu. Constructing the desired set of units and making the conversion is straightforward too. Press 1 (or any other number), press RS UNITS, select VOL, press NXT (to get next menu), select L. To complete the rate statement, press RS UNITS, select TIME, press RS and select S (for seconds). The next step makes the conversion. Press LS UNITS and select CONV.

Running these examples and others of your own design a couple of times and using Figure 2.5. as a guide soon shows how combinations of a wide range of units of length, area, volume, mass, speed, power, in addition to others, are readily converted among themselves. It would be hard to argue with those who decide that just the unit conversions features of the HP48G makes the computer worth its money.

It is nothing short of a revelation to discover that burdens imposed by the tyrannical gods of unit designation have been lifted by the avenging HP48G. Freedom from confusion, hours of newly found time and a sense of confidence and wellbeing all come when one has learned the few basic and simple techniques shown in Exercises 2.8. and 2.9.

For the arrogant few who believe that unit conversions are trivial exercises, hardly deserving much attention for anyone well-versed in basic math - a challenge and a test! Using paper and pencil only, determine how many teaspoons per hectare per day are 5.3×10^3 stere per square foot per minute. Those with an HP48G will readily see from Figure 2.5. how just a few keystrokes solves the problem. It'll take the others a little time to catch up, so HP48G owners, start reading the next section that describes how to construct customized menus, a valuable feature of the HP48G.

Section 2.8. Customized Menus

Converting units for a single value, even if they are unusual and complicated, is easy with the UNITS menu. Exercises 2.8. and 2.9 show some simple examples. But the UNITS menu can be put to even better use as sets of units become more involved and more specific to a specialty area. They are still easily managed with a little care. Exercise 2.10. gives examples. There are several benefits for working through the different sections of this next exercise, even if the example units are not of particular interest or use. It will help develop familiarity with complex manipulation of the UNITS menu and it will provide a valuable background for learning how to construct customized menus addressed by CST. These skills will then allow constructing customized menus of one's own interest. It will soon become clear in reviewing Exercise 2.10. that it deals with units that are not listed in the UNITS menu. It will also soon become clear that not having the units immediately available doesn't matter, they are easily written anyway.

Exercise 2.10. Complex Units Conversions

Problem Statement No. 1: Convert $52.63 \text{ BTU}/(\text{ft}^2 \cdot \text{min})$ to Watts/in^2 .

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | 52.63, ENTER, RS, UNITS, NXT, ENRG, BTU, RS, UNITS, AREA, RS, FT ² , RS, UNITS, TIME, RS, MIN, 1, ENTER, RS, UNITS, NXT, POWR, W, RS, UNITS, AREA, RS, IN ² |
| 2. | LS, UNITS, CONV (read: "6.43 W/in ² ") |

Problem Statement No. 2: Convert 63.71 Watts/in² to Calories/(hr*cm²).

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 63.71, ENTER, RS, UNITS, NXT, POWR, W, RS, UNITS, AREA, RS, IN ² , 1, ENTER, RS, UNITS, NXT, ENRG, KCAL, RS, UNITS, TIME, RS, H, RS, UNITS, AREA, RS, CM ² |
| 2. | LS, UNITS, CONV (read: "8.49 KCAL/hr*cm ² ") |

Problem Statement No. 3: Convert 14.03 BTU/(ft²*min) to horsepower/ft².

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 14.03, ENTER, RS, UNITS, NXT, ENRG, BTU, RS, UNITS, AREA, RS, FT ² , RS, UNITS, TIME, RS, MIN, 1, ENTER, RS, UNITS, NXT, POWR, HP, RS, UNITS, AREA, RS, FT ² |
| 2. | LS, UNITS, CONV (read: "0.33 hp/ft ² ") |

Even the conversion of complex and perhaps unfamiliar sets of units is manageable with the UNITS menu and with using a few simple skills to construct appropriate statements at different lines in the view window. It would be laborious, however, to have to repeat all the different steps for the problem solutions in Exercise 2.10. were these unit conversions used frequently. It would be much easier, efficient and practical to construct a customized menu for them. Then, little more than a couple of keystrokes would provide the final unit conversion, no matter how complex it was. Exercise 2.11. shows how to create a customized menu for the conversions shown in Exercise 2.10.

Exercise 2.11. Creating and Using Customized Menus**Part 1. Creating a Customized Menu**

Problem Statement: Create a customized menu to convert values among the units of BTU/(ft²*min), Watts/in², horsepower/ft² and Cal/(hr * cm²).

Solution:

(**Hint:** In the following sequence, the symbol "_" is constructed by using the RS function of the multiplication key. The process of multiplication is symbolized by "*". "RC" indicates press the right cursor key (6th key, 2nd row)).

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, { }, 1, RS, _, RS, UNITS, NXT, ENRG, BTU, ÷, LS, (), RS, UNITS, AREA, FT^2, *, RS, UNITS, TIME, MIN, RC, SPC, 1, RS, _, RS, UNITS, NXT, POWR, W, ÷, RS, UNITS, AREA, IN^2, SPC, 1, RS, _, RS, UNITS, NXT, POWR, HP, ÷, RS, UNITS, AREA, FT^2, SPC, 1, RS, _, RS, UNITS, NXT, ENRG, KCAL, ÷, LS, (), RS, UNITS, TIME, H, *, RS, UNITS, AREA, CM^2, ENTER |
| 2. | LS, MODES, MENU, CST, STO (press VAR, then CST (keyboard)) |

Part 2. Using a Customized Menu**Problem Statements:**

1. Convert 52.63 BTU/(ft²*min) to Watts/in².
2. Convert 63.71 Watts/in² to Calories/(hr*cm²).
3. Convert 14.03 BTU/(ft²*min) to horsepower/ft².
4. Convert 4.56 horsepower/ft² to Watts/in².
5. Convert 16.92 Calories/(hr*cm²) to horsepower/ft².

Solutions:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 52.63, BTU/F, LS, W/IN (read: "6.43 W/in ² ") |
| 2. | 63.71, W/IN, LS, KCAL (read: "8.49 kcal/(h * cm ²)") |
| 3. | 14.03, BTU/F, LS, HP/FT (read: "0.33 hp/ft ² ") |
| 4. | 4.56, HP/FT, LS, W/IN (read: "23.61 W/in ² ") |
| 5. | 16.92, KCAL, LS, HP/FT (read: "24.52 hp/ft ² ") |
-
-

The customized directory designed in Exercise 2.11. was stored in the HOME directory. Pressing VAR shows this menu listed as CST among whatever other entries and subdirectories exist at the same time in the HOME directory. Editing the customized menu requires no more than pressing VAR, then CST (view window) and finally pressing LS EDIT. Deletions or additions to the customized directory are made by moving the arrow cursor to an appropriate location, then activating the appropriate operation. Pressing ENTER returns the edited statement to line 1 in the view window, then LS, MODES, MENU, MENU reconstructs the new version of the customized menu. If no changes are to be made after LS EDIT has been activated, just pressing CANCEL abandons the editing process.

Different customized menus can be constructed in any subdirectory. Erasing a customized menu requires first going to the subdirectory in which it exists, then pressing VAR. To delete the customized menu, press LS, { }, CST (view window), ENTER, LS, PURGE. Pressing the CST key now shows only blank rectangles at the bottom of the view window. Whatever customized menu was there before, no longer exists. (Hint: Don't erase the customized menu generated in Exercise 2.11. until Exercise 2.12. in the next section has been completed.)

Section 2.9. Arithmetic with Complex Units

Creating a customized menu, as demonstrated in Exercise 2.11. has more utility than just converting among sets of complex units, although that is certainly a valuable technique. Calculations can also be made among values using complex and customized units. This procedure is illustrated in Exercise 2.12.

Exercise 2.12. Calculations with Complex Units

Problem Statement: A temperature regulated machine had been modified over the years to have three different built-in heaters (A, B and C), each of which was separately regulated. These heating elements were obtained from different manufacturers and each is calibrated in different units. What is the total heat production of the heaters for Conditions 1, 2 and 3 shown in Table 2.1.? (Hint: Use the customized menu constructed in Exercise 2.11. to solve this problem.)

Table 2.1. Heater Element Output			
Heater	A (Cal/(hr*cm ²))	B (Watts/in ²)	C (BTU/(ft ² * min))
Condition 1	24.1	125.6	718.4
Condition 2	14.2	50.6	946
Condition 3	45.2	210.1	516.4

Solutions:

1. For "Condition 1":

Step Press
1. CST, 24.1, KCAL, 125.6, W/IN, 718.4, BTU/F, +, + (read: "3227.78 Btu/ft²*min")

<u>To Express in:</u>	<u>Press:</u>	<u>read:</u>
W/in ²	LS, W/IN	394.15 W/in ²
hp/ft ²	LS, HP/FT	76.11 hp/ft ²
Calories/(hr*cm ²)	LS, KCAL	52.33 kcal/h*cm ²

2. For "Condition 2":

Step Press
1. CST, 946, BTU/F, 50.6, W/IN, 14.2, KCAL, +, + (read: "36.34 kcal/(h*cm²)")

<u>To Express in:</u>	<u>Press:</u>	<u>read:</u>
W/in ²	LS, W/IN	272.66 W/in ²
HP/ft ²	LS, HP/FT	52.65 hp/ft ²
BTU/(ft ² *min)	LS, BTU/F	2232.89 Btu/(ft ² *min)

3. For "Condition 3":

Step Press
1. CST, 45.2, KCAL, 516.4, BTU/F, 210.1, W/IN, +, + (read: "612.30 W/in²")

<u>To Express in:</u>	<u>Press</u>	<u>read:</u>
HP/ft ²	LS, HP/FT	118.24 hp/ft ²
BTU/(ft ² *min)	LS, BTU/F	5014.25 Btu/(ft ² *min)
KCAL/(hr*cm ²)	LS, KCAL	81.61 kcal/(h*cm ²)

There are several ways to solve the problems presented in Exercise 2.11., each of which is equally valid. All of them would be awkward, frustrating and time-consuming with just paper-and-pencil, even for someone who is familiar with heat production units. All solutions are easy and accurate, however, for someone familiar with constructing customized menus and using them arithmetically. The operations described in Exercise 2.11 required no more than keying each value with its appropriate units, then adding several levels of the STACK. It is worthwhile to note that for each condition, the units for

the answer are provided in those for the last data entry. This feature makes it easy to get the answer in the units you desire. Just be sure they are the units for the last entry you make before doing the arithmetic.

Being familiar with the HP48G's menu structure and knowing how to create and use VAR options provides skills that go beyond just the management of units. They are the foundation for learning how to write and solve equations, as described in the next chapter.

Chapter 3

Writing and Solving Equations

Most calculators and computers do more than make just a single calculation. They are often used to make strings of calculations and to solve equations. In a typical application, you enter each number one at a time and perform each arithmetic or mathematical operation also one step at a time to yield eventually your final solution. Easy enough for simple calculations, but not good enough when an equation has many components or when it uses complex variables. Until now, solving such complex equations required either using expensive computer software or writing a program of your own. Times have changed.

The HP48G solves even the most complex equations without having to write a formal computer program. If you can write the equation on paper, you can key it into the computer in a matching form and solve it either numerically or symbolically for any one of its variables. The ease of entering the equation and solving it is a unique characteristic of the HP48G which makes it a powerful mathematical tool. Working through a few examples in this chapter will give you the basic skills for writing and solving equations of your own interest. In learning these techniques, you will see how the VAR menus described in Chapter 2 play an important role. As you complete each exercise in this chapter and have a good grasp of the fundamental operations it demonstrates, you will gain much more if you construct examples of your own which use them. The more examples you create, the bigger library of applications you will accumulate and the more easily the basic operations will be remembered.

Be prepared to be amazed and astonished. The HP48G has in its own memory (ROM) all the rules of algebra, mathematical logic and equation-arranging and solving skills you've struggled with over the years. For many, these skills may be rusty, for others they may be all but solidified. The HP48G provides the lubrication, though, to get it all working again better than ever.

Section 3.1. Writing an Equation and Solving it

There are several ways to enter an equation into the HP48G and to solve it. The following exercise shows one of the simpler techniques. A feature of this first exercise is to show how the equation is written into the computer in exactly the order you would write it on paper or state it in words. A second exercise (Exercise 3.2.) shows how similar steps are used to write and store a more complicated equation. Both exercises construct menus within the HOME directory. Notice as you work through each exercise that the word

"HOME" remains displayed at the upper left of the view window as a handy reminder of the name of the directory you are using. This status reminder will become more important as you construct and use subdirectories in Chapter 4.

If you make a mistake in keying any element of an equation, simply erase the incorrect entry with the ERASE key (5th key, 4th row), in the same way that you erased numbers and letters in earlier exercises. If you recognize an error in some previous step, use the cursor control keys to position the cursor at the miskeyed statement, use the ERASE key to eliminate it, then key the correct entry. Move the cursor back to where you were when you identified the error by using the appropriate cursor control keys, then continue constructing your equation. If attempts to key an equation result in a hopeless jumble, bail out of the equation writing mode by pressing CANCEL. Once you're back at the HOME menu, you can start over again to write the equation, or go on to something else. Sometimes just starting over saves time and aggravation compared to trying to patch up a bad job with extensive editing and multiple corrections.

Reminder: Keying equations and editing them requires using one or more of the cursor control keys. In this book, **LC** refers to the key that moves the cursor to the left (4th key, 2nd row), **RC** refers to the key which moves it to the right (6th key, 2nd row), **UC** refers to the one that moves it up (5th key, 1st row) and **DC** refers to the one that moves it down (5th key 2nd row). The symbol "*" indicates multiplication.

The first step in solving any equation is to write it so it shows precise relationships for its independent and dependent variables. Entering an equation in such a form into the memory of the HP48G is a straightforward procedure, as described in Exercises 3.1. and 3.2. The process uses the powerful "Equation Writer" functions.

Exercise 3.1. Writing and Storing a Simple Equation

Problem Statement: Write and store the equation: $X=A+2B+3C$

Solution:

<u>Step</u>	<u>Press</u>
1.	LS, EQUATION
2.	α , α , X, LS, =, A, +, 2, B, +, 3, C, ENTER
3.	', α , α , S, I, M, P, α , STO

The simple equation in Exercise 3.1. was easy to write as a step-by-step process into the view window of the HP48G, and it was just as easy to store it under the menu title "SIMP". In writing the equation, its elements were addressed in the same order you would use to describe the equation to someone. For example, reading the equation, "X equals A plus 2B plus 3C" follows the same sequence you used to write the equation for the computer. If you had chosen to state the equation as, "X equals A plus 2 times B plus 3 times C", you could have used those steps also in writing the equation. They would have been listed in Step 2 as: " α , α , X, LS, =, A, +, 2, *, B, +, 3, *, C". The expression would appear the same no matter which of these two expression styles was used.

It is explicit and clear to most people that the expressions in Exercise 3.1. "2B" and "3C" mean "2 times B" and "3 times C", respectively. It would not be so clear were the statement, for example, "2BC" to appear in an equation. Does this mean "2 times B times C" or does it mean "2 times BC"? If you'd be unable to select between these meanings when someone described the question to you, so would your HP48G be confused when you directed it to solve the equation.

The solution is simple. If the expression "2BC" means "2 times B times C", then key the expression as "2, B, *, C" or as "2, *, B, *, C". This is easy to remember because it follows the same order and rules you'd use to describe the equation in words. You'd state, "2 times B times C". If, however, the expression "2BC" means "2 times BC", then key it as "2, *, BC". This means that you want the computer to multiply 2 times whatever number is designated by the symbol "BC". The same clarity you'd guarantee with such statements is required when you write it as part of an equation.

Exercise 3.2. Writing and Storing a Complex Equation

Problem Statement: Write and store the equation:

$$Y = \frac{D^B}{\sqrt{F}} + \log(G)$$

Solution:

<u>Step</u>	<u>Press</u>
1.	LS, EQUATION
2.	α , Y, LS, =, α , D, y^x , α , E, RC, \div , \sqrt{x} , α , F, RC, RC, +, RS, LOG, α , G, RC, ENTER
3.	' , α , α , C, O, M, P, α , STO

The equation in Exercise 3.2. was a little more complicated than the one in Exercise 3.1., but with just a little familiarity with the sequential operations required for generating ALPHA symbols, using the cursor controls and function keys, its writing process is surprisingly easy. The next step after writing an equation is solving it. As for so many operations for the HP48G, there are several ways to do this. Exercises 3.3. and 3.4. introduce the different ways to use the SOLVE operation.

Exercise 3.3. Equation Solutions with RS SOLVE

I. The SIMP equation

A. Selecting and solving the equation

- | | |
|-------------|---|
| <u>Step</u> | <u>Press</u> |
| 1. | RS, SOLVE (see Figure 3.1.), OK, CHOOS, (use DC to highlight "SIMP"), OK. |

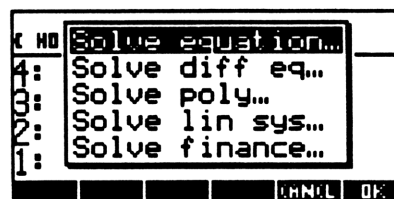


Figure 3.1. RS SOLVE

B. Solve for "X" when A=10, B=12 and C=14

- | | |
|-------------|--|
| <u>Step</u> | <u>Press</u> |
| 1. | (move cursor to "A:" by RC), 10, ENTER, 12, ENTER, 14, ENTER, DC, SOLVE (read: "X:76") |

C. Solve for "C" when X=15.6, A=17.1 and B=19.2

- | | |
|-------------|---|
| <u>Step</u> | <u>Press</u> |
| 1. | (with the cursor at "X:") 15.6, ENTER, 17.1, ENTER, 19.2, ENTER, SOLVE (read: "C:-13.30") |
| 2. | CANCEL |

II. The COMP equation

A. Selecting and solving the equation

- | | |
|-------------|--|
| <u>Step</u> | <u>Press</u> |
| 1. | RS, SOLVE, OK, CHOOS, (use DC to highlight "COMP"), OK |

B. Solve for "Y" when D=4.14, E=2, F=9.93, G=12.4

- | | |
|-------------|--|
| <u>Step</u> | <u>Press</u> |
| 1. | (move cursor to "D:" by RC), 4.14, ENTER, 2, ENTER, 9.93, ENTER, 12.4, ENTER, DC, SOLVE (read: "Y:6.53250...") |

C. Solve for "Y" when $F=30$ and other values remain unchanged

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | (move cursor to "F:" by DC, RC), 30, ENTER, UC, UC, SOLVE
(read: "Y:4.22267...") |
| 2. | CANCEL |
-
-

Exercise 3.4. describes a different way to solve the SIMP and COMP equations. The first step is to select the equation, the second step is to identify the solution method.

Hint: Letters designating elements in an equation when LS SOLVE is used are chosen from the menu inside the view window. Use the white keys at the top of the keyboard to make these selections. Ignore the letters A to F printed at the lower right of each white key. For example, in Step I.B.1. of the solution in Exercise 3.4., key 10, then press the second white key from the left to enter this number as "A" (as listed at the bottom of the view window). Key 12, then press the third white key from the left to enter this number as "B", etc.)

Exercise 3.4. Equation Solutions with LS SOLVE

I. The SIMP equation

A. Selecting and solving the equation

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, SOLVE (see Figure 3.1.), OK, CHOOS (use DC to highlight SIMP), OK, CANCEL |

B. Solve for "X" when $A=10$, $B=12$ and $C=14$

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, SOLVE, ROOT, SOLVR, 10 (Reminder: letters X, A, B and C refer to those shown at the bottom of the view window; selections are made using corresponding white keys) A, 12, B, 14, C, LS, X (read: "X:76.00"). |

C. Solve for "C" when $X=15.6$, $A=17.1$ and $B=19.2$

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | 15.6, X, 17.1, A, 19.2, B, LS, C (read: "C=-13.30") |
| 2. | CANCEL, VAR, LS, CLEAR |

II. The COMP equation

A. Selecting and solving the equation

Press

Press

1. RS, SOLVE, OK, CHOOS, (use DC to highlight COMP) OK, CANCEL

B. Solve for "Y" when D=4.14, E=2, F=9.93 and G=12.4

Step

Press

1. LS, SOLVE, ROOT, SOLVR, 4.14, D, 2, E, 9.93, F, 12.4, G, LS, Y
(read: "Y:6.53")

C. Solve for "Y" when F=30 and other values remain unchanged

Step

Press

1. 30, F, LS, Y (read: "Y:4.22")
2. CANCEL, VAR, LS, CLEAR

You may have noticed statements displayed at the upper left of the view window as you completed Exercise 3.4. They indicated each step in the process of solving the equation. The HP48G verified each number you entered for the solution of equations and indicated each variable. For example, when you began the solution at Step I.B.1. by keying "10", then pressing A, the message at the top of the view window confirmed that now "A: 10.00". When you keyed "12", then pressed B, "B: 12.00" appeared. Once all data had been entered and you completed the steps "LS, X", the machine indicated its current status was "Solving for X". Similar confirming status displays will appear as you solve other equations.

Another number besides the answer appeared when the last calculation was complete. The word "Zero" was displayed at the upper left of the view window. This indicated that a "root solution" or a final solution had been found for the equation. The "Zero" will appear again as solutions are made for most of the equations used in this book.

After each solution, the HP48G remembers how you have defined each of the VAR menus for the equation. To see a list of these definitions, along with those for all VAR designations in the HOME directory, press RS MEMORY, then use the up or down cursor control keys for the review. Press CANCEL, VAR, LS CLEAR to return to the opening screen. Additional VAR menus for X, C, B, A, among others, have been constructed. Another way to see how each VAR menu has been defined is to press the white key corresponding to a selected symbol. For example, Using the left or right cursor keys to bring the VAR menu "A" to the bottom of the view window, then pressing the white key

under it shows it has the definition "10", as used in Exercises 3.3. and 3.4. All VAR menus will remain in the menu list until you erase them. The next section shows how to erase unwanted menus.

Section 3.2. Cleaning-up VAR Menus

As you write more and more equations, the VAR menu in whatever directory you are working will list each of the titles you have constructed for them. It will list also each of the variables in each equation. These menus will remain until you clear them either selectively or as a group. This section shows how to eliminate menus displayed by VAR. The ones you have seen so far have all been in the single directory, "HOME".

You probably noticed in completing Exercises 3.3. and 3.4. that no variable name was repeated. Exercise 3.2., for example, used the symbols X, A, B and C, and Exercise 3.4. used the symbols Y, D, E, F and G. Had the symbol A, for example, been used in both exercises, errors would be likely if the number stored in A for the solutions in one exercise were accidentally and incorrectly used in the solution of a different equation. Menus within any one directory with the same title are used interchangeably. Operating in the HOME directory, for example, the computer is unable to distinguish between a stored value defined as "A" that you need for one equation from that stored as "A" required for the solution of some other equation.

There are a couple of ways around this problem. One way is to write equations in any single directory that do not use the same variable name. That's how the problem was solved in constructing Exercises 3.3. and 3.4. A somewhat awkward strategy, but it works. A more clever and practical solution is to write different equations in different subdirectories. Yet another solution is to erase VAR menus when they are no longer needed.

There are several ways to erase menus. One will be used to erase the menus X, A, B and C used for Exercise 3.3. Another will be used to erase the menus Y, D, E, F, and G associated with Exercise 3.4. The third way, described at the end of this section, is an alternative to either of these methods.

To try the first method, press VAR to display the menus for the HOME directory. Use NXT, if necessary, until you see the symbol "X" in the view window. Press the apostrophe key (1st key, 2nd row), then designate the menu "X" by pressing the white key under the view window corresponding to the menu titled "X". The symbol 'X' now appears at the lower left of the view window. Then key LS PURG to see the menu "X" disappear and to see remaining menus shift to the left to assume new positions in the menu display. To eliminate the menu titled "A", display it in the view window with NXT, if it is not already there. Then, press, ', use the appropriate white key to select the menu titled "A", then key LS PURG. Repeat these steps in an appropriate manner to eliminate the remaining menus B and C. The equation itself, listed as VAR menu "SIMP" is

eliminated in a similar way. Any other menu is erased in any directory in a similar way using the LS PURG operation.

Removing menus one at a time is practical if there are only a few of them to eliminate. If there are many menus to be erased, it is more expedient to list them as a group for elimination. Menus used for Exercise 3.4. will be erased in this way. First, place the set of pointed brackets (that is, "{ }") in the view window. These are selected using the left-shifted function of the plus key (5th key, bottom row). To put them in the view window, press LS { }. Use NXT to scroll through the displayed HOME directory menus until you find "Y". Select it for erasure by pressing the white key corresponding to its position in the view window. Press in order the white keys for menus D, E, F, G and COMP in a similar way. You may have to use either NXT or LS PREV to move the menu display back and forth to find the menu titles you need.

The view window now displays between pointed brackets ("{ }") the menus you selected. If you discover a mistake in the list, use the left and/or right cursor keys to position the arrow, then use the erase key to eliminate any selection. When your list is correctly constructed, press ENTER, then LS PURG to erase all designated menus. This process must be done with care, of course, because once menus are erased, they cannot be retrieved. RS UNDO returns the "{...}" statement, but not the individually defined VAR menus. They can be reconstructed, of course, but they cannot be resurrected once the LS PURG function has operated.

The alternative method for erasing VAR menus is to press RS MEMORY, use the down cursor control to highlight the object to be eliminated, then press NXT, and finally activate PURG by pressing the white key marked "B". As for the other two methods, once the PURG function is activated, there's no turning back.

Section 3.3. Making Changes in Stored Equations

It is a common task to have to amend an equation in some way after it has been written and stored. One reason might be there was an error in the way it was originally written. Another might be that it is easier to modify an equation already in memory than it is to write another one from scratch that is basically similar. This section describes how to amend existing equations using the function LS EDIT. Its operation is demonstrated in Exercise 3.5.

Exercise 3.5. Amending an Existing Equation Using EDIT

I. **Problem Statement:** Write and store the following equation as "TEST":

$$Z = 2A + \frac{4B}{3C} - D^{0.12}$$

Solution:

<u>Step</u>	<u>Press</u>
1.	LS EQUATION
2.	α , α , Z, LS, =, 2, A, +, 4, B, \div , 3, C, α , RC, -, α , D, y^x , .12, RC, ENTER
3.	' , α , α , TEST, α , STO

II. Problem Statement: Change the equation "TEST" to read:

$$Z=5A+\frac{4B}{3C}-D^{0.48}$$

Solution:

(Hint: The symbol " \leftarrow " means press the erase key (5th key, 4th row))

<u>Step</u>	<u>Press</u>
1.	VAR, TEST
2.	LS, EDIT, RC (4 times), \leftarrow , 5, DC, LC, \leftarrow , \leftarrow , 48, ENTER
3.	' , α , α , TEST2, α , STO

(Hint: Press VAR, then the white key at the far left to see the amended equation "TEST2". LS CLEAR clears the view window for the next exercise.)

Section 3.4. An Alternative Way to Solve an Equation

No matter how an equation is written, it expresses specific relationships among its different elements. Solving the equation by hand requires executing step-by-step procedures to act on each of these relationships. Operations using a computer program to solve the equation function in the same way by following sequential instructions to complete a sequence of actions. The program's algorithm is a set of instructions for the computer to use which are often about the same steps you would follow yourself if you were solving the equation with paper and pencil. The advantage of solving the equation with a computer program is, of course, operations are automatic. This saves time and effort, and guarantees greater accuracy. The program won't forget important steps, but you might.

If an equation is so much like a computer program in presenting sequential instructional steps, how come an equation can't be written in a VAR menu just like a program (or *vice versa*)? It can. Exercise 3.6. gives a simple example just to show how easy it is to do. How to write more complex programs is described later in the book.

Exercise 3.6. Writing a Program

Problem Statement: Write a program to evaluate:

$$\frac{A+B}{B^A}$$

Solution: Write the expression in a program and store it as "TEST3"

<u>Step</u>	<u>Press</u>
1.	LS, <<>>, ', LS, (), α , A, +, α , B, RC, \div , α , B, y^x , α , A, RC, ENTER
2.	', α , α , T, E, S, T, 3, α , STO

Problem No. 1: Evaluate TEST3 when A=0.12 and B=1.09

(Hint: It is easy to be confused in how the white keys (A to F) are used in this exercise. Exercise 3.4. described how to use them to select variables listed in the view window for an equation's solution. This exercise uses them in a different way because the menu labeled TEST3 contains a program for the solution of the expression, not just the equation itself. Step 1 in both Problems 1 and 2 in this exercise requires using the ALPHA defined white key "A" at the far left. Step 2 requires using the ALPHA defined white key "B" (second from the left). For this exercise and others with similar construction, the menus titled "A" and "B" inside the view window are ignored for data entry.)

Solution:

<u>Step</u>	<u>Press</u>
1.	0.12, ENTER, ', α , A, STO
2.	1.09, ENTER, ', α , B, STO
3.	VAR, TEST3, EVAL (read: "1.20")

Problem No. 2: Evaluate TEST3 when A=3.05 and B=4.93

Solution:

<u>Step</u>	<u>Press</u>
1.	3.05, ENTER, ', α , A, STO
2.	4.93, ENTER, ', α , B, STO
3.	TEST3, EVAL (read: "0.06")

The expression in TEST3 could have been written just as well using LS EQUATION followed by keying appropriate symbols similar to those described in Exercise 3.2. Writing it directly into a VAR menu as a program, though, requires only a couple of different steps. One of them is that line 1 in the view window needs first to be set for program writing. "LS, <<>>" accomplishes this. The apostrophe symbol (') indicates that elements of an equation will follow in subsequent keystrokes. The parenthesis symbols are used just as they would be for LS EQUATION constructions to separate elements in the equation so they are grouped appropriately for solution. Once the program is written, it is stored like any other one which would appear in line 1 of the view window.

Solving TEST3 requires first generating numerical values for variables A and B, then asking the program to evaluate them. The evaluation is initiated with the instruction EVAL. New values of A and B can easily be stored in the correspondingly titled VAR menus and TEST3 can be easily re-EVAL-uated. Editing TEST3 would follow steps similar to those described in Exercise 3.5.

Writing a program within a VAR menu to solve such a simple equation as TEST3 admittedly has limited practical value and it may seem just an awkward way to do things. Using strategies followed in Exercises 3.3. and 3.4. must seem more attractive at this stage. Exercise 3.6., though, is an easy introduction to program writing and it demonstrates important principles. You'll cash-in on having worked through this exercise when programming is presented in later chapters.

First, though, there is another way to solve equations, as introduced in the next section. All who struggle trying to remember some of the long-forgotten, half-remembered or infrequently used rules of algebra and mathematics will welcome with open minds learning this next set of HP48G operations.

Section 3.5. Solving Equations Algebraically

It may not always be necessary to enter an equation as a variable and solve it numerically as described in Sections 3.4. and 3.5. Nor will it always be the right choice to write a program for it, as described in Chapter 5. A convenient alternative is to solve it in its algebraic form. This section describes how to do that.

As for other operations, the HP48G offers alternatives for solving equations symbolically. Exercise 3.7. shows one way. Exercise 3.8. shows another. Exercise 3.9. presents useful tips for displaying equations in different formats and for storing a single equation under different VAR names and in different subdirectories. For purpose of illustration, Exercises 3.7. and 3.8. use relatively simple equations. Strategies used in these examples, however, apply just as well to more complicated equations. For many, using the HP48G to isolate an equation's variable will be one of its more important features.

Exercise 3.7. Solving an Equation Algebraically

Problem Statement: Solve the following equation for "B":

$$Z=5A+\frac{4B}{3C}-D^{0.48}$$

(**Hint:** This equation was stored as TEST2 in Part II of Exercise 3.5.)

Solution:

<u>Step</u>	<u>Press</u>
1.	RS, SYMBOLIC, DC, DC, DC (to highlight "Isolate Var..."), OK
2.	CHOOS, (use up or down cursor control keys to highlight "TEST2"), OK
3.	DC, α , B, OK, OK (read: 'B=(Z+D^0.48-5*A)*(3*C)/4')
4.	(To save solution as TEST4) ', α , α , T, E, S, T, 4, α , STO

Exercise 3.8. An Alternative for Algebraic Solutions

Problem Statement: Solve the following equation for "A"

$$B=\frac{(Z+D^{0.48}-5A)(3C)}{4}$$

(**Hint:** This equation was stored as TEST4)

Solution:

<u>Step</u>	<u>Press</u>
1.	TEST4, LS. SYMBOLIC
2.	', α , A, ENTER, ISOL (read: 'A=(Z+D^0.48-B*4/(3*C))/5')
3.	(To save as TEST5) ', α , α , T, E, S, T, 5, α , STO

It's hard to see how solving equations either numerically (Exercises 3.3. and 3.4.) or symbolically (Exercise 3.7.) could be any easier! It might be convenient, however, to see equations like TEST4 and TEST5 in some other format than that shown in Exercises 3.7. and 3.8. It may also be necessary to save them under different names without destroying their original addresses or VAR locations. Exercise 3.9. shows how to do this.

Exercise 3.9. Equation Display and Storage

Problem Statement: Display the equation stored as "TEST5" in "Equation Writer format and then store it also as TEST6.

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|-------------------------|
| 1. | VAR, TEST5, DC (read:) |

$$A = \frac{Z + D^{0.48} - \frac{B4}{3C}}{5}$$

- | | |
|----|--|
| 2. | CANCEL, CANCEL, ENTER |
| 3. | ', α , α , T, E, S, T, 6, α , STO |

Identical forms of the equation called TEST5 are now stored in VAR addresses TEST5 and TEST6. As described in the next chapter, the contents of a VAR menu can be transferred and stored in different subdirectories with about the same procedures as described in Exercise 3.9.

Section 3.6. The LS DEF Operation

There are many ways to write and use equations with the HP48G, as described in several sections of this book. They can be stored and used, for example, as VAR options, operated upon directly from the keyboard, built into other programs, solved with the SOLVE and SOLVR options, and in other ways too. Perhaps the simplest way to use an equation, though, would require just listing numbers for one or more of its variables in the view window, then with a single keystroke, solve the equation and see the answer in the view window. Structuring equations to be used in this way is easy with the LS DEF function (2nd key, 2nd row). It provides a direct and practical way to solve an equation with either single or multiple variables. Also, it requires no great insight into how programs are written, since the LS DEF operation will write the program all by itself and design it to accept input right from lines in the view window. Learning to use the LS DEF operation is a good introduction to equation and program structures.

Although writing complex equations into a customized program with structured alphanumeric data requests and answer displays is highly useful in many applications,

equation use through LS DEF is also worthwhile learning. Exercise 3.10. illustrates some applications for this useful option when an equation is written using the LS EQUATION feature.

Exercise 3.10. Using LS DEF

I. Solving an equation with a single variable

Problem Statement: Solve the following equation for values of "A" of 2, 3 and 4:

$$SOLA = \frac{\text{Log}(A)}{A^{0.12}} + 0.16$$

Solution:

A. Generate the program "SOLA" by:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, EQUATION, α, α, S, O, L, A, α, LS, (, α, A, RC, LS, =, RS, LOG, α, A, RC, ÷, α, A, y ^x , .12, RC, RC, +, .16, ENTER, LS, DEF |

B. Use the "SOLA" program. Press VAR first, then if digit display set for FIX 4:

- | <u>Step</u> | <u>Press</u> |
|-------------|--------------------------|
| 1. | 2, SOLA (read: "0.4370") |
| 2. | 3, SOLA (read: "0.5782") |
| 3. | 4, SOLA (read: "0.6698") |

II. Solving an equation with two variables

Problem Statement: Solve the following equation when "B" equals 19.36 and "C" equals 2.19. Solve it again when "B" equals -0.16 and "C" equals 0.73:

$$SOLB = B e^C$$

(Hint: Answers are displayed with 2 digits to the right of the decimal)

Solution:

A. Generate the program "SOLB" by:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, EQUATION, α , α , S, O, L, B, α , LS, (), α , B, SPC, α , C, RC, LS, =, α , B, LS, e ^x , α , C, RC, ENTER, LS, DEF |

B. Use the "SOLB" program

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 19.36, ENTER, 2.19, SOLB (read: "172.99") |
| 2. | .16, +/-, ENTER, .73, SOLB (read: "-0.33") |

III. Solving an equation with more than 2 variables

Problem Statement: Solve the following equation when "A", "B" and "C" are different for different calculations, but "A" and "B" are constants in the same series of calculations:

$$SOLC = \frac{A^{0.19}}{\ln(B)} - \left(\frac{A}{B}\right)^C$$

Problem No. 1: A=4.0; B=0.23

- Case 1: C=2.00
- Case 2: C=0.14
- Case 3: C=-0.60

Problem No. 2: A=3.26; B=15.17

- Case 1: C=0.09
- Case 2: C=0.21
- Case 3: C=0.32

Solutions:

A. Generate the program "SOLC".

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, α , α , S, O, L, C, α , LS, (), α , α , A, SPC, B, SPC, C, α , RC, LS, =, α , A, y ^x , .19, RC, ÷, RS, LN, α , B, RC, RC, -, LS, (), α , A, ÷, α , B, RC, RC, y ^x , α , C, RC, ENTER, LS, DEF |

B. Solve the sample problems

Problem No. 1, Case 1:

<u>Step</u>	<u>Press</u>
1.	4, ENTER, .23, ENTER, LS, STACK, NXT, DUP2, 2.0, ENTER, VAR, SOLC (read: "-303.34")

Problem No. 1, Case 2:

<u>Step</u>	<u>Press</u>
1.	←, LS, STACK, NXT, DUP2, .14, ENTER, VAR, SOLC (read: "-2.38")

Problem No. 1, Case 3:

<u>Step</u>	<u>Press</u>
1.	←,LS, STACK, NXT, DUP2, .6, +/-, ENTER, VAR, SOLC (read: "-1.07")

Problem No. 2, Case 1:

<u>Step</u>	<u>Press</u>
1.	LS, CLEAR, 3.26, ENTER, 15.17, ENTER, LS, STACK, NXT, DUP2, .09, ENTER, VAR, SOLC (read: "-0.41")

Problem No. 2, Case 2:

<u>Step</u>	<u>Press</u>
1.	←, LS, STACK, NXT, DUP2, .21, ENTER, VAR, SOLC (read: "-0.26")

Problem No. 2, Case 3:

<u>Step</u>	<u>Press</u>
1.	←, LS, STACK, NXT, DUP2, .32, ENTER, VAR, SOLC (read: "-0.15")

An important feature of programs generated by the LS DEF option is that an equation produced by the LS EQUATION option is written directly into the VAR menu for subsequent use, as shown in Exercise 3.10. This exercise also shows how data are taken directly from one or more lines in the view window for program input. This is a significant advantage, but it comes at the cost of having to remember how data are ordered at different lines when more than one input is used. It also requires remembering how multiple variables are placed and defined in the program and in the equation itself.

Constructing a program with LS DEF that is then selected as a VAR option requires that input variables be defined within parentheses immediately after the title of the program to the left of the equals symbol (see Step 1 in each of the solutions for Exercise 3.10.). The order in which these terms is stated establishes the sequence in which numbers must appear at different lines in the view window just prior to program execution. The symbol appearing first in the parentheses must be at line 1, the one appearing next must be at line 2, etc. Like any other program stored in a VAR menu, those constructed by LS DEF can be edited by first pressing the apostrophe key, then pressing the appropriate menu key, then LS EDIT. The cursor controls are then used to target sites for editing.

Programs generated by LS DEF appear only in the VAR menu for the directory which was active at the time of writing the equation. If it were written in the HOME directory, for example, only its own VAR menu would list it. If it were written in some subdirectory, only its own VAR menu would list it. The advantage is similar to that provided by constructing customized menus for unit conversions. In this way, a selected subdirectory can be constructed to hold programs for a particular series of calculations without cluttering or filling the VAR menu of the HOME directory or its subdirectories. For LS DEF generated programs, just as for the management of all VAR menus, their contents can be scanned by pressing RS MEMORY. Also, each or all can be erased by pressing LS, { }, selecting menu items to be eliminated by pressing each menu key for them, pressing ENTER, then LS PURG.

Section 3.7. A Special Feature For Special Cases

The HP48G has many hidden values and features. One is the way it solves equations. Although perhaps fortuitous, it is, nonetheless, of special value in obtaining numerical approximations for equations that cannot be solved explicitly. An example is perhaps the best way of demonstrating the uniqueness and utility of this strategy. What more appropriate example to use than one that is a unique strategy too? But, a little background is necessary.

The developing embryo or fetus of all warm-blooded animals (that is, birds and mammals) requires not only a mechanically protected and chemically controlled environment in which to mature, but also one that is thermally stable. For most mammals, this setting is the mother's uterus or pouch. For most birds, it is the nest where the fertilized egg is kept warm, moist and protected by one of the brooding parents, sometimes taking turns holding it close to their bodies.

The "Brush Turkey", a large Australian bird, uses a different technique. It scrapes nearly seven tons of debris from the forest floor into an approximately 400 cubic foot, about 3 foot high, mound in which it lays its eggs. Heat is produced in the mound by fungus and microbial action on the vegetation, and heat is lost from the mound by

radiation, evaporation, conduction and convection to the environment. Diligent and nearly constant attention by the parents in grooming and caring for the mound assures the stability of moisture and temperature for the incubating eggs buried in it.

As shown in Figure 3.2., there is a dynamic balance between rates of heat production and heat loss at some maintained level of heat flux and mound temperature. If mound temperature falls below an optimum level, microbial action slows, but the rate for heat production is still transiently higher than that for heat loss. This favors heat storage and the return of mound temperature to a higher steady state. Conversely, if environmental conditions are such that mound temperature rises above an optimum level, heat loss rate transiently exceeds that of heat production, and mound temperature falls.

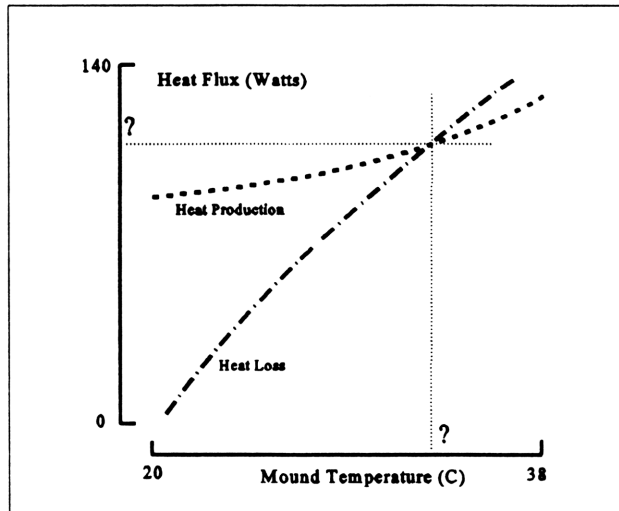


Figure 3.2. Nest Characteristics

These simple dynamics go a long way in providing a nearly self-regulating thermal environment for egg incubation.

Calculating the steady states for mound temperature and heat flux is not as easy as it might appear at first. Where the heat production and heat loss curves intersect in Figure 3.2 indicates the unique mound temperature and heat flux at which mound heat production and heat loss rates are equal. Assuming that at a steady state temperature (T ; °C), heat production rate (H_p ; Watts) is approximated by:

$$H_p = 55.44e^{0.021(T)}$$

and heat loss rate (H_l ; Watts) is approximated by:

$$H_l = 208.1 \ln(T) - 617.4$$

then, when $H_p = H_l$,

$$55.64e^{0.021(T)} = 208.1 \ln(T) - 617.4$$

This is an implicit equation that cannot be solved explicitly. It is impossible to isolate the variable "T" so the equation has the form: "T= ..." A value for mound temperature can be approximated, nonetheless, because the HP48G arrives at equation solutions through a trial-an-error, iterative process. Exercise 3.11 shows how.

Exercise 3.11 Calculating Mound Temperature

Problem Statement: Determine a steady state temperature (T ; °C) for the relationship:

$$55.64e^{0.02T} = 208.1\text{Ln}(T) - 617.4$$

Solution:

I. A Numerical Solution

A. Write the equation

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, 55.64, *, LS, e ^x , .021, *, α, T, RC, LS, =, 208.1, *, RS, LN, α, T, RC, -, 617.4, ENTER |
| 2. | ', α, α, N, E, S, T, α, STO |

B. Solve the equation

1. RS, SOLVE, OK (to select "Solve equation"), CHOOS, (place cursor on "NEST..."), OK, SOLVE (read: "T:33.26+")
2. CANCEL

II. A Graphical Solution

A. Write the equation

A.1. For heat production (HP)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, EQUATION, 55.44, *, LS, e ^x , .021, *, α, T, RC, ENTER
(read: " ' 55.44*EXP(0.02*T ' ") |
| 2. | ', α, α, H, P, α, STO |

A.2. For heat loss (HL)

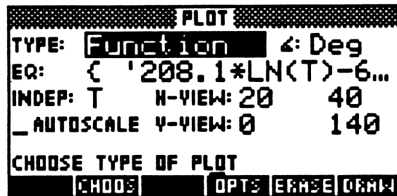
- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, 208.1, *, RS, LN, α, T, RC, -, 617.4, ENTER
(read: " ' 208.10*LN(T)-617.40 ' ") |
| 2. | ', α, α, H, L, α, STO |

B. Construct the graphic

Step

Press

1. RS PLOT ("EQ:" is highlighted), CHOOS, (highlight "HL"), ✓CHK, (highlight "HP"), ✓CHK, OK, DC (to highlight "INDEP:"), α , T, ENTER (to highlight "H-view"), 20, ENTER, 40, ENTER, RC (to highlight "V-view"), 0 (zero), ENTER, 140, ENTER (see "Display 1"), ERASE, DRAW

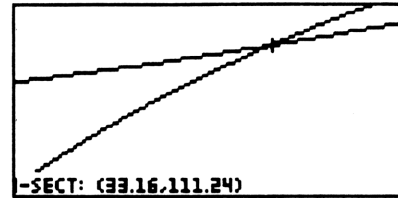


```

PLOT
TYPE: Function 4: Deg
EQ: { '208.1*LN(T)-6...
INDEP: T  H-VIEW: 20  40
AUTOSCALE V-VIEW: 0  140
CHOOSE TYPE OF PLOT
CHOOS  OPTS  ERASE  DRAW

```

Display 1



Display 2

2. TRACE, (X,Y), NXT, FCN, ISECT (see "Display 2"; read: "I-SECT: (33.16, 111.24)", CANCEL, CANCEL, LS, CLEAR

Solving either the equation for heat loss (H_l) or the one for heat production (H_p) numerically or graphically shows that when mound temperature is about 33.2°C , heat flux is about 112 Watts. Is it a biological and thermodynamic coincidence that, considering the metabolic heat production of the embryo itself, its temperature is likely to be close to an optimal level for incubation? Is it a coincidence also that the steady state heat production of the mound at about 33°C is approximately that of a resting, adult human? How did the Brush Turkey ever figure all this out without an HP48G? (For more information, see "The Brush Turkey", *Scientific American*, Dec., 1991)

Section 3.8. Taking Score and Looking Ahead

You have by now built your basic skills to an important level for using the HP48G. Applying these abilities to more complex mathematical operations is a reasonable and useful next step. The next chapter takes you to the next level of proficiency by introducing how the HP48G uses arrays to manipulate sets of numbers in an efficient and powerful way. Although this mathematical tactic may be new to many, it is easy to learn. You will soon become used to its notation and how it is applied. It will be introduced in the context of calculating simple statistics, first on a single column of numbers and then on columns of 2 or more data sets.

Chapter 4 also introduces how equations are written for different sets of numerical data and how such curve-fitting gives valuable insight into relationships between

variables. Once the equations are deduced, you will then see how to use the HP48G to plot them and how to use these on-screen graphs for graphical solutions, much like the example shown in Exercise 3.11.

If you have taken advantage of the principles demonstrated so far in this book's exercises, it is likely you have written and solved many equations of your own. Either your HOME directory is getting rather crowded, or you're getting tired of having to PURG menu options from it to make room for others. Chapter 4 introduces how to generate an equation and program filing system using subdirectories. You'll soon see how your imagination is the only required tool to customize your HP48G computer to your specific needs.

You are at an important transition in using the HP48G. You have learned many of its basic operations for controlling data processing as well as those for equation writing, editing and use. You are now in the enviable position of putting all of this to work for you. You are about to see the HP48G's power open up and turn on its afterburners.

Chapter 4

Basic Statistics

Engineers, physicians, people in business, scientists, students and many others use statistics of one form or another to describe the work they do. An application might be as simple as calculating the average of a group of numbers. It might be more complex as in calculating variances within a data group or as detailed as determining confidence limits among data or computing trends for time varying functions. Regardless of the level of complexity, the HP48G's built-in programs make many statistical calculations straightforward.

This chapter has several goals. One is to introduce the HP48G's statistical menu (STAT) and show how it is used for making several different kinds of useful calculations. It will be used, for example, for computing the average (MEAN) and standard deviation (SDEV) of groups of numbers. This menu is also used for deriving different types of equations for sets of data and for determining their validity. Basic to these operations, this chapter describes how to construct and edit data in the forms of an array and how to present on-screen graphics for different functions. You will also see how to calculate predictive values from a set of data and evaluate relationships between interdependent variables.

There is another goal too. To help organize the growing number of equations you are now writing and using, sections at the end of this chapter show how to construct subdirectories for their storage. This organizational strategy has several useful spinoffs. Not only will each working directory be less cluttered and easier to use, but also each will allow you to use similarly defined variables among the different subdirectories. Also, a well thought-out directory structure for your different programs, equations and calculations will help greatly in accessing them quickly and simply. But the next best step is to learn how to enter data and evaluate it using options through the STAT menus. The following section begins the process.

Section 4.1. Evaluating a Single Column of Numbers

A simple and commonplace statistical analysis is just to calculate the average (MEAN) of a column of numbers. Using RPN techniques described in Section 2.2., you could use just the HP48G's logic to work with numbers stored in the computer's STACK to calculate their average. For example, starting with an empty STACK (obtained by LS CLEAR), one way to calculate the average for a set of data is simply to add each number in the series then divide by the number of entries. For example, key 45.23, ENTER, 17.94, +,

3.47, +, 12.91, +, 4, ÷ shows 19.89 in the view window as the average of this series.

Other statistics for a group of numbers like, for example, their standard deviation, could be computed in a similar (but laborious) way with RPN operations using numbers in the STACK. There are much easier ways to do all this and many other calculations too using RS STAT and LS STAT functions. Predictably, the HP48G provides options for calculating basic statistics. RS STAT presents the opening display shown in Figure 4.1. LS STAT generates a set of menus shown in Figure 4.2.

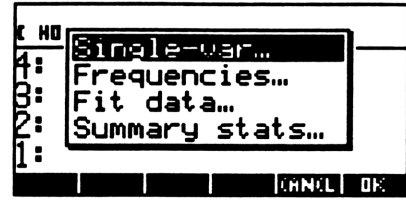


Figure 4.1. RS STAT

Figure 4.2. The LS STAT Menu

	Menu Keys					
page	A	B	C	D	E	F
1	DATA*	Σ PAR*	1VAR*	PLOT*	FIT*	SUMS*
The "*" indicates a directory whose menus are:						
*DATA						
1	Σ +	Σ -	CL Σ	Σ DAT		STAT*
* Σ PAR						
1	XCOL	YCOL	MODL*	Σ PAR	RESET	INFO
2						STAT*
*MODL						
1	LINFI	LOGFI	EXPFI	PWRFI	BESTF	Σ PAR*
*1VAR						
1	TOT	MEAN	SDEV	MAX Σ	MIN Σ	BINS
2	VAR	PSDEV	PVAR			STAT*
*PLOT						
1	BARPL	HISTP	SCATR			STAT*
*FIT						
1	Σ LIN	LR	PREDX	PREDY	CORR	COV
2	PCOV					STAT*
*SUMS						
1	Σ X	Σ Y	Σ X ²	Σ y ²	Σ X*Y	N Σ
2						STAT*

Exercise 4.1. demonstrates calculations using RS STAT. Exercise 4.2. shows how similar calculations are accomplished using the LS STAT menus. A little practice with each of these techniques will soon show which is preferred. Be prepared to see that RS STAT may be the best for some problems, and LS STAT works easier for others.

Menus shown in Figure 4.2. are used to control many statistical and curve-fitting operations. Examples in Exercise 4.2. describe first how to obtain basic statistics on a single column of numbers. Later sections explain how to make similar calculations on more than one set of data and introduce curve-fitting techniques.

Exercise 4.1. Calculations Using RS STAT

Problem Statement:

Body weights were measured for five small animals, as shown in Table 4.1. What is the average, minimum, maximum and total weights for the group, and how do calculations of standard deviation and variance indicate the distribution of individual weights?

No.	Wgt. (oz.)
1	6.3
2	7.8
3	9.4
4	5.7
5	4.9

Solution:

A. Enter Data

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, STAT, OK (to select "Single Var..."), EDIT, 6.3, ENTER, DC, 7.8, ENTER, 9.4, ENTER, 5.7, ENTER, 4.9, ENTER, ENTER |

B. Select Calculations

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | DC, DC, LC, ✓CHK, RC, ✓CHK, RC, ✓CHK, RC, ✓CHK, RC, ✓CHK, RC, ✓CHK, OK |

C. Read Solutions

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | UC (6 times to see "line 6"; read: "Mean:6.82, Std. Dev: 1.79, Variance:3.21, Total: 34.10", DC (5 times to see "line 1"; read: "Maximum: 9.40, Minimum: 4.90"), CANCEL, LS, CLEAR |

Exercise 4.2. Calculations Using LS STAT

Problem Statement:

Temperatures were measured for five locations in a container of viscous material that was presumably thermally uniform. Data are shown in Figure 4.2. What is the average and standard deviation for these measurements?

No.	Temp.(°C)
1	46.3
2	52.1
3	47.6
4	49.0
5	51.7

Solution:

- | Step | Press |
|------|---|
| 1. | LS, STAT, DATA, CL Σ , 46.3, Σ +, 52.1, Σ +, 47.6, Σ +, 49, Σ +, 51.7, Σ +, STAT, 1VAR, MEAN (read: "49.34"), SDEV (read: "2.53") |

(Hints: 1. "CL Σ " at Step 1 cleared the statistical register before the temperature data were entered. This was essential for Exercise 4.2. because the weight data were already resident from Exercise 4.1. Had "CL Σ " not been used, the temperature data would have been appended to the existing weight data. It's a good idea to use "CL Σ " before each new set of statistical calculations. Exercise 4.3. shows how to do this for RS STAT operations.

2. If an incorrect value had been entered with " Σ +", key the same value, then press " Σ -" to remove it from the statistical register. Continue with " Σ +" to enter new data.)

Other calculations can be made on data in the statistical register besides those demonstrated in Exercises 4.1. and 4.2. Using LS STAT, "SUMS" (selection "F" in the opening menu display shown in Figure 4.3.) generates additional menu selections for calculating ΣX , ΣY , ΣX^2 , ΣY^2 and $\Sigma(X*Y)$, as well as for showing the number of data entries ($N\Sigma$). Using RS STAT, move the cursor to "Summary stats..." in the opening display (Figure 4.1.), then press "OK". Next, move the cursor to the 4th row of the new display (Figure 4.3.) and use " \checkmark CHK" to select the required summary statistic. "OK" makes the calculation.

```
SUMMARY STATISTICS
ΣDAT:
X-COL: 1  Y-COL: 2
CALCULATE:
- ΣX - ΣY - ΣX² - ΣY² - ΣXY - NΣ
ENTER STATISTICAL DATA
EDIT CHOOSE  CANCEL OK
```

Figure 4.3. Summary Stats

Main points to learn from Exercise 4.1. are how to enter data for statistical and curve-fitting analyses, and how to use menus on the different "pages" of the STAT register to make each calculation. People who have struggled through computations like this with only a hand calculator, or even worse, a slide rule in days of yore, will soon see the powerful advantages they have with the HP48G. People who don't have this questionably useful background still have the advantage of the HP48G, of course, but not the mathematical scar tissue.

Section 4.2. Editing Stored Data

Data entered into the statistical registers remain in the computer even if you erase the display of the STAT menus by LS CLEAR, clear the view window by CANCEL, or use the computer for almost any other calculations. They remain stored even if you turn the computer off. Stored data can receive additional entries (by keying LS STAT, DATA then the new number, then pressing $\Sigma+$) or they can be edited or erased selectively, as described next.

It's easy to change any data entry in the statistical register whether either RS STAT or LS STAT operations are underway. Exercise 4.2. described how to correct an erroneous entry at the time of placing data into the statistical register. Making changes at any other time is just as straightforward.

Press RS, STAT, OK, EDIT, then move the cursor to the cell whose number is to be changed. The row number, column number and the value for the highlighted data are shown at the lower left of the view window. Key the new number, then press ENTER. Pressing ENTER again returns the RS STAT display. The new number has been stored. Alternatively, press LS, STAT, DATA, Σ DAT, DC to display the data table. Move the cursor to the target cell, key the new data entry and press ENTER twice. LS, CLEAR clears the STACK.

Section 4.3. Using Two Columns of Numbers

Although the operations in Exercise 4.1. and 4.2. are useful for demonstrating basic steps for data entry, calculations and editing, they come nowhere near the potential of the HP48G to make statistical analyses. The next step will show how data are entered as paired columns into the statistical registers. They are then available not only for the calculation of statistical characteristics of both sets (as in Exercises 4.1. and 4.2.), but also for finding relationships between the data sets. As examples, this will be done by calculating a correlation coefficient, by defining an equation which best-fits these data as functions of one another, by graphing the data and by displaying the best-fit equation within the graph. These are powerful statistical procedures that have many practical applications.

Section 4.3.1. Basic Calculations

Exercises 4.3. and 4.4. show how basic statistical calculations are performed on columns of matched data. For these examples, data are entered as an array with two columns. Others, of course, might have 3, 4 or more columns. An array is simply an ordered arrangements of mathematical elements constructed in rows and columns. The general structure is no more than the familiar spreadsheet of data. For the examples here, the elements of the arrays are numerical data. An advantage of entering data in this way is that

calculations can be performed simultaneously for the data set. Also, relationships can be defined for data between columns. This gives unique opportunities for curve-fitting of data of constructing graphs to gain insight into them. A couple of examples will make all this clear. Exercise 4.3. demonstrates how functions provided by RS STAT operate on a data array.

Exercise 4.3. Array Functions with RS STAT

Problem Statement:

A chemist determined that the setting time for an epoxy cement being developed depended on how much catalyst was used. Data are shown in Table 4.3. What are the average catalyst weight and setting time for this test, and how did these observations vary, as indicated by the calculation of standard deviation?

Mix No.	Catalyst (grams)	Set Time (hr)
1	25.4	1.64
2	29.3	2.28
3	32.6	2.75
4	41.7	3.90
5	47.1	4.38

Solution:

A. Enter Data

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, STAT, OK (to select "Single var..."), NXT, RESET, DC (to select "Reset all"), OK, NXT, RS, MATRIX |
| 2. | 25.4, ENTER, 1.64, ENTER, DC, 29.3, ENTER, 2.28, ENTER, 32.6, ENTER, 2.75, ENTER, 41.7, ENTER, 3.9, ENTER, 47.1, ENTER, 4.38, ENTER, ENTER |
| 3. | DC, DC, LC, ✓CHK, RC, ✓CHK, OK (read: "Mean:35.22, Std.Dev:8.96"), LS, CLEAR |
| 4. | RS, STAT, OK (to select "Single var..."), RC (to select column number "COL:"), 2, ENTER (to designate data in column 2), DC, LC, ✓CHK, RC, ✓CHK, OK (read: "Mean:2.99, Std. Dev:1.13"), LS, CLEAR |

(Hint: Calculations are for data in column 2, the "set time".)

Exercise 4.4. shows how to use LS STAT to make similar calculations. For both, data will be entered either manually or automatically in the form "[N1 N2]", with the pair of numbers enclosed in square brackets and a space separating them into adjacent columns.

Calculations in these exercises are purposely made to be simple. This allows seeing

more clearly just the process by which data are entered and calculations are made. These skills will be put to more challenging use in the next section.

Exercise 4.4. Array functions with LS STAT

Problem Statement: A chemist found that a newly developed compound, CHEMX, increased the hardness of a kind of paint. Experimental data are summarized in Table 4.4.

Table 4.4. Effects of CHEMX		
Measure	Conc. (mg/dl)	Hardness Index
1	0.13	6.98×10^{-4}
2	0.98	9.28
3	0.51	0.43
4	0.73	2.32
5	0.20	0.01
6	0.84	4.49
7	0.68	1.66

Solution:

A. Enter the Data

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, STAT, DATA, CLΣ, LS, [], .13, SPC, 6.98, EEX, +/-, 4, Σ+, .98, SPC, 9.28, Σ+, .51, SPC, .43, Σ+, .73, SPC, 2.32, Σ+, .2, SPC, .01, Σ+, .84, SPC, 4.49, Σ+, .68, SPC, 1.66, Σ+, STAT |

B. Make the Calculations

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | 1VAR, TOT (read: "[4.07 18.19]"), MEAN (read: "[0.58 2.60]"), SDEV (read: "[0.32 3.35]"), MAXΣ (read: "[0.98 9.28]"), MINΣ (read: "[0.13 6.98E-4]"), NXT, VAR (in view window, position "A"; read: "[0.10 11.22]"), LS, CLEAR, CANCEL, STAT |
| 2. | DATA, ΣDAT, DC (to see data in statistical register) |

At the end of Exercise 4.4, data are displayed in the statistical register with all digits shown, except for the first number in the second column. If you wanted to show more numbers in the view window, press WID➔. The symbol "..." (called an ellipsis) indicates there are

digits in the stored data which are not displayed in the table. Any single data entry can be viewed with all its digits, though, by positioning the highlighted cursor over it and reading the complete number at the bottom left of the view window. The hyphenated numbers in front of the displayed data entry indicate the row and column for it in the STAT table.

If you need to see fewer digits displayed in the table, press ←WID to decrease column width. Using any of the 4 cursor controls (LC, RC, UC, DC) allows you to view, or edit if you need to, any number in the table, even though any one of them might be at first off-screen to either side or to the top or bottom of the view window.

You can modify the data display in the table in other ways too. For example, press NXT to display the set of menus on the next "page". These allow you to delete a row of data targeted by the cursor (with -ROW), add a row (with +ROW) or perform analogous operations on data columns with the menu choices -COL and +COL. Also, you can copy a highlighted number to line 1 of the STACK (with →STK) or gain access to STACK control operations (with ↑STK). Saving an edited table requires simply pressing ENTER. If you wish to abandon the data display and not save the changes you've made, press CANCEL.

The techniques you've seen so far in this chapter will more quickly become second-nature if you design exercises of your own. An important feature of the HP48G is its ability to make calculations on data organized in an array.

Section 4.3.2. Curve-Fitting

Exercise 4.5. (a couple of pages later) shows how to model data you entered for Exercise 4.4. with different types of equations. This is a powerful analytic procedure that yields valuable information about relationships between the dependent and independent variables in an equation. The calculations you are about to review would not at all be easy without the power of the HP48G. A brief introduction about modeling data will most likely be helpful.

Many different kinds of equations can be fitted to a set of data. No matter how complicated the equation is, there is nothing particularly special or useful in just deriving it. Whatever value the equation has lies in its being a tool for accurately describing and predicting other data and for serving in other analyses. Getting the equation itself may be an interesting mathematical exercise, but its power is in knowing how to use it. This section shows you how to do both.

Programs in the HP48G allow you to fit 4 of the more frequently used types of equations. It lets you determine if your data are best predicted by a linear, a logarithmic, an exponential equation or by one which describes a power function. As a shortcut, it will find the best-fitting equation without your having to choose the type. The general form and equation for each of these kinds of relationships is shown in Figures 4.4. to 4.7.

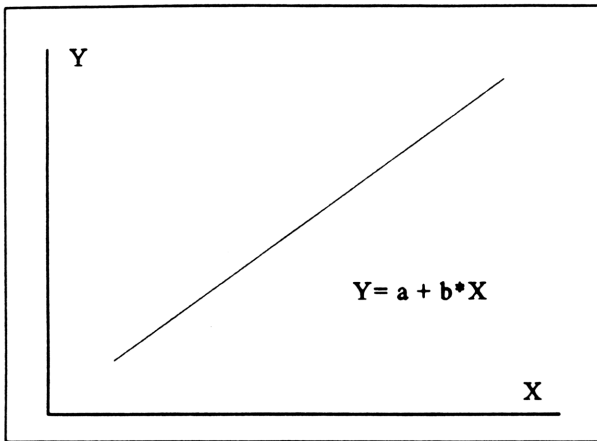


Figure 4.4. Linear Function

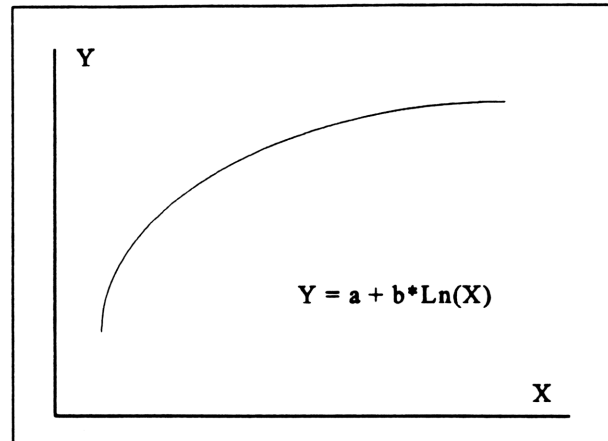


Figure 4.5. Logarithmic Function

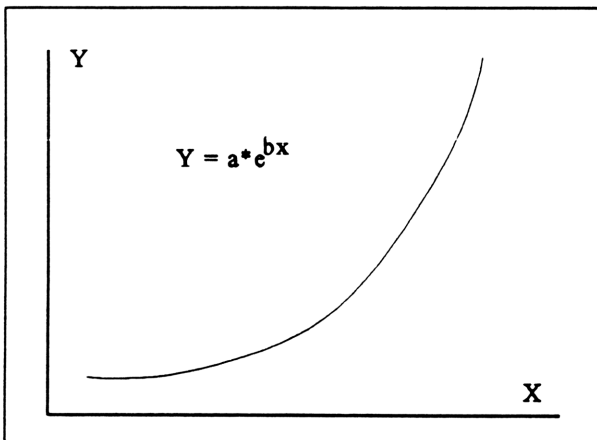


Figure 4.6. Exponential Function

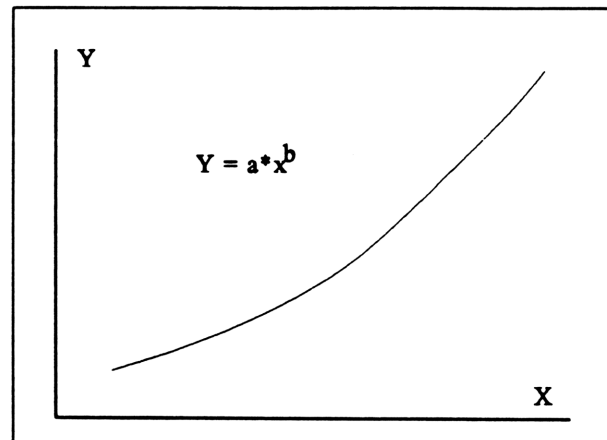


Figure 4.7. Power Function

The general forms for each of the types of equations shown in Figures 4.4. to 4.7. have analogous elements. For each, "a" defines a Y intercept value (the value of Y when X=0) and "b" defines a slope for each line (how much the Y value changes for a change in the X value). Defining the equation that most accurately represents such relationships is valuable for calculating values that cannot be measured directly and for other kinds of analyses.

The practical value for curve-fitting a set of data may not be apparent to everyone at first. The basic idea is to determine the formal mathematical relationship (in the form of an equation) between some event and some other event you think might be related to it. For the data in Exercise 4.4., for example, it would be valuable to have more detailed information about

the relationship between CHEMX and its effect on the physical properties of the paint under development.

There are many ways in which this kind of information is useful. If you were in business, for example, and were trying to determine the best way to present your products to the public, you might ask, "*How are my sales affected by the amount of money I put into different kinds of advertising?*". If you were in agriculture, you might ask of your data, "*How do the number of weeds I get in my crops decrease when I use different concentrations of a herbicide?*" If you were a medical researcher, you might ask, "*What is the effectiveness of this new anti-cancer drug in reducing tumor size?*" The basic question is: "*What is the best equation which describes how my dependent variable (that is, the event that has occurred) varies as a function of my independent variable (that is, the event I think it is related to)?*"

It is conventional to identify the independent variable in a test with the symbol "X" and to label the dependent variable "Y". This is more important than just following tradition. You will need to employ this style in identifying each column of data you test in this way using the HP48G. It is also conventional when drawing a graph to show relationships between data to plot values for the dependent variable (Y) along the vertical axis and plot values for the independent variable along the horizontal axis (X). This is how graphs are defined in Figures 4.4. to 4.7., and it's how you'll see data presented throughout this book.

Exercises 4.5. and 4.6. introduce basic curve fitting procedures. Once data have been entered into the statistical register. (Exercises 4.3. and 4.4.), they are available for a number of calculations including those which derive equations to predict their interrelationships. Curve fitting can be accomplished using the RS STAT functions (Figure 4.1.; Exercise 4.5.) or those available by LS STAT (Figure 4.2.; Exercise 4.6.). The calculations will be the same, of course, it's just that the mechanisms for getting them are different. Try each to see what's best for you.

Exercise 4.5. Curve-fitting with RS STAT

Problem Statement:

Using data entered in Exercise 4.4., calculate best-fit linear, logarithmic, exponential and power function equations. Determine which equation form is most accurate for these data by calculating a correlation coefficient for each.

Solution:

- (**Hint:** 1. If these data have been cleared, repeat Step 1 in Exercise 4.4.
2. Calculations show 3 digits to right of decimal set by: LS, MODES, FMT, 3, FIX)

Make the Calculations

A. For Linear Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, STAT, DC, DC (to highlight "Fit data..."), OK, CHOOS (highlight " Σ DATA: [[.13..."), OK, DC |

(Hint: If "X-COL:1 Y-COL:2", go to Step 3, otherwise complete Step 2)

- | | |
|----|--|
| 2. | Highlight number for X-Col.; key "1", then ENTER, 2, ENTER |
| 3. | CHOOS (highlight "Linear Fit"), OK, OK (read: " $-2.469+8.716*X$ "; Correlation: 0.831; Covariance" 0.889"), LS, CLEAR |

B. For Logarithmic Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | (Same as for A.1.) |
| 2. | (Same as for A.2.) |
| 3. | CHOOS, DC (to highlight "Logarithmic Fit"), OK, OK (read: " $4.861+3.035*LN(X)$ " Correlation: 0.701; Covariance"1.819 "), LS, CLEAR |

C. For Exponential Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | (Same as for A.1.) |
| 2. | (Same as for A.2.) |
| 3. | CHOOS, DC (to highlight "Exponential Fit"), OK, OK (read: " $0.001*EXP(10.703*...;$ Correlation: 0.969; Covariance"1.091 "), LS, CLEAR |

D. For Power Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | (Same as for A.1.) |
| 2. | (Same as for A.2.) |
| 3. | CHOOS, DC (to highlight "Power Fit"), OK, OK (read: " $9.953*X^{4.547}$ "; Correlation: 0.998; Covariance"2.724"), LS, CLEAR |

(Hint: For automatic calculation of the "best-fit" equation based on its correlation coefficient, enter "Best-Fit" at Step 2. Read data for Power Fit equation)

Exercise 4.6. Curve-fitting with LS STAT

Problem Statement: (same as for Exercise 4.5.)

Solution:

Setup (Hint: If data for Exercise 4.4 have been erased, repeat its Step 1)

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | RS, STAT, DC, DC (to highlight "Fit data..."), OK, CHOOS (highlight "ΣDAT: [.13..."), OK, CANCEL |

A. For Linear Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, STAT, ΣPAR, MODL, LINFI, NXT, STAT, FIT, ΣLINE
(read: "'-2.469+8.716*X' "), CORR (read: "0.831), NXT, STAT |

B. For Logarithmic Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, STAT, ΣPAR, MODL, LOGFI, NXT, STAT, FIT, ΣLINE
(read: "' 4.861+3.035*LN(X)'"), CORR (read: "0.701), NXT, STAT |

C. For Exponential Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, STAT, ΣPAR, MODL, EXPFI, NXT, STAT, FIT, ΣLINE
(read: "'0.001*EXP(10.703*X)'"), CORR (read: "0.969), NXT, STAT |

D. For Power Fit

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, STAT, ΣPAR, MODL, PWRFI, NXT, STAT, FIT, ΣLINE
(read: "'9.953*X^4.547' "), CORR (read: "0.998), NXT, STAT |

(Hint: For automatic calculation of the "best-fit" equation based on its correlation coefficient: LS, STAT, ΣPAR, MODL, BESTF (read: (data for Power Fit)), NXT, STAT, FIT, CORR (read: "0.998"))

Which of these four equations calculated in Exercise 4.4. best predicts relationships between the dependent and independent variables for the data you have entered is indicated, of course, by which model has the correlation coefficient closest to 1.0. For these data, the model using a power function equation yields the best-fit.

The calculation of a correlation coefficient (commonly symbolized by "r") for a set of data provides a useful statistic. Like an equation itself, though, it has no intrinsic value, it has to be accurately and appropriately interpreted. When it receives such an analysis, though, it reveals how well one variable relates to another.

Were a correlation coefficient calculated to be zero for a set of data, it indicates there is no predictive relationship between the data. The dependent variable (Y) does not vary as a function of the independent variable (X). Such a relationship is likely to be deduced were you to compare, for example, how many birds come each hour to your feeder in the backyard over the period of a week, compared to the stock market price index for the same week.

Were a correlation coefficient calculated to be 1.00 (the highest possible value) for a set of data, it indicates that there is a highly predictive relationship between the data you have compared. The dependent variable (Y) would change characteristically in a very predictable fashion for each change in the independent variable (X). A relationship similar to this is revealed for how the paint hardness index varies as a function of the concentration of CHEMX in Exercise 4.4. when a power function equation is used to describe the relationship.

A correlation coefficient calculated to be -1.00 would define a similarly excellent relationship between the events being compared. It would show, though, that as the independent variable (X) increased, the dependent variable (Y) decreased, and *vice versa*. Had the chemist in Exercise 4.2., for example, compared the effects of concentrations of CHEMX to a "softness index" for his paint, you'd reasonably expect that this index would decrease in a very predictable way as the concentration of CHEMX increased.

The correlation coefficient for a set of data yields yet additional information. As a rule-of-thumb, subtracting the calculated correlation coefficient from 1.00 indicates a good first-order estimate of how much of the relationship between the variables X and Y is unaccounted for by the best-fit equation computed for them. For example, a correlation coefficient of 1.00 indicates the most accurate evaluation with zero percent of the data effects left to be explained (since $1.00 - 1.00 = 0.00$). A correlation coefficient of 0.93, for example, shows a generally good equation-fit for the data set, but not a perfect one. About 7% of the observed values for the dependent variable are not related to the independent variable for whatever equation was used as a test model (since $1.00 - 0.93 = 0.07$). A correlation coefficient of 0.50 indicates that only about 50% of the relationship between two events can be predicted accurately by the equation tested, not a good predictor at all.

Just because there is a high correlation coefficient calculated for the relationship between two variables is not good grounds to expect that one causes the other. There might be, for example, only by happenstance a good correlation between the changes in gasoline consumption in a small midwestern town in the United States and variations in the price of eggs in a village in southeast Asia. The logic is flimsy, though, if one then expects that not driving

one's car so much in the midwestern town is going to affect overseas agricultural values.

On the other hand, the implication is strong from data in Exercise 4.2. that adding increased concentrations of CHEMX to a batch of paint increases its hardness. That's a good working hypothesis for the test. Inferences drawn from the statistical tests provide valuable guideposts for the chemist's research team as they start to track down the suspected and expected causal relationship. It might be, of course, just coincidence, but more lab tests will soon provide the answer. The statistical test, though, just indicates the phenomena are related, nothing else.

Section 4.3.3. Making Predictions

A major value in determining a "best-fit" equation for a set of data, as described in Exercises 4.5. and 4.6., is that predictions can be made from it. Any value for the dependent variable (Y) can be predicted from a corresponding value for the independent variable (X), and *vice versa*. How accurate these predictions are depends on how reliable the equation represents the data set (indicated by the correlation coefficient), and that predictions are made generally within the range of the data set from which the equation was derived.

This means that for the data presented in Exercise 4.4., the "hardness index" for a batch of paint can be predicted for any concentration of CHEMX now that a "best-fit" equation has been determined for the experimental data (Exercises 4.5. and 4.6.). Also, the concentration of CHEMX required to produce any desired level of hardness can also be determined. Once a "best-fit" equation has been deduced, these predictions can be made graphically or equationally (Exercise 4.7.)

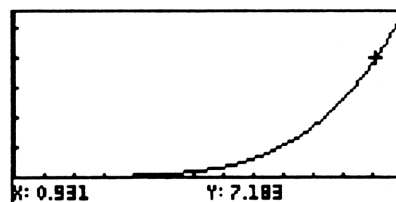
Exercise 4.7. Predictions

Problem Statement: Based on data in Exercise 4.4. and calculations completed in Exercises 4.5. and 4.6., determine the concentration of CHEMX predicted to produce a "hardness index" of about 7.2 for a paint mixture.

(**Reminder:** The "best-fit" equation for the data in Exercise 4.4. was: $Y=9.953 \cdot X^{4.547}$.)

Graphic Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | ', 9.953, *, α , X, y^x , 4.547, ENTER |
| 2. | ', α , α , C, H, E, M, α , STO |
| 3. | RS, PLOT, CHOOS (highlight "CHEM: '9.953..."), OK, DC, RC, 0 (zero), ENTER, 1, ENTER, \checkmark CHK, ERASE, DRAW (see function in: "Display 1") |
| 4. | TRACE, (X,Y), (hold down RC until "Y:7.183"; read: "X:0.931), CANCEL, CANCEL |



Equation Solution:

(Hint: Assume HOME directory is empty)

Construct the Equation

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | , α , Y, LS, =, 9.953, *, α , X, y^x , 4.547, ENTER |
| 2. | , α , α , H, A, R, D, α , STO |

A. Using RS SOLVE

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | RS, SOLVE, OK (to select "Solve equation..."), CHOOS, OK (to select "HARD: Y=9.953..."), 7.2, ENTER, SOLVE (read: ".93126+"), CANCEL |

B. Using LS SOLVE

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, SOLVE, ROOT, SOLVR, 7.2, Y (view window), LS, X (view window; read: "X:0.931"), CANCEL, LS, CLEAR, VAR |

Section 4.4. A Mystery Solved with the HP48G

For many real life situations, obtaining statistical data for sets of numbers is only the beginning of the analysis. Despite the beliefs of many, the numbers do not speak for themselves, they have to be interpreted. Statistical profiles of data and equations that define their relationships give only the first clue for seeing what they mean in the context of a specific problem. Without them, though, analysis would be either more difficult or impossible. Exercise 4.8. gives an example of how statistical data are combined with equation solutions to yield valuable insight into a problem.

Exercise 4.8. Curve-fitting With Equation Solutions

Problem Statement: A partially clothed body was discovered in a wooded area about 5:00 PM on a cloudy afternoon in early fall. The coroner arrived at the scene at 5:30PM and recorded the deceased's body temperature at 30 minute intervals over the next several hours (Table 4.5). A wind was blowing during this time, but air temperature remained about 70°F. What time did the person die? (Hint: Objects cool exponentially under constant environmental conditions. Normal body temperature is assumed to be 98.6°F.)

Table 4.5. Body Temperatures (T_b)		
Observation	Time (clock hours)	Body Temp. ($^{\circ}\text{C}$)
1	5:30	83.5
2	6:00	80.5
3	6:30	78.2
4	7:00	76.4
5	7:30	75.0
6	8:00	73.9
7	8:30	73.0
8	9:00	72.3

Solution:

A. Define the differences between body and air temperatures as a function of measurement time intervals.

Table 4.6. Body (T_b) and Air (T_a) Temperature Differences			
Observation	Time (min.)	Interval (hrs.)	$T_b - T_a$ ($^{\circ}\text{F}$)
1	0	0.0	13.5
2	30	0.5	10.5
3	60	1.0	8.2
4	90	1.5	6.4
5	120	2.0	5.0
6	150	2.5	3.9
7	180	3.0	3.0
8	210	3.5	2.3

(**Hint:** The first temperature measurement was at 5:30 PM, then measured at 30 minute intervals)

B. Determine numerical values for the exponential equation using data in Table 4.7.

(Hint: The first data entry for "Time" in Table 4.6. is chosen to be "0.0001" because a zero cannot be used in this calculation.)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, MATRIX, 0.0001, ENTER, 13.5, ENTER, DC, .5, ENTER, 10.5, ENTER, 1, ENTER, 8.2, ENTER, 1.5, ENTER, 6.4, ENTER, 2, ENTER, 5, ENTER, 2.5, ENTER, 3.9, ENTER, 3, ENTER, 3, ENTER, 3.5, ENTER, 2.3, ENTER, ENTER |
| 2. | ', α , α , T, E, M, P, α , STO |
| 3. | RS, STAT, DC, DC (to select :Fit data..."), OK, CHOOS (move cursor to "TEMP:[[.0001. ..."), OK, DC, DC, CHOOS, (highlight: "Exponential Fit"), OK, OK (read: "13.56*EXP(-.50*X...) |

(Hint: The "best-fit" equation is: $(T_b - T_a) = 13.56 e^{-0.5(\text{time})}$; $(T_b - T_a)$ has units of degrees Fahrenheit and (time) has units of hours.)

C. Determine time of death

(Hint: Solve for X to indicate time in hours before the first temperature measurement at 5:30PM when body temperature was 28.6°F above air temperature (98.6 - 70 = 28.6).)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, EQUATION, 28.6, LS, =, 13.56, *, LS, e^x , -, .5, *, α , T, RC, ENTER, ', α , α , D, E, A, D, α , STO |
| 2. | RS, SOLVE, OK (to select "Solve Equation..."), CHOOS, (highlight DEAD: '28.6=13...'), OK, SOLVE (read: "-1.49+"), CANCEL, LS, CLEAR |

(Hint: It's a good guess that the person was alive with a normal body temperature of 98.6°F about 1.5 hours before the coroner made the first measurement at 5:30PM. This means that death occurred about 4:00PM when the process of body cooling began.)

D. A Graphical Solution

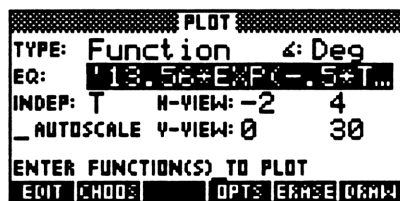
A. Write the equation

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, 13.56, *, LS, e^x , -, .5, *, α , T, RC, ENTER (read: "' 13.56*EXP(-0.50*T) '") |
| 2. | ', α , α , T, M, P, α , STO |

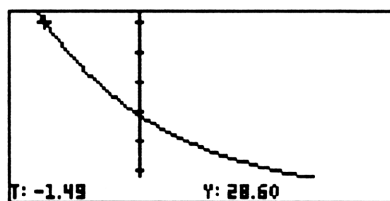
B. Construct the graphic

(**Reminder:** The function "CANCL" is activated by key "F" under the view window. The function "CANCEL" is obtained by pressing the "ON" key.)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, PLOT ("EQ." is highlighted), CHOOS, (highlight "TMP: '13.56..."), OK, DC, α , T, ENTER, 2, +/-, ENTER, 4, ENTER, RC, 0 (zero), ENTER, 30, ENTER (see "Display 1"), ERASE, DRAW |
| 2. | TRACE, (X,Y), LC (until "Y:28.6"; read: "T:-1.49"; see "Display 2"), NXT, CANCL, CANCEL |



Display 1



Display 2

(**Hint:** the graphical solution confirms that body temperature was about 98.6°F (that is, 28.6°F above ambient temperature (70°F)) about 1.5 hours (4:00PM) before the first body temperature was measured at 5:30PM.)

Figure 4.8. is a graph for the solution to the mystery described in Exercise 4.8. For other situations, of course, some features of this problem will remain the same and others will change depending on different personal and environmental conditions. For example, for a normal person, initial body temperature (about 98.6°F) will always be the same independently of sex, age, body size, etc. The rate at which body temperature falls, however, depends on the person's size and surface area, how the body is exposed to the environment, whether the clothing or skin surface is wet or immersed in water, the wind speed, air temperature, the warming effects of direct sunlight and many other factors. If any set of these conditions is constant, though, an exponential function will describe best how temperature decreases with time, as it does in Exercise 4.8. The rate of cooling will change, but the initial temperature value and the form of the best-fit equation will remain the same.

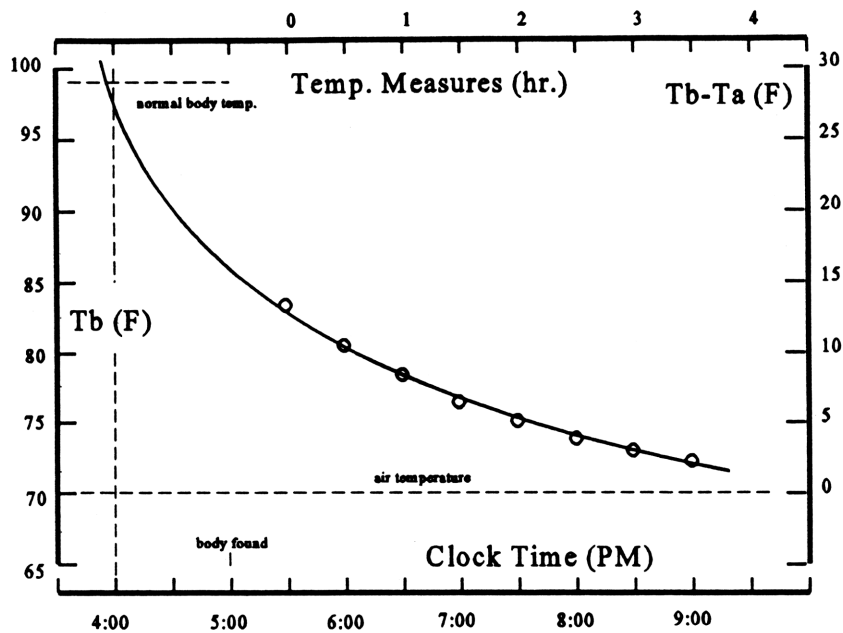


Figure 4.8. Body Temperature Changes

Section 4.5. Directories: Basic Concepts

Section 3.3. introduced an important basic idea that numbers, numbers with units and even equations are stored as menu options which are then brought to the view window by pressing VAR. So far, all data and equations have been stored in this way under the HOME directory. A reminder that you were using this directory has been shown in the upper left of the view window as you worked through all of the exercises and examples.

There is no reason not to continue to use the HOME directory. It's worked well for you up to now and will continue to do its job. But the more VAR menus you construct, the more cluttered the HOME directory menu display becomes and the more "pages" you have to search with NXT to find the menu option you need. Also, you become progressively more burdened by not being able to use the same menu designation for a variable. For example, if you used the symbol "N" as a variable name (as you most likely did for Exercise 2.4.), you cannot use "N" again in the HOME directory without running the risk of making an incorrect calculation. There is an easy solution. Construct your own set of subdirectories, name them what you want and store in each of them whatever you want.

Some people find the concepts of directory construction a little confusing at first. Organizing, constructing and using them is no more difficult, though, than the process you'd

use for any other kind of filing system. An example gives the basic idea.

Imagine you are just starting a small business, but have already a number of steady customers. You will need, of course, to maintain records for these people about the orders they've placed, the deliveries you've made, the bills you've submitted to them, the payments you've received, correspondence you've exchanged with them, advertising you've generated and keep files of other business

transactions too. It's unlikely you'd keep all of these important papers mixed up together in an old shoe box under the sink. More than likely, you'd invest in a file cabinet and organize it with a system similar to that shown in Figure 4.9.

If you used the system shown in Figure 4.9., one file drawer (one directory) would be titled "BILLS" and other drawers (directories) might have "CORRESPONDENCE", "INVENTORY" or similar headings. In the directory "BILLS" would be most likely a series of file folders (subdirectories) indicating different types of bills. Some might be labelled "PAID", "OUTSTANDING" or "TO BE SENT". Each of these folders might contain additional subdirectories, like "OVERDUE 3 MOS.", "OVERDUE 6 MOS.", and the like. Other file draws would be subdivided and appropriately titled in a similar way. The purpose of this kind of organization is, of course, to make it easier to find and use items when you need them..

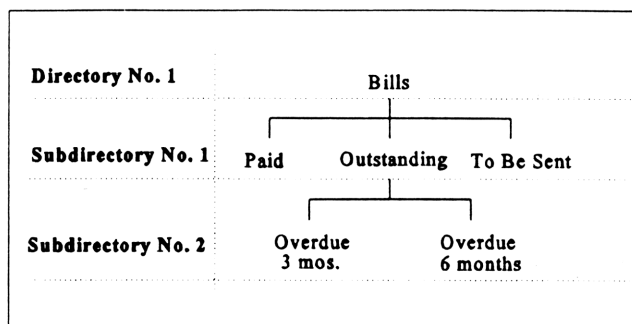


Figure 4.9. A Basic Filing System

Section 4.6. Directory Structure and Construction

Figure 4.10. lists the commands presented by RS MEMORY. They are used for constructing, editing and otherwise manipulating the contents of subdirectories. Figure 4.11. shows similar controls using LS MEMORY. Exercise 4.9. gives simple examples for creating subdirectories, moving files among them, changing their contents and erasing them when they are no longer needed. Once these skills are acquired, it's a good idea to use them so that each new problem, calculation or program is listed in its own subdirectory. The HOME directory will then list and provide access to these different subdirectories, but will not contain any programs of its own. Besides the generally good organization this provides, there are hidden benefits in terms of controlling the use of local variables. All this leads to efficient use of the HP48G, less confusion among programs and a reduced chance for errors when local labels are used in programs.

Figure 4.10. RS MEMORY

		Menu Keys					
page	A	B	C	D	E	F	
1	EDIT*	CHOOS	✓CHK	NEW*	COPY*	MOVE*	
2	RCL	PURG	SIZE		CANCL	OK	

The "*" indicates a directory whose menus are:

*EDIT (same for *NEW, *COPY and *MOVE)

1	EDIT	CHOOS			CANCL	OK
2	RESET	CALC	TYPES		CANCL	OK

Figure 4.11. LS MEMORY

		Menu Keys					
page	A	B	C	D	E	F	
1	MEM	BYTES	NEWO		DIR*	ARITH*	
2	ARCHI	RESTO					

The "*" indicates a directory whose menus are:

*DIR

1	PATH	CRDIR	PGDIR	VARS	TVARS	ORDER
2						MEM*

*ARITH

1	STO+	STO-	STO*	STO/	INCR	DECR
2	SINV	SNEG	SCON			MEM*

Exercise 4.9. Subdirectory Functions**A. Creating a Subdirectory**

Problem Statement: Construct a subdirectory called "TEST" in the HOME directory.

Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, MEMORY, NEW, DC, α , α , T, E, S, T, α , ENTER, \checkmark CHK, OK, CANCEL |
- (Alternative: ', α , α , T, E, S, T, α , ENTER, LS, MEMORY, DIR, CRDIR)

B. Moving a File to a Subdirectory

Problem Statement: Create a file in the HOME directory called "ABC" that contains the number "123", then move it to the TEST subdirectory.

Solution: Beginning in the HOME directory:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | 123, ENTER, ', α , α , A, B, C, α , STO |
| 2. | RS, MEMORY (highlight "ABC:123"), \checkmark CHK, MOVE, CHOOS, (highlight "TEST"), OK, OK, CANCEL |

(Hint: To copy a file, use "COPY" instead of "MOVE" at Step B.2.)

C. To Change the Contents of a File

Problem Statement: Change the contents of "ABC" in subdirectory TEST from "123" to "789".

Solution: Beginning in the HOME directory:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | TEST, RS, MEMORY, (highlight "ABC 123"), EDIT, EDIT, (move cursor to end of "123" and erase it), 7, 8, 9, ENTER, OK, NXT, CANCL, CANCEL |

D. To Erase a Subdirectory File

Problem Statement: Erase file "ABC" in the TEST subdirectory.

Solution: Beginning in the HOME directory:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | TEST, RS, MEMORY, (highlight "ABC"), \checkmark CHK, NXT, PURG |

Section 4.7. More Progress

Working directly from the keyboard with the functions in the LS STAT and RS STAT menus gives powerful advantages in calculating basic statistics and for fitting curves to data. This chapter gave several examples. There is even more power and ease of use when the STAT functions are addressed by steps in a program. See the programs "Useful Statistics" and "Automatic Curve Fitting" in Chapter 8.

All the skills you have acquired by now can be applied with benefit as you continue to explore the full potential of the HP48G. One of its special features is how it stores and executes programs. It is a major bonus in using this machine to be able to write customized programs for your own needs and applications. The next chapter gets you started.

Chapter 5

Basic Programming

There are several ways to use the HP48G's programming capabilities to solve equations and to perform other logic tasks. This chapter describes a few simple programs to get you started with the basic techniques. In no case does it show the only way to write a program and maybe not even the best way to do it. Programs here are, after all, designed to promote step-by-step learning for the beginner. Which kind of program structure is best for any specific application depends on how you want to enter data, what you want the program to do with them and how you want to see solutions displayed or manipulated for other calculations. A practical standard is that any program is a good one if it runs accurately and quickly and takes no more programming steps or memory than absolutely necessary.

Section 5.1. Getting Started

Just getting started writing programs is often hard for those who have little or no experience with it. So, the easier the first programs are, the better. From a simple beginning, you'll see how more and more features are added to programs in other exercises as the chapter continues and as you build your skill level. It won't be long before you'll set off on your own to construct programs as involved and as complicated as you need. Breaking the ice is the hardest part.

Section 5.1.1. A Program Map

It doesn't take more than a couple of hours for the beginning driver to learn how to start and stop a car, how to make it turn at a corner, how to steer a desired path and how to perform other basic maneuvers. Even with these skills, though, taking a trip from Chicago to Los Angeles, for example, would be chancy without a well-thought out plan. The familiar road map is, of course, the model most of us use for such a plan. It defines where we start, what route we take, and what our destination is for a particular trip. In a similar way, it doesn't take more than a couple of hours for the beginner to learn how to turn a computer on and off, how to perform its basic mathematical and logic operations and how to activate appropriate keystrokes. You've already done this for the HP48G by progressing this far in the book. Writing a computer program using these skills will be easier and less time consuming if you also learn how to use a map (of sorts) to document where you want to start in the program, what set of sequential steps and endpoint you want.

Most people who've taken car trips know it doesn't matter too much whose map you use, as long as it's accurate and easy to read. Ones provided by gas stations are just about as good for the purpose of the trip as are those bought in supermarkets or in bookstores. Similarly, flow diagrams for a computer program work well even though one may be organized slightly differently or uses different symbols than another. It's important the same symbols mean the same thing from place to place in the same program, but one style works about as well as another. Figure 5.1. defines the symbols used for flow diagrams in this book.


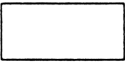
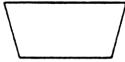
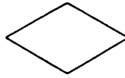


<i>Symbol</i>	<i>Operation</i>	<i>Note</i>
	<i>Start or Stop</i>	<i>1 input or 1 output</i>
	<i>Procedure</i>	<i>1 input and 1 output</i>
	<i>Data Entry</i>	<i>1 input and 1 output</i>
	<i>Decision</i>	<i>1 input; 1 of 2 outputs</i>
	<i>Display</i>	<i>1 input and 1 output</i>
	<i>Off-page Connector</i>	<i>1 input or 1 output</i>

Figure 5.1. Flow Diagram Symbols

Many of the first programs described in this chapter could be written without any flow diagram at all. The logic for them is simplicity in itself and the math operations are trivial. But as programs become more complicated, the flow diagrams become more important. Getting used to using them in the beginning is a good habit and saves time in the long run. Of the many hours invariably spent in de-bugging a complicated program, few are used to track down mathematical or computational errors. Most are spent trying to get the logic of the program to operate in the way you want. Simple programs will be difficult to write accurately the first time without a flow diagram and complicated ones will be easier to write accurately with a flow diagram, even for the expert.

Figure 5.2. The PRG Menu

		Menu Keys					
page	A	B	C	D	E	F	
1	BRCH*	TEST*	TYPE*	LIST*	GROB*	PICT*	
The "*" indicates a directory whose menus are:							
*BRCH							
1	IF*	CASE*	START*	FOR*	DO*	WHILE*	
2	IFT	IFTE					
*TEST							
1	==	≠	<	>	≤	≥	
2	AND	OR	XOR	NOT	SAME	TYPE	
3	SF	CF	FS?	FC?	FS?C	FC?C	
4	LININ						
*TYPE							
1	OBJ→	→ARR	→LIST	→STR	→TAG	→UNIT	
2	C→R	R→C	NUM	CHR	DTAG	EQ→	
3	TYPE	YTYPE					
*LIST							
1	ELEM*	PROC*	OBJ→	→LIST	SUB	REPL	
*GROB							
1	→GRO	BLAN	GOR	GXOR	SUB	REPL	
2	→LCD	LCD→	SIZE	ANIM			
*PICT							
1	PICT	PDIM	LINE	TLINE	BOX	ARC	
2	PIXON	PIXOF	PIX?	PVIEW	PX→C	C→PIX	
For BRCH							
*IF							
1	IF	THEN	ELSE	END		BRCH*	
*CASE							
1	CASE	THEN	END			BRCH*	
*START							
1	START	NEXT	STEP			BRCH*	
*FOR							
1	FOR	NEXT	STEP			BRCH*	
*DO							
1	DO	UNTIL	END			BRCH*	
*WHILE							
1	WHILE	REPEA	END			BRCH*	
For LIST							
*ELEM							
1	GET	GETI	PUT	PUTI	SIZE	POS	
2	HEAD	TAIL					
*PROC							
1	DOLIS	DOSUB	NSUB	ENDS	STREA	REVL	
2	SORT	SEQ					

Section 5.1.2. Menus for Program Writing

Like so many other of the HP48G's features, writing programs depends on using the machine's built-in menus. Their organization is somewhat different from other menus, as shown in Figure 5.2.

For the most part, the subdirectories and menus shown in Figure 5.2. are used the same way as others you've seen in earlier chapters. Just pressing PRG, for example, shows a list of directories, each of which has its own set of menus. A difference, though, is that menus within BRCH have several functions. They are shown at the bottom of the figure.

Section 5.2. Storing Equations and Data as Variables.

An equation and a computer program share important characteristics. Each is a statement of relationships which if evaluated correctly lead to an accurate conclusion. For an equation, this conclusion is often the calculation of a number, or if it is being solved symbolically, it is a new statement of mathematical relationships among its variables. For a computer program, the conclusion may be expressed as a numerical value, or it may be a non-numerical statement about some decision the computer has been programmed to reach. Just as in solving an equation, though, steps in using a computer program must be taken one at a time following specific rules. Also like mathematical rules, those for program writing are simple for the most part, but they cannot in any way be ignored or misused.

The simplest way to construct a set of usable steps to solve an equation using the HP48G is to store the equation itself as a VAR menu. In a sense, the equation becomes its own computer program. The simplicity and utility of this strategy is that the user has no decisions to make except to identify numerical values for the equation to use and then just has to press a button to start the solution process. You were introduced to this technique in Exercise 3.1.

Expanding on such a simple plan is a good place to start becoming familiar with the processes of program writing. A simple example shows this procedure is much easier to learn and use than it might sound. Exercise 5.1. describes how the HP48G VAR menus are used to solve a single equation which determines a student's examination grade. Several other programs will be written in this chapter. Take advantage of the directory system you constructed for your computer in Exercise 4.9. for constructing them. Make your own choice about where you want to keep these programs, but a good suggestion is to store them under each program's name within the subdirectory.

The first exercise shows how to solve an equation by storing it in a VAR menu. It presents the simplest style for program construction. Figure 5.3. shows its flow diagram.

Exercise 5.1. Examination Score

Problem Statement:

Sam Tutor teaches a large class and decides that a computer program would help him score examinations. Each exam in the course has a different number of questions and bonus points. Write a program which will calculate each student's percent score based on the number of earned points (the "raw score"), bonus points and questions on each exam.

Solution:

A. Define the Equation.

$$ES = \frac{100 (C+B)}{N}$$

where:

- ES = examination score (percent)
- C = number of correct questions, or "raw score"
- B = examination bonus points (1 point equals 1 question)
- N = number of questions on the exam

B. Write the Program

Hint: The program will contain the equation and will appear as:

```
<<'100*(C+B)/N'→NUM>>
```

Step

Press

1. LS, <<>>, ', 100, *, LS, (, α, C, +, α, B, RC, ÷, α, N, RC, LS, →NUM, ENTER
2. ', α, E, α, S, STO

(**Hint:** The equation is now written in the form of a program and is stored in VAR as "ES". To review the equation, press: ', ES, ENTER, LS, EDIT. Press CANCEL to clear the view window and return to normal operation.)

C. Use the Program.

One of the exams Mr. Tutor gave to his class had 80 questions and 3.5 bonus points. Calculate the percent score for each of the students whose "raw scores" are listed in Table 5.1.

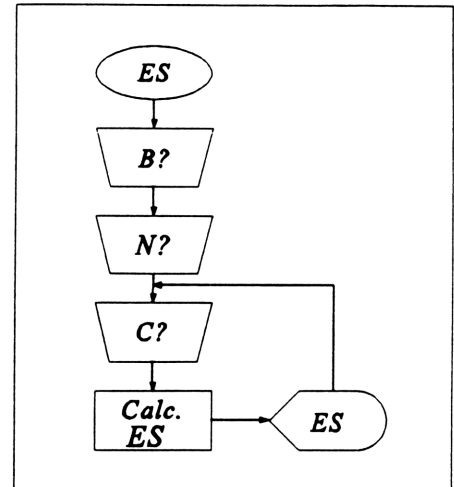


Figure 5.3. Exercises 5.1. and 5.2.

(**Hint:** As you complete the following steps, it's easy to get confused in distinguishing the menu options titled "B" and "C" from the white keys which have at their lower right corner the same letters. It will help to remember that any time you press ALPHA, the next keystroke will generate the letter printed to the right of the key. If you don't press ALPHA first, then the choice "C", for example, can be a menu option selected from the display at the bottom of the view window.)

Table 5.1. Student Raw Scores, Exam 1	
Student	Raw Score
A. Lincoln	71
G. Wash	68
T. Jeff	70
B. Franklin	75
D. Madison	72

To make the calculations:

Step	Press
1.	3.5, ENTER, ', α , B, STO
2.	80, ENTER, ', α , N, STO
3.	71, ENTER, ', α , C, STO, ES (read: "93.13")
4.	68, ENTER, ', α , C, STO, ES (read: "89.38")
5.	70, ENTER, ', α , C, STO, ES (read: "91.88")
6.	75, ENTER, ', α , C, STO, ES (read: "98.13")
7.	72, ENTER, ', α , C, STO, ES (read: "94.38")

(**Hint:** Don't erase this program yet. It will be used in the next exercise.)

As Described in Chapter 3, numerical solutions for an equation stored in a VAR menu require having a number defined for each symbol in the equation except, of course, for the one you're solving for. Using a program to solve an equation requires the same kind of definition. Steps A and B in Exercise 5.1. provide that information for the constants B and N. Once they have been entered, a student's exam score (ES) is calculated by defining C then activating the program ES. Exam scores for other students are calculated simply by redefining C (the "raw score") for each of them. This is performed by Steps 3 to 7 in Stage C. How this operation related to general program flow is shown in the flow diagram (Figure 5.3.).

Mr. Tutor's course, like most others, has several different examinations throughout the term. Also, other instructors use the program he has written. The program is easily modified to meet the specific conditions for these different applications. The number of questions on each exam, for example, is easily changed just by repeating Steps A and B

Exercise 5.1. Both of these pieces of information remain as constants for scoring any single exam and only the student's raw score needs to be changed for each calculation.

This programming design has both advantages and disadvantages. An advantage is that the numerical values for the two constants (B and N) need to be defined in their respective VAR menus only once. The equation listed in the VAR menu "ES" uses these stored data, along with that in VAR menu "C" for each calculation. A disadvantage of the program is that it requires several keystrokes to enter the raw score for each student each time a new calculation is made. The next section and Exercise 5.2. show a more direct and easier way to enter data into a program.

Section 5.3. Using Local Variables.

A more practical program for Mr. Tutor and others to use for calculating examination grades would have the constants for bonus points and the number of exam questions entered only once, but have the program read each student's raw score from a value keyed into line 1 of the STACK. Exercise 5.2. shows how to rewrite the program to include this feature. The flow diagram is the same as that shown in Figure 5.3.

Exercise 5.2. Examination Score - A Better Idea

Problem Statement:

Amend the program in Exercise 5.1. so that each student's "raw score" is keyed into line 1 for calculation, but it doesn't have to be stored in a VAR menu.

Solution:

Hint: The program will appear as:

```
<< → C ' 100*(C+B)/N ' →NUM >>
```

A. Amend the program "ES" which you wrote for Exercise 5.1.

Step

Press

1. ' , ES, ENTER, LS, EDIT, RC, RS, →, α, C, ENTER

(**Reminder:** Pressing ENTER stores the amended program under the same menu title, "ES". The program described in Exercise 5.1. no longer exists. It has been replaced by the amended one.)

B. Using the Program

Use the newly defined program titled ES to compute examination scores for students whose "raw scores" are listed in Table 5.2. This examination had 115 questions and 4.2 bonus points.

To make the calculations:

<u>Step</u>	<u>Press</u>
1.	4.2, ENTER, ', α , B, STO
2.	115, ENTER, ', α , N, STO
3.	92, ES (read: "83.65")
4.	80, ES (read: "73.22")
5.	70, ES (read: "64.52")
6.	110, ES (read: "99.30")
7.	100, ES (read: "90.61")

Student	Raw Score
A. Lincoln	92
G. Wash	80
T. Jeff	70
B. Franklin	110
D. Madison	100

The program described in Exercise 5.2. has many features similar to the one you used in Exercise 5.1. It is different, however, in that data used by the equation stored in a VAR menu are drawn either from the STACK (for a student's raw score (C)) or from VAR menus (for the bonus points (B) and for the number of questions (N)). The program is only a little more complicated than the first one, but it is much easier and faster to run. In this sense, it is a more practical format for Mr. Tutor and his academic associates to use.

The amended instruction (" \rightarrow C") in Exercise 5.2. instructs the computer to take the value for C (the student's "raw score") from level 1 in the view window. This number is not stored anywhere after that, it is used just once in the ES calculation. As shown in Steps 3 to 7 in Exercise 5.2., each new "raw score" for each student is keyed into line 1, then just pressing ES computes the student's exam score. An advantage of this program's format is that fewer keystrokes are necessary than in Exercise 5.1. to calculate each student's grade.

Mr. Tutor has other calculations, however, to make for his course. Not only does he need to determine a percent grade for each exam, he also has to calculate a final course grade for each student at the end of the term. This grade depends, of course, on how well each student has performed on each exam during the term and on how much each exam is worth in determining the final grade. Exercise 5.3. shows how this new calculation is made just about as easily as was the grade for each exam in Exercise 5.2. Figure 5.4. presents the flow diagram.

Exercise 5.3. Course Score I

Problem Statement:

Sam Tutor's course has 3 examinations. Each has a different number of questions, bonus points and weight in determining a student's score for the course. These data are listed in Table 5.3. Write a program that calculates each student's score for the course based on the number of questions he/she got right on each exam. Design the program so that this information is listed in the STACK for the program to use.

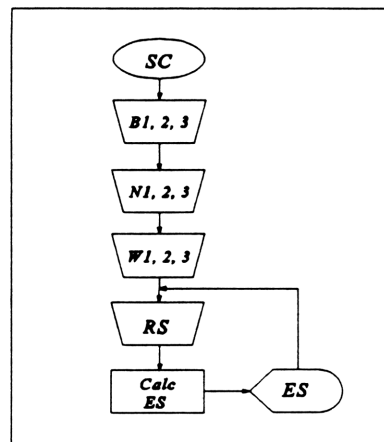


Figure 5.4. Exercises 5.3. and 5.4.

Table 5.3. Examination Data			
Examination	No. of Quest.	Exam Wgt (%)	Bonus Pts.
1	30	25	5
2	40	20	3
3	60	55	1

Solution:

A. Define the Equation.

$$CS = W1(ES1) + W2(ES2) + W3(ES3)$$

where:

CS = score for the course (percent)

W1,W2,W3 = weights of Exams 1,2 and 3 (W1+W2+W3=100)

ES1,ES2,ES3 = percent scores on Exams 1,2 and 3

or,

$$CS = \frac{25(A+5)}{30} + \frac{20(B+3)}{40} + \frac{55(C+1)}{60}$$

(Reminder: 25(A+5)/30 yields the same result as: 100(.25(A+5)/30))

where:

A,B,C = number of correct questions ("raw score") for any student on exams 1,2,3

B. Write the Program.

Hint: the program will appear as:

<<> A B C '25*(A+5)/30+20*(B+3)/40+55*(C+1)/60'>>

Step

Press

1. LS, << >>, RS, →, α, A, SPC, α, B, SPC, α, C, ', 25, *, LS, (), α, A, +, 5, RC, ÷, 30, SPC, +, 20, *, LS, (), α, B, +, 3, RC, ÷, 40, SPC, +, 55, *, LS, (), α, C, +, 1, RC, ÷, 60, ENTER
2. ', α, S, α, C, STO

(Hint: Program is now stored under menu "SC")

Stage 3. Run the Program

Calculate course scores for each of the students whose exam scores are shown in Table 5.4.

Table 5.4. Exam Raw Scores for "Course A"			
Student	Exam 1	Exam 2	Exam 3
A. Lincoln	20	18	40
G. Wash	23	22	50
T. Jeff	18	17	43
B. Franklin	25	27	59
D. Madison	19	25	57

To make the calculations:

Step

Press

1. 20, ENTER, 18, ENTER, 40, SC (read: "68.92")
2. 23, ENTER, 22, ENTER, 50, SC (read: "82.58")
3. 18, ENTER, 17, ENTER, 43, SC (read: "69.50")
4. 25, ENTER, 27, ENTER, 59, SC (read: "95.00")
5. 19, ENTER, 25, ENTER, 57, SC (read: "87.17")

Exercise 5.3. shows how more than one piece of data is drawn from the STACK to be used in an equation. The raw scores for each exam are first listed in the STACK for a particular student, then pressing SC begins the calculation of the course grade. As shown in the equation, how much each exam contributes to the overall grade is determined by multiplying the percent score for each exam (calculated by dividing total exam points (raw score plus bonus points) by the number of exam questions) by its relative percent weight. The sum of these products states the student's final grade.

The instruction " \rightarrow A B C" in the program tells the computer to use the first three numbers in the STACK (lines 1, 2 and 3) and define for each the variables A, B, and C, respectively, for program use. An advantage to the program is that data for each exam are readily entered into the STACK and the calculation is made with the press of a single key (SC). Although this program would work well in many applications, it has at least a couple of disadvantages. One is that Mr. Tutor will have to write a new version of the program, or edit the present one, if the number of questions on an exam, its relative weight or its bonus points change from term-to-term, as they probably would. This is not hard to do, but it's an extra job. Another disadvantage is that whoever uses this program in whatever form it has must remember the specific order for entering raw score grades. Incorrectly ordered data in the STACK would lead, of course, to an inaccurate calculation. The next section presents a little more sophisticated program that solves these problems.

Section 5.4. Combining STACK and VAR Data

The equation used in the next exercise has the same format as the one in Exercise 5.3. The flow diagram is shown in Figure 5.4. What is different is its flexibility for use, thanks to the way in which the program is written. The weight of each exam, its bonus points and its number of questions are entered by the user at the beginning of the program. Any change in these data for subsequent use requires only entering new constants. The program itself doesn't have to be edited or rewritten when it is used in more than one application. This new program is so much more practical than the one you've just written, there's no point in keeping the one now stored as "SC". Erase it by pressing: ', SC, ENTER, LS, PURG.

Exercise 5.4. Course Score II

Problem Statement: Construct a program for calculating a student's score for a course that allows changing constants for exam weight, number of questions and bonus points.

Solution:

A. Define the equation

$$SC = \frac{W1(A+B1)}{N1} + \frac{W2(B+B2)}{N2} + \frac{W3(C+B3)}{N3}$$

where:

A,B,C = number of correct answers ("raw score") for exams 1, 2, 3

B1,B2,B3 = bonus points for exams 1, 2, 3

N1,N2,N3 = number of questions on exams 1, 2, 3

W1,W2,W3 =weight of exams 1, 2, 3

(Hint: W1+W2+W3=100)

B. Write the program

Hint: the program will appear as:

<< → A B C 'W1*(A+B1)/N1+W2*(B+B2)/N2+W3*(C+B3)/N3'>>

Step

Press

1. LS, <<, RS, →, α, A, SPC, α, B, SPC, α, C, SPC, ', α, α, W, 1, *, LS, (), A, +, B, 1, α, RC, α, α, ÷, N, 1, SPC, +, W, 2, *, LS, (), B, +, B, 2, α, RC, α, α, ÷, N, 2, SPC, +, W, 3, *, LS, (), C, +, B, 3, α, RC, α, α, ÷, N, 3, α, RC, ENTER
2. ', α, S, α, C, STO

C. Run the program

Use the data in Table 5.5. to construct the program's constants.

Table 5.5. Constants for Course Score			
Examination	No. of Quest.	Exam Wgt (%)	Bonus Pts.
1	45	20	3
2	60	25	5
3	120	55	12

<u>Step</u>	<u>Press</u>
1.	3, ENTER, ', α , B, 1, STO
2.	5, ENTER, ', α , B, 2, STO
3.	12, ENTER, ', α , B, 3, STO
4.	45, ENTER, ', α , N, 1, STO
5.	60, ENTER, ', α , N, 2, STO
6.	120, ENTER, ', α , N, 3, STO
7.	20, ENTER, ', α , W, 1, STO
8.	25, ENTER, ', α , W, 2, STO
9.	55, ENTER, ', α , W, 3, STO

D. Use data in Table 5.6. to calculate each student's course score.

Table 5.6. Exam Raw Scores for "Course B"			
Student	Exam 1	Exam 2	Exam 3
A. Lincoln	32	50	94
G. Wash	40	49	101
T. Jeff	28	45	87
B. Franklin	25	52	102
D. Madison	39	48	96

To make the calculations:

<u>Step</u>	<u>Press</u>
1.	32, ENTER, 50, ENTER, 94, NXT, SC (read: "87.06")
2.	40, ENTER, 49, ENTER, 101, SC (read: "93.40")
3.	28, ENTER, 45, ENTER, 87, SC (read: "79.99")
4.	25, ENTER, 52, ENTER, 102, SC (read: "88.44")
5.	39, ENTER, 48, ENTER, 96, SC (read: "90.25")

The program in Exercise 5.4. is a step forward in that it lends itself to being customized without much trouble. Only the steps required to enter new constants have to be repeated with appropriate values to customize the program for a new course. The program itself doesn't have to be rewritten. But it still requires that data about each exam be

ordered in the STACK registers in a particular way. Programs described in the next section show how to structure data requests so the user is given a specific instruction about what piece of information is needed next. Once you are comfortable with these techniques, rewrite the program in Exercise 5.4. to use this strategy.

Section 5.5. Constructing Data Requests

Drawing data from the STACK is convenient for data entry, but unless the program is used fairly frequently, it's likely one's memory will soon fade about how numbers have to be listed there. Unfortunately, human memory is not as permanent as that of the HP48G. Constructing explicit requests for data input at appropriate places in a program's flow would be more than just a convenience. It would reduce chances for error. It is useful also to attract the user's attention to a displayed data request by sounding a tone when it appears in the view window. Exercise 5.5. introduces these techniques.

The program you will write next, that will help George design a floor he is laying, has a different organization than others used so far. It uses two VAR locations, one for the program itself titled "REC" (for calculating the area of a rectangle) and another titled "AREA" for the equation used for the calculation. The program's arithmetic has been purposely kept simple so programming features can be seen more clearly, as it has been for the remainder of the programs in this chapter. Figure 5.5. shows the flow diagram.

Note as you progress through the next exercise that you first write and store the equation for the calculation, then write and store a program which uses it by referring to its VAR name.

Exercise 5.5. Area of a Rectangle

Problem Statement:

George Mastic is designing the layout of a parquet floor that will be made of rectangular pieces of exotic and rare woods of different sizes. The kinds of wood he will use are particularly expensive and he wants to know the area he needs for each new piece before he cuts it. Write a program that displays data requests accompanied by a 400 Hz tone that lasts for a quarter of a second, then makes the calculation for him.

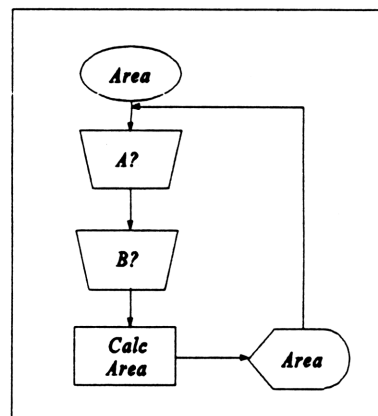


Figure 5.5. Exercise 5.5.

Solution:

A. Define the equation

$$A=L*W$$

where:

A = area of rectangle in units of entered data

L,W = length and width of rectangle in corresponding units

B. Write the equation.

Hint: The equation will appear as:

<<-> A B 'A*B'>>

Step

Press

1. LS, << >>, RS, →, α, A, SPC, α, B, ', α, A, *, α, B, ENTER
2. ', α, α, A, R, E, A, α, STO

C. Write the program.

Hint: The program will appear as:

<<"ENTER L THEN W" 440 .25 BEEP PROMPT AREA 'AREA' → TAG>>

(**Hint:** Keying this program requires entering a decimal. In the following step-by-step instructions, this is designated as " .")

Step

Press

1. LS, << >>, RS, " ", α, α, E, N, T, E, R, SPC, L, SPC, T, H, E, N, SPC, W, α, RC, 440, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, α, α, A, R, E, A, α, ', α, α, A, R, E, A, α, RC, PRG, TYPE, →TAG, ENTER
2. ', α, α, R, E, C, α, STO

(**Hint:** The program is now stored in VAR under REC.)

D. Run the program.

Use data in Table 5.7. to calculate areas for different sized pieces of wood George has available.

Table 5.7. Wood Sizes		
Piece	length	width
1	13.9 cm.	35.2 cm.
2	0.13 ft.	0.76 ft.
3	4.8 in.	3.9 in.

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | VAR, REC, 13.9, ENTER, 35.2, LS, CONT (read "AREA: 489.28") |
| 2. | REC, .13, ENTER, .76, LS, CONT (read "AREA: 0.10") |
| 3. | REC, 4.8, ENTER, 3.9, LS, CONT (read "AREA: 18.72") |

The program in Exercise 5.5. is initialized by pressing REC. It is run by entering data as instructed ("ENTER L THEN W") and telling the program to continue (LS CONT). What could be simpler? The instructions in AREA of "→A B" function similar to those in Exercise 5.4. to take data from the STACK and define them for the appropriate equation. Next, the calculation listed in AREA is made for them. Results are displayed with the tag (→TAG) then AREA as a reminder of what was calculated.

Displaying a data request and a reminder about how data are entered are useful features in Exercise 5.5. Program statements within quotation marks define exactly how the data request is displayed. The instruction PROMPT holds the displayed phrase until data are keyed and the program is instructed to continue its step-by-step flow with LS, CONT. It would also be useful to have a tone accompany the displayed request for data, if that is your choice. Tones are constructed defining the frequency of the tone (in Hz), then the length of time (in seconds) to hear the tone, followed by the instruction BEEP. The program continues after LS CONT is pressed to execute the instructions in AREA, then returns to display results with the tag "AREA".

The program REC took instructions from another VAR menu, AREA, to calculate the areas that George needs. Instructions to enter data for a calculation and the equation for using them need not exist outside of main program flow like this. They can be included just as well as steps in the program itself. There are good reasons to know how to use both types of constructions, as you'll see in writing more complicated programs later in this book. But for now, examine the next exercise which shows how to store all instructions, data requests, tone statements, calculations, display formats and other important step-by-step directions in a single program. The flow diagram is shown in Figure 5.6.

Exercise 5.6. Calculating Volumes

Problem Statement:

George has several boxes he intends to use for storage. Write a program to calculate the volumes of these boxes based on their inside measurements.

Solution:

A. Define the equation.

$$V = L * W * D$$

where:

V = volume of box in units of entered data

L, W, D = length, width and depth of box in corresponding units

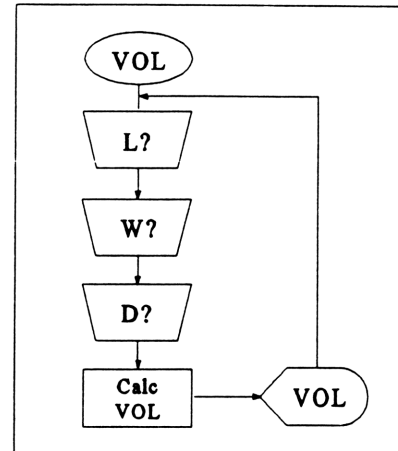


Figure 5.6. Exercise 5.6.

B: Write the program.

Hint: The program will appear as:

```
<<"INPUT L, W, D" 800 .25 BEEP PROMPT → L W D 'L*W*D' 750 .5 BEEP 'VOL'
→TAG>>
```

(Note: Keying this program requires entering decimals and commas. In the step-by-step instructions, these are designated as " , ." and " , , " respectively.)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, << >>, RS, " ", α, α, I, N, P, U, T, SPC, L, LS, ,, W, LS, ,, D, α, RC, SPC, 800, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, RS, →, α, α, L, SPC, W, SPC, D, SPC, α, ', α, α, L, *, W, *, D, α, RC, SPC 750, SPC, .5, SPC, PRG, NXT, OUT, NXT, BEEP, ', α, α, V, O, L, α, RC, PRG, TYPE, →TAG, ENTER |
| 2. | ', α, α, V, O, L, α, STO |

C. Run the program.

Use data in Table 5.8. to calculate volumes of different sized boxes.

Table 5.8. Box Sizes			
Box No.	Length	Width	Depth
1	20.9	18.3 cm.	15.0 cm.
2	0.9 ft.	1.3 ft.	1.1 ft.
3	11.9 in.	14.3 in.	10.2 in.

Step

Press

1. VAR, VOL, 20.9, ENTER, 18.3, ENTER, 15, LS, CONT (read: "VOL: 5737.05")
2. VOL, .9, ENTER, 1.3, ENTER, 1.1, LS, CONT (read: "VOL: 1.29")
3. VOL, 11.9, ENTER, 14.3, ENTER, 10.2, LS, CONT (read: "VOL: 1735.73")

When only two pieces of data (L and W in Exercise 5.5.) or three (L, W and D in Exercise 5.6.) are used in a program, it probably doesn't make much difference that the data request disappears from the view window when ENTER is pressed to enter the first number. This wouldn't work as well if more data entries were required, because the sequence of entering data could easily be forgot. Also, losing the display for a data request would be a problem if data needed to be entered sequentially as the program ran to perform interim calculations. Exercise 5.7. demonstrates techniques to modify the VOL program so that a separate data request is made for each piece of information.

Exercise 5.7. Calculating Volume - The Next Step

Problem Statement: Edit the program VOL so that each request for data is made one at time.

Solution:

- A. Define the equation. (same as in Exercise 5.6.)
- B. Editing or writing the program

There are two choices. Either use LS EDIT to edit the program VOL, or use LS PURG to erase the current program VOL from memory and write a new program. Whether edited or rewritten, the new program will appear as:

```
<<"INPUT L" 800 .25 BEEP PROMPT "INPUT W" 800 .25 BEEP PROMPT "INPUT D"
800 .25 BEEP PROMPT→L W D 'L*W*D' 750 .25 BEEP 750 .25 BEEP 'VOL'→TAG>>
```

To write the new program:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, << >>, RS, " ", PRG, NXT, IN, INPUT, α , L, RC, SPC, 800, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, RS, " ", PRG, NXT, IN, INPUT, SPC, α , W, RC, SPC, 800, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, RS, " ", PRG, NXT, IN, INPUT, SPC, α , D, RC, SPC, 800, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, RS, \rightarrow , α , α , L, SPC, W, SPC, D, α , RC, ', α , α , L, *, W, *, D, α , RC, SPC, 750, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, SPC, 750, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, SPC, ', α , α , V, O, L, α , RC, PRG, TYPE, \rightarrow TAG, ENTER |
| 2. | ', α , α , V, O, L, α , STO |

C. Run the program.

Use data in Table 5.8. to test the new program.

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | VAR, VOL, 20.9, LS, CONT, 18.3, LS, CONT, 15, LS, CONT (read: "VOL: 5737.05") |
| 2. | VOL, .9, LS, CONT, 1.3, LS, CONT, 1.1, LS, CONT (read: "VOL: 1.29") |
| 3. | VOL, 11.9, LS, CONT, 14.3, LS, CONT, 10.2, LS, CONT (read:"VOL: 1735.73") |

Exercise 5.1. presented a program that worked well enough, but it required more keystrokes than necessary to enter data and it was not interactive at all. It provided no help to the user about what data were required and when they needed to be entered. The program described in Exercise 5.7., however, was much more interactive and easy to use. The user was reminded about what inputs were required and was asked for them at appropriate times in program flow. From a user's point of view, the program is about as good as it needs to be.

One feature in the program in Exercise 5.7. is especially handy. Sounding a tone and asking for each piece of data one at a time makes it easier for the user. No effort is required to remember how many data need to be entered, what order is required for them, or where one is in the sequence of providing required information. This is not a difficult job for such a simple calculation as determining the volume of a box. Imagine, though, how much more confusing it would be and how much greater the chance for error were more data required to be entered. Signaling each request for data one at a time makes sense in many applications. The next exercise brings together several programming features introduced in this chapter. It will use program instructions as well as equations stored as VAR options.

The program's value is that it introduces how a single calculation can be made by executing instructions and using equations stored elsewhere than in the program itself. The flow diagram for the next exercise is the same as for the last two (see Figure 5.6.). To keep the slate clean, eliminate the existing program called VOL by pressing: ', VOL, LS, PURG, then, LS, CLEAR.

Exercise 5.8. VOL - Final Version

Problem Statement:

Write a program to calculate the volume of a box that demonstrates how VAR options are used as subroutines.

Solution:

A. Define the equation. (The equation is the same as in Exercise 5.6.)

B. Write the Program.

I. Construct the program to sound a tone (TON1) and to display a data request and contain a PROMPT function.

The program will appear as: << 800 .25 BEEP PROMPT >>

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, << >>, 800, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, PRG, NXT, IN, NXT, PROM, ENTER |
| 2. | ', α, α, T, O, N, 1, α, STO |

II. Construct a program to sound a tone (TON2) to signal answer display.

The program will appear as: << 750 .25 BEEP >>

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, << >>, 750, SPC, .25, SPC, PRG, NXT, OUT, NXT, BEEP, ENTER |
| 2. | ', α, α, T, O, N, 2, α, STO |

III. Construct lines to input data and list the equation (CALC).

The program will appear as: << →L W D 'L*W*D' >>

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, << >>, RS, →, α, α, L, SPC, W, SPC, D, α, ', α, α, L, *, W, *, D, ENTER |
| 2. | ', α, α, C, A, L, C, α, STO |

IV. Construct main program (VOL). The program will appear as:

```
<< "INPUT L" TON1 "INPUT W" TON1 "INPUT D" TON1 CALC TON2 TON2 'VOL'
→TAG >>
```

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, << >>, RS, " ", PRG, NXT, IN, INPUT, α, L, RC, SPC, α, α, T, O, N, 1, SPC, α, RS, " ", INPUT, α, W, RC, SPC, α, α, T, O, N, 1, SPC, α, RS, " ", INPUT, α, D, RC, SPC, α, α, T, O, N, 1, SPC, C, A, L, C, SPC, T, O, N, 2, SPC, T, O, N, 2, SPC, α, ', α, α, V, O, L, α, RC, PRG, TYPE, →TAG, ENTER |
| 2. | ', α, α, V, O, L, α, STO |

C. Use data in Table 5.8. and the steps in Stage C of Exercise 5.7. to test the program. Begin by pressing VAR, use NXT or LS, PREV to find VOL. Press VOL, then enter appropriate numbers from Table 5.8. using LS, CONT.

Section 5.6. Finding Program Errors

Writing successful programs for the HP48G, or for any other computer, requires a combination of knowledge about the machine, experience with its language structure, planning, patience, persistence and much of the time, just plain luck. Sometimes even seemingly simple programs cannot be written without errors, especially when one is first starting to learn how to construct them, or trying to learn a new programming technique. Although even just one error might be small, it can still sabotage accuracy and program flow. Also, it can be easily overlooked in just rereading one's notes.

There are several tools available to the HP48G user to help with editing a program. Once its title is listed as a VAR menu option in a directory or subdirectory, there are at least two ways to review it. The simplest is to use the EDIT feature. To do this, press VAR, then ', then press the key under the view window which corresponds to the program's title to bring it to line 1 of the view window. Next, press ENTER, then LS EDIT to see the first few lines of the program. The cursor control keys allow seeing all program statements that can be edited as required.

Another useful tool for debugging a program is to execute its lines one at a time with the programming being stopped automatically after each step. Exercise 5.9. demonstrates how to do this using the program "VOL" (Exercise 5.8.) as an example.

Exercise 5.9. Single-step Execution of a Program

Problem Statement: Examine and execute the program "VOL" one step at a time.

(**Hint:** Complete Steps in "B" of Exercise 5.8 to construct the "VOL" program if necessary. Data for "Box No. 1" in Table 5.8. are used for debugging the program.)

Solution:

<u>Step</u>	<u>Press</u>
1.	VAR, ', VOL, PRG, NXT, RUN, DEBUG
2.	SST (read "INPUT L" as the first program line). SST (hear TON1), 20.9, SST, SST (read "INPUT W"), SST (hear TON1), 18.3, SST, SST (read "INPUT D"), SST (hear TON1), 15, SST, SST (read "CALC" and "5737.05), SST (hear TON2), SST (hear TON2), SST (read "VOL"), SST (read "VOL:5737.05), SST, SST

(**Hint:** When you are finished with the program, press: LS, { }, VOL, CALC, TON1, TON2, ENTER, LS, PURG to erase all its VAR menus, then RS, CLEAR to clear the view window.)

Step 1 in the solution for Exercise 5.9. first brings the program title to line 1 of the view window. Pressing PRG, NXT, RUN provides access to functions for executing a program step-by-step. Pressing DEBUG begins the review process for the program "VOL".

Exercise 5.9. shows that each press of SST activates each program line one at a time. Using SST steps through the program giving useful insight into what's going right and what might be going wrong. Were the program under review to contain subroutines, pressing SST↓ (rather than SST) would branch the program review process to activate each of its lines in sequence, just as for the listing of the main part of the program. Pressing just SST in a program with subroutines displays the subroutine title, but does not direct the review process through it. Just a little practice with the options in the DEBUG menu soon shows how valuable they are in finding program errors.

This chapter was intended to get you started programming the HP48G. It introduced concepts for constructing flow diagrams, showed how to encode program instructions within "<< >>" and demonstrated basic procedures for entering data, making calculations with them and presenting different forms of output. It also demonstrated how to debug a program. These are important basic skills upon which you can build a variety of programs to solve equations in your own area of interest. You are able to do that now. The next chapter introduces additional techniques for program construction.

Chapter 6

Programming Strategies

The HP48G is a versatile as well as a powerful machine. As you become more familiar with its basic operations, you'll see how it allows many of the same tasks to be performed in different ways. The advantage, of course, is that you're not locked into just one "correct way" of doing things as you are with other simpler and more limited calculators and handheld computers. The styles for the exercises presented in this book, for example, show one way, but by no means the only way, to make each calculation. What's the best way for you depends on your skills in using the HP48G and on your preferences. The first two sections in this chapter introduce a few of the many alternatives for program construction and data entry. You'll see others as you progress through the chapter.

Section 6.1. Programming with RPN

All exercises presented so far used a standard algebraic statement in the body of the program. You saw this expression in each exercise just after the hint that, "The program will appear as:". For example, Exercise 5.3. showed how to calculate a percent score for a student based on the evaluation of an earned "raw score" for an exam, bonus points for it and on how much weight each exam had been assigned by the instructor. Exercise 5.3. shows the equation as:

$$CS = \frac{25(A+5)}{30} + \frac{20(B+3)}{40} + \frac{55(C+1)}{60}$$

Program structure with an algebraic statement of the equation lists the program as:

```
<<>A B C '25*(A+5)/30+20*(B+3)/40+55*(C+1)/60'>>
```

The program could have used a sequence of RPN instructions just as well as it did the algebraic statement to make the calculation. Exercise 6.1. shows how RPN is used to calculate the percent course score that was solved with an algebraic expression in Exercise 5.3.

Exercise 6.1. Programming with RPN

Problem Statement: Write a program using RPN to solve the following equation (symbol definitions are stated in Exercise 5.3.):

$$CS = \frac{25(A+5)}{30} + \frac{20(B+3)}{40} + \frac{55(C+1)}{60}$$

A. Write the program.

The program will appear as:

```
<<>> A B C <<A 5 + 25 * 30 / B 3 + 20 * 40 / + C 1 + 55 * 60 / +>>>
```

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, <<>>, RS, ➤, α, α, A, SPC, B, SPC, C, SPC, LS, <<>>, A, SPC, 5, SPC, +, SPC, 25, SPC, *, SPC, 30, SPC, ÷, SPC, B, SPC, 3, SPC, +, SPC, 20, SPC, *, SPC, 40, SPC, ÷, SPC, +, SPC, C, SPC, 1, SPC, +, SPC, 55, SPC, *, SPC, 60, SPC, ÷, SPC, +, ENTER |
| 2. | ' , α, α, C, R, P, N, α, STO |

(**Hint:** The program is now stored under VAR menu CRPN, indicating "course score in RPN".)

B: Run the program.

Use data in Table 5.4. to test the program.

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | 20, ENTER, 18, ENTER, 40, CRPN (read: "68.92") |
| 2. | 23, ENTER, 22, ENTER, 50, CRPN (read: "82.58") |
| 3. | 18, ENTER, 17, ENTER, 43, CRPN (read: "69.50") |
| 4. | 25, ENTER, 27, ENTER, 59, CRPN (read: "95.00") |
| 5. | 19, ENTER, 25, ENTER, 57, CRPN (read: "87.17") |

There is no difference, of course, between answers when the course score is calculated by a program with an algebraic statement (Exercise 5.4.) or by one which uses RPN (Exercise 6.1.). The difference is only programming style.

An important common feature for the programs in Exercises 5.4. and 6.1. is that data are first entered from lines 1 to 3 in the view window and defined as variables A, B and C,

respectively. The algebraic expression in Exercise 5.4. is included between apostrophes and the entire program is stated between the symbols << >>, as are all other programs for the HP48G whether they use either algebraic or RPN logic.

The program in Exercise 6.1. includes its sequential instructions for an RPN solution between the symbols << >> which themselves are between the program's symbols << >>. The steps are easy to follow if one remembers how data are manipulated by RPN one at a time in pairs. For example, the program reads, "Take the value for A, then add 5 to it, multiply the sum by 25, divide the product by 30, take the value for B, add 3 to it, multiply by 20, divide by 40, ..." etc. The answer finally appears at line 1 in the view window just the same as it does for Exercise 5.4.

It's important in keying the program steps for Exercise 6.1. to separate each RPN symbol by a space. For example, the statement "A 5 +" is not the same as "A5 +" or "A5+". Spaces between symbols and statements are not as important when programs are written in algebraic form, as shown in the first program listed in this chapter. That is not to say they are unimportant, it's just that they are not as important as when RPN is used. For example, the first program in this chapter shows that spaces between the symbols "A", "B" and "C" are important for defining these inputs.

Although there are only two ways in which to construct the logic for mathematical operations in programs for the HP48G (algebraic or RPN statements), there are several different ways to enter data for calculations. A couple of them are shown in the next section.

Section 6.2. Other Ways to Enter Data

How data are entered for a program is not always just a matter of style. Numbers often have to be entered in a specific sequence, as seen in some of the programs in the last chapter. You've also seen how different program constructions control for data to be entered appropriately at each stage. There are other controls you can employ. For example, sometimes it's preferable to have the machine's STACK operations disabled during the phase of data entry. Exercise 6.2. gives an example. The flow diagram is in Figure 6.1. and definitions for the exercise are shown in Figure 6.2. Figure 6.1. also shows how program flow is modified in Exercise 6.3.

As for other programs shown as exercises in this book, the calculation is purposely kept simple so that programming strategies are more easily seen. Once a few programming skills are in hand, it's always easy to expand computational complexity.

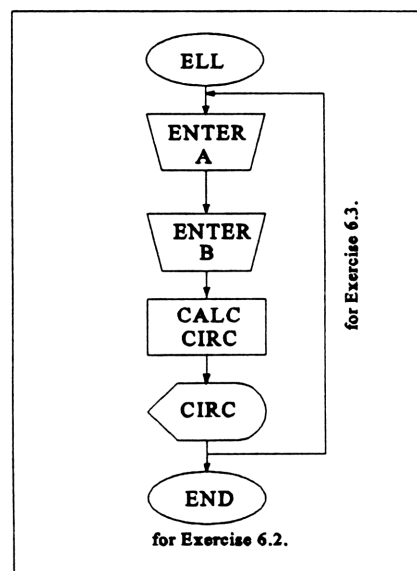


Figure 6.1. Exercise 6.2 and 6.3

Exercise 6.2. Data Entry with ENTER and INPUT

Problem Statement:

Write a program using the INPUT command which calculates the circumference of an ellipse.

A. Define the equation.

$$C = 2\pi \sqrt{\frac{A^2 + B^2}{2}}$$

where:

C = circumference of an ellipse

A = radius of short axis

B = radius of long axis

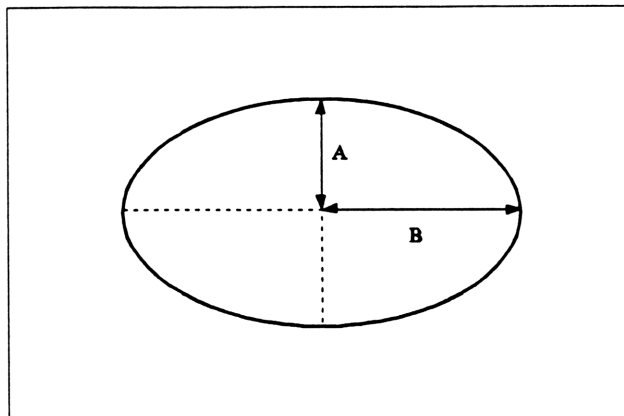


Figure 6.2. Ellipse Definitions

B. Write the program.

The program will appear as:

```
<< CLEAR TN "ENTER A" ":A:" INPUT OBJ-> TN "ENTER B" ":B:" INPUT OBJ-> ->A
B ' 2 * π * √ (( SQ (A) + SQ (B)) / 2)'->NUM TN TN 'CIRC' ->TAG>>
```

Step

Press

1. LS, <<>>, LS, CLEAR, α, α, T, N, SPC, α, RS, " ", α, α, E, N, T, E, R, SPC, A, α, RC, SPC, RS, " ", RS, :, α, A, RC, RC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ->, α, α, T, N, α, SPC, RS, " ", α, α, E, N, T, E, R, SPC, B, α, RC, SPC, RS, " ", RS, :, α, B, RC, RC, SPC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ->, RS, ->, α, α, A, SPC, B, SPC, α, ', 2, *, LS, π, *, √x, LS, (, LS, (, LS, X², α, A, RC, +, LS, X², α, B, RC, RC, ÷, 2, RC, SPC, RC, LS, ->NUM, α, α, T, N, SPC, T, N, SPC, α, ', α, α, C, I, R, C, α, RC, PRG, TYPE, ->TAG, ENTER
2. ', α, α, E, L, L, α, STO
3. LS, <<>>, 500, SPC, .1, SPC, PRG, NXT, OUT, NXT, BEEP, ENTER
4. ', α, α, T, N, α, STO

The program written at Step 3 and stored at Step 4 will appear as: << 500 .1 BEEP >>

C. Run the program

Use data in Table 6.1. to calculate the circumferences for 3 ellipses.

Table 6.1. Data for Ellipses			
	Ellipse Number		
Radius	1	2	3
A	3.28 in.	0.34 cm.	0.5 miles
B	7.19 in.	1.01 cm.	1.39 miles

Step

Press

1. VAR, ELL, 3.28, ENTER, 7.19, ENTER (read: "CIRC:35.11")
2. ELL, .34, ENTER, 1.01, ENTER (read: "CIRC:4.73")
3. ELL, .5, ENTER, 1.39, ENTER, (read: "CIRC:6.56")

Exercise 6.2. introduces several new features. The view window is cleared automatically for each calculation (by CLEAR). Also, a data request has an explanatory statement for it at the top of the view window and a constructed request at line 1, and data are entered by pressing ENTER after each new number is keyed. The command "INPUT" overrides any use of the STACK for other calculations and holds the view window display for controlled data entry only. Whatever number is entered, the command "OBJ→" converts it to a usable number and "→A B" defines the STACK contents for use in the equation. The statement "→NUM" operates as described earlier to provide a numerical evaluation for the calculation including the symbol " π ".

The program works well to make the simple calculation for the circumference of an ellipse. Having to press "ELL" each time to start the program, however, would be cumbersome if many calculations are to be made in a series. Exercise 6.3. shows how to modify the ELL program for such an operation. It is constructed so that the ELL program is automatically reactivated after each calculation.

Exercise 6.3. Automatic Program Execution

Problem Statement: Amend the program in Exercise 6.2. so that the answer for each calculation is displayed for a period of 5 seconds, then the program is automatically started again for the next calculation.

A. Amend the program.

The new program will appear as:

```
<< CLEAR TN "ENTER A" ":A:" INPUT OBJ→ TN "ENTER B" ":B:" INPUT OBJ→ →
A B ' 2 * π * √ (( SQ (A) + SQ (B)) / 2)'→NUM TN TN 'CIRC' →TAG 3 DISP 5 WAIT
ELL >>
```

<u>Step</u>	<u>Press</u>
1.	, ELL, ENTER, LS, EDIT
2.	(move the cursor to be between "→TAG" and ">>" at the end of the program), SPC, 3, PRG, NXT, OUT, DISP, 5, PRG, NXT, IN WAIT, α, α, E, L, L, ENTER

B. Use data in Table 6.1. to test the program.

<u>Step</u>	<u>Press</u>
1.	ELL, 3.28, ENTER, 7.19, ENTER (read: "CIRC:35.11"), .34, ENTER, 1.01, ENTER (read: "CIRC:4.73"), .5, ENTER, 1.39, ENTER (read: "CIRC:6.56")

(**Hint:** To suspend program operation, press CANCEL twice.)

The amended program instruction "3 DISP" in Exercise 6.3. directs the computer to display the calculated answer at line 3 in the view window. The instruction "5 WAIT" controls program operation to stop for 5 seconds, after which the next program instruction "ELL" directs the program to start over again. Displaying the answer at some other line would be accomplished, of course, by using some appropriate number other than 3 before "DISP". Similarly, holding the answer display for some other length of time would be achieved by using some appropriate number other than 5 before "WAIT".

There are other ways to modify program operation. For example, the program could be amended by the steps: PRG, NXT, RUN, HALT to read:

```
<< CLEAR TN "ENTER A" ":A:" INPUT OBJ→ TN "ENTER B" ":B:" INPUT
OBJ→ → A B ' 2 * π * √ (( SQ (A) + SQ (B)) / 2)' →NUM TN TN 'CIRC' →TAG
HALT ELL>>
```

This new program would calculate the circumference of an ellipse and display the answer at line 1, just the same as in Exercise 6.2. Pressing: LS, CONT would start the program over again without having to press ELL. In contrast to the operation of the program in Exercise 6.2., the keyboard and view window would be free for other calculations in the meantime. When these other operations are complete, pressing LS, CONT starts another calculation. The ELL program waits patiently in the wings while you use the computer for other tasks.

Being able to program the same task in different ways is an important, powerful and attractive feature of the HP48G. It opens many doors for alternative approaches to problems and it gives the user a variety of tools and options for program design. Having so many choices, though, may be itself a little challenging to the beginner, leading to confusion, feeling overwhelmed and being discouraged trying to find the "right way" to do things.

There is an easy way out of any such discomfort. Be courageous in trying different

programming techniques - there is nothing to be lost. The computer's internal logic will check your programs for syntax and programming errors. Not being intimidated by the machine's BEEP when you try to store a set of instructions in which there is one or more errors can be a first step to learning something new. Any program structure you are comfortable with that yields accurate results and doesn't take too much memory or time to run can be justifiably defended as a "good program". The fact that someone else solves the same problem(s) in another way makes their program different, but not necessarily better. Exercise 6.4. demonstrates several different but equally good ways to solve the same simple problem.

Exercise 6.4. Programming Options

The purpose of this exercise is to show there are several, equally good ways to solve an equation using the HP48G. After each of the solutions has been constructed, key a number, then press A, B, C, D, or E to activate each type of solution.

Problem Statement: Write a program to solve: $X=2(Y+8)$

Solution No. 1: The program will appear as: $\langle\langle\rangle\rightarrow Y \langle\langle Y 8 + 2 * \rangle\rangle$

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, $\langle\langle\rangle\rangle$, RS, \rightarrow , α , Y, LS, $\langle\langle\rangle\rangle$, α , Y, SPC, 8, +, 2, *, ENTER |
| 2. | ' , α , A, STO |

Solution No. 2: The program will appear as: $\langle\langle\rangle\rightarrow Y '2 * (Y+8)'\rangle\rangle$

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, $\langle\langle\rangle\rangle$, RS, \rightarrow , α , Y, SPC, ' , 2, *, LS, () , α , Y, +, 8, ENTER |
| 2. | ' , α , B, STO |

Solution No. 3: The program will appear as: $\langle\langle\text{DUP } 8 + 2 * \rangle\rangle$

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, $\langle\langle\rangle\rangle$, LS, STACK, NXT, DUP, 8, +, 2, *, ENTER |
| 2. | ' , α , C, STO |

Solution No. 4: The program will appear as: $\langle\langle 'Y' \text{ STO } Y 8 + 2 * \rangle\rangle$

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, $\langle\langle\rangle\rangle$, ' , α , Y, RC, STO, α , Y, SPC, 8, +, 2, *, ENTER |
| 2. | ' , α , D, STO |

Solution No. 5: The program will appear as: << 'Y' STO Z →NUM>>

<u>Step</u>	<u>Press</u>
1.	LS, << >>, ', α, Y, RC, STO, α, Z, LS, →NUM, ENTER
2.	', α, E, STO
3.	LS, << >>, α, Y, SPC, 8, +, 2, *, ENTER
4.	', α, Z, STO

The program keyed at Step 3 will appear as: <<Y 8 + 2 *>>

The next section shows how to apply many of the elementary programming skills you now have to the construction of more sophisticated problem solving. It introduces techniques for using decision making, branching, testing and constructing loops within a program.

Section 6.3. Decision Structures

A unique and one of the more important features of the handheld computers that distinguishes it from a calculator is not only that it can be programmed to solve equations, but these programs can also be written to make decisions at one place or another in their flow. Program operation and the pattern of its flow can be made to depend intimately on how decision statements are constructed and where they appear in the program.

This section introduces the two basic decision statements "IF/THEN/END" and "IF/THEN/ELSE/END". The logic of these statements is simple. The computer is instructed: IF (some condition exists), THEN (perform this operation); and IF (some condition exists), THEN (perform this operation), ELSE (do something else). Although this structure is easily used to control symbolic and mathematical operations in a program, the syntax may get a little convoluted and hard to follow in the program itself. The basic idea, though, is no more complicated than the one a driver makes, for example, in the thought process: IF (the gas gauge reads near empty), THEN (stop at the next gas station), ELSE (drive on by). How a program uses this basic logic structure is demonstrated in Exercises 6.5. and 6.6.

Exercise 6.5. Controlled Choices

Problem Statement:

Calculating a student's score for a course ("SC"; Exercise 5.4.) is usually not the last step in academic evaluation. A letter grade, typically on a scale of "A to F", is commonly used to report each student's status at the end of the term. Write a program which automatically assigns such a letter grade using the criteria listed in Table 6.2. The flow diagram is shown in Figure 6.3.

Table 6.2. Grading Criteria	
Minimum % SC	Grade
68	A
74	B
70	C
51	D
<51	F

Solution:

A. Write the Program

The program will appear as:

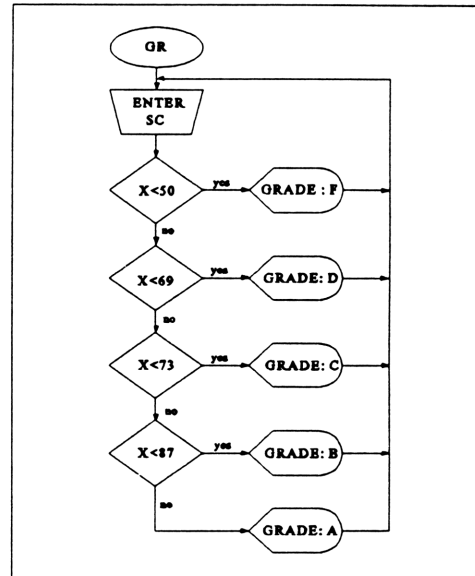


Figure 6.3. Exercise 6.5.

```
<< →X <<CLEAR IF 'X≤50' THEN "F" P END IF 'X≤69' THEN "D" P END IF 'X≤73'
THEN "C" P END IF 'X≤87' THEN "B" P END "A" P>> >>
```

Step

Press

1. LS, << >>, RS, →, α, X, LS, << >>, LS, CLEAR, PRG, BRCH, IF, IF, ', α, X, PRG, TEST, ≤, 50, RC, PRG, BRCH, IF, THEN, RS, " ", α, F, RC, SPC, α, P, END, IF, ', α, X, PRG, TEST, ≤, 69, RC, PRG, BRCH, IF, THEN, RS, " ", α, D, RC, SPC, α, P, END, IF, ', α, X, PRG, TEST, ≤, 73, RC, PRG, BRCH, IF, THEN, RS, " ", α, C, RC, SPC, α, P, END, IF, ', α, X, PRG, TEST, ≤, 87, RC, PRG, BRCH, IF, THEN, RS, " ", α, B, RC, SPC, α, P, END, RS, " ", α, A, RC, SPC, α, P, ENTER
2. ', α, α, G, R, α, STO
3. LS, <<>>, RS, " ", α, α, G, R, A, D, E, SPC, I, S, α, RC, PRG, TYPE, →TAG, PRG, NXT, RUN, HALT, α, α, G, R, ENTER
4. ', α, P, STO

The program keyed at Step 3 will appear as:

```
<<"GRADE IS" →TAG HALT GR>>
```

B. Use data in Table 6.3. to test the program.

Table 6.3. Scores for "Course C"	
Student	Course
A. Einstein	68.2
H. Volta	75.4
B. Lagosi	63.1
J. Hoffa	71.0
S. Goldwin	89.3

Step	Press
1.	68.2, GR (read: "GRADE IS D")
2.	75.4, GR (read: "GRADE IS B")
3.	63.1, GR (read: "GRADE IS D")
4.	71, GR (read: "GRADE IS C")
5.	89.3, GR (read: "GRADE IS A")

The "IF/THEN/END" structure for the program in Exercise 6.5. sets the language not only for the decision to be made, but also for its limits of operation within the program. If the condition of the test ("IF $X \leq NN.N$ ") is tested to be true, then program flow continues with the instructions immediately following the "THEN" statement. If the condition of the test is not met, then program flow continues with the first instruction immediately past the next "END" statement. Simple enough.

For example, when the student's percent "Score for the Course" is keyed, the program defines the score as "X" and the first test is to ask "is $X \leq 50$ ". If the test is true, then the alpha symbol "F" is constructed and the next instruction (P) directs program flow to use the phrase "GRADE IS". Program flow is then halted to await the next score to be presented for test. If any test is not true, flow skips to the next program instruction immediately after the next "END" to try the next test. The program is, of course, searching for an appropriate statement to display depending on the level of the student's score. If all tests are false for an entered score, then the good news "GRADE IS A" is displayed as the only possible last option in the logic series.

There are many applications for the "IF/THEN/END" simple structure. It can be used to set or clear a FLAG (as described in the next chapter), to define or redefine some variable, to control program flow (as it did in the last exercise) and to perform many other useful services. Some of the sample programs described in Chapter 8 show how nested versions of the "IF/THEN/ELSE" structure gives additional power in controlling program flow.

Logic in a program may be a little more complicated than performing an action if only one condition exists, as in the "IF/THEN/END" construction. There might be another option on which to act. The "IF/THEN/ELSE/END" logic structure gives this additional choice, as demonstrated in Exercise 6.6. The flow diagram is in Figure 6.4.

Exercise 6.6. Addition with Conditional Branching

Problem Statement. Design a program that requests a series of numbers of indefinite length, then displays their sum when the last number has been entered.

Solution.

A. Define the equation.

$$TOTAL = N1 + N2 \dots Nn$$

B. Write the Program

Where:

TOTAL = sum of entered numbers

$N_{1,2,n}$ = entered numbers

Step

Press

1. LS, <<>>, 0 (zero), SPC, ', α , S, RC, STO, α , α , R, E, Q, ENTER
2. ', α , α , T, O, T, A, L, α , STO

Program keyed at Step 1 will appear as: <<0 'S' STO REQ>>

3. LS, <<>>, LS, CLEAR, α , α , T, N, SPC, α , RS, " ", α , α , E, N, T, E, R, SPC, N, α , RC, SPC, RS, " ", RS, ::, α , N, RC, RC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ➔, SPC, ', α , N, RC, SPC, STO, α , α , T, S, T, ENTER
4. ', α , α , R, E, Q, α , STO

Program keyed at Step 3 will appear as:

<<CLEAR TN "ENTER N" ":N:" INPUT OBJ➔ 'N' STO TST>>

5. LS, <<>>, PRG, BRCH, IF, IF, α , N, SPC, 0 (zero), SPC, PRG, TEST, =, PRG, BRCH, IF, THEN, α , A, ELSE, α , B, END, ENTER
6. ', α , α , T, S, T, α , STO

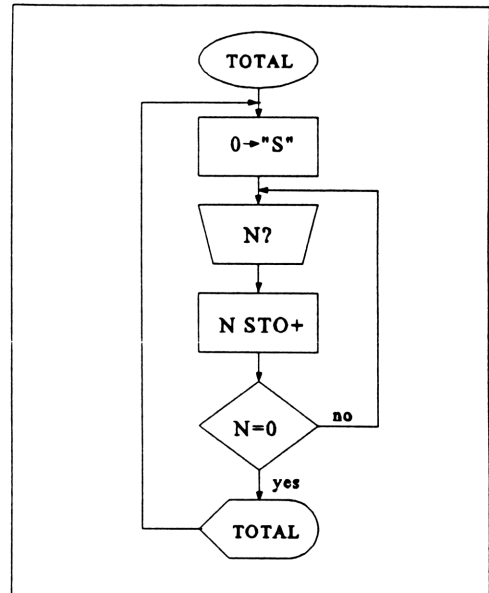


Figure 6.4. Exercise 6.6.

Program keyed at Step 5 will appear as: <<IF N 0 = = THEN A ELSE B END>>

7. LS, <<>>, LS, CLEAR, PRG, NXT, OUT, CLLCD, α , α , T, N, SPC, T, N, SPC, S, α , SPC, ', α , α , T, O, T, A, L, α , RC, PRG, TYPE, \rightarrow TAG, 3, PRG, NXT, OUT, DISP, 2, PRG, NXT, IN, WAIT, α , α , T, O, T, A, L, ENTER
8. ', α , A, STO

Program keyed at Step 7 will appear as:

<<CLEAR CLLCD TN TN S 'TOTAL' \rightarrow TAG 3 DISP 2 WAIT TOTAL>>

9. LS, <<>>, α , N, SPC, ', α , S, RC, LS, MEMORY, ARITH, STO+, α , α , R, E, Q, ENTER
10. ', α , B, STO

Program keyed at Step 9 will appear as: <<N 'S' STO+ REQ>>

11. LS, <<>>, 800, SPC, .3, PRG, NXT, OUT, NXT, BEEP, ENTER
12. ', α , α , T, N, α , STO

Program keyed at Step 11 will appear as: <<800 .3 BEEP>>

D. Run the program.

The following example calculates the sum of the numbers: 3.9, 8.7, 4.0, -10.2, 3.9, -2.1 and 7.9.

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | VAR, TOTAL, 3.9, ENTER, 8.7, ENTER, 4, ENTER, 10.2, +/-, ENTER, 3.9, ENTER, 2.1, +/-, ENTER, 7.9, ENTER, 0, ENTER (read: "TOTAL:16.10") |

(Hint: Pressing CANCEL twice exits the program.)

The flow diagram for Exercise 6.6. shows how each associated program is addressed sequentially for decision making, branching, display control and other operations. First a VAR menu entitled "S" is constructed and a zero is stored in it. The next instruction directs program flow to "REQ" where a data request is constructed. Whatever number is entered is stored in "N". Program flow then is directed to "TST" where a decision is made based on the test of N. If the entered number is not equal to zero, the program activates "B" where N

is stored additively in "S". The next number in the series is requested by once again using the instructions in "REQ". When an entered number is zero, instructions in "A" control displaying "S" which is the sum of all numbers entered so far. The program then pauses for 2 seconds and begins again with the activation of "TOTAL". The first step is to "clear" "S" by storing a zero to make ready for the next calculation. Pressing CANCEL twice returns the computer to its original display and the TOTAL program is deactivated.

There are a couple of instructions in this program that require clarification. The instructions constructed at Step 5 of "PRG, BRCH, IF, IF" begins the "IF/THEN/ELSE/END" logic required for program flow. The instruction "CLLCD" at Step 7 blanks the screen display so that the calculated answer can be read more easily. The instruction "STO+" at Step 9 demonstrates how VAR menus can be used for programmed arithmetic operations, not just for the storage of constants.

Although any number or other symbols could be used to terminate data entry, zero is used for this function in Exercise 6.6. The reason, of course, is that any number might be a legitimate data entry in a program designed to be able to add both positive as well as negative numbers of any magnitude. Zero would never be used in this way, so it provides a convenient basis for testing when the data entry phase is over.

The most likely point for confusion in the TOTAL program is the use of the symbol "=" in the test "IF N 0 =" at Step 5. The HP48G (and many other computers too) makes an important distinction between the symbols "=" and "=". The symbol "=" designates a statement of fact. For example, "A=B" is read as, "A is equal to B". The symbol "=" is used in tests that ask a question. For example, "A B =" asks, "is A equal to B?". This distinction takes a little getting used to for the beginner, but it is an important one.

Both the IF/THEN/END and the IF/THEN/ELSE/END constructions can be used to test more than just the simple statements like "is X less than N?", "is X equal to zero?", "is X greater than Y?", and the like. They can be used to test FLAG status, as described in the next chapter, and to test several different combinations of conditions at once. For example, program branching can be directed on the basis of not only if one, but if two conditions are met, if either one or both conditions are met, if one but not the other condition is met, and if neither of two conditions is met. These more complicated conditional tests are structured using the instructions ("AND", "OR", "XOR" and "NOT") listed in "page" 2 of the menu accessed by PRG, TEST. Exercise 6.7. gives a demonstration. The flow diagram for the exercise is in Figure 6.5.

Exercise 6.7. Tests with Double Conditionals

Problem Statement: Harvey Pascal needs a program to determine if the oil pressure and temperature in a system he is monitoring are within safe limits. Oil pressure must be between 50 and 150 PSI and temperature must be no less than 100°F and no more than 250°F. Write a program that displays appropriate messages about these parameters based on the evaluation of entered data.

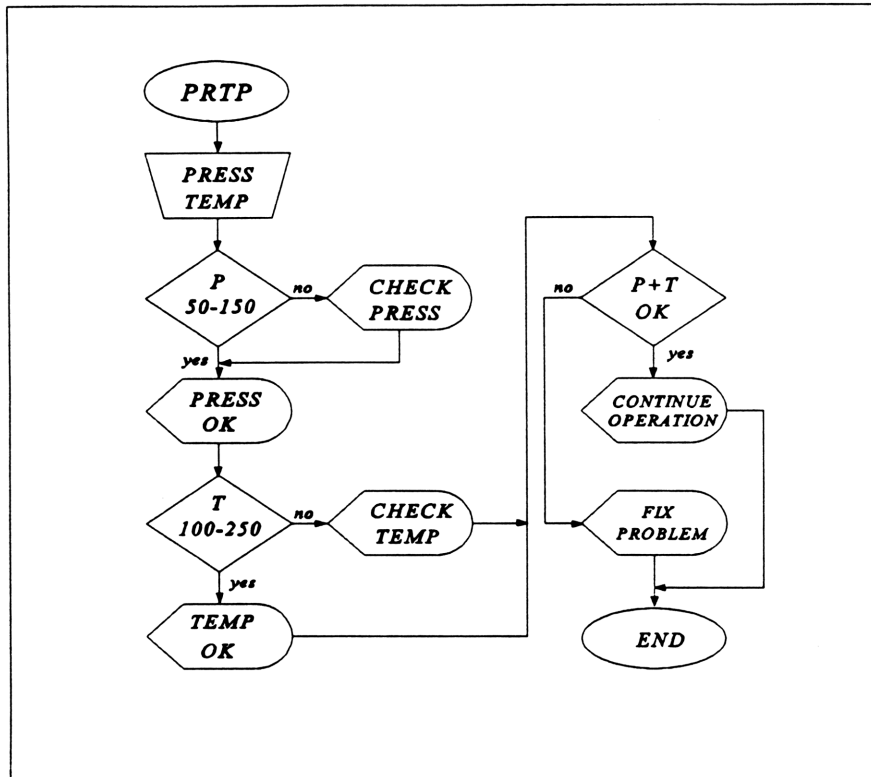


Figure 6.5. Exercise 6.7.

Solution

A. Write the Program.

(Hint: the symbol " ← " indicates "next line". It is keyed by first pressing RS then the decimal key.)

The program will appear as:

```

<< CLEAR TN 0 'T' STO 0 'P' STO "ENTER PRESS AND TEMP" {" :PRESS(PSI): (←)
:TEMP(F):" { 1 0 } V } INPUT OBJ→ 'T' STO 'P' STO IF 'P>50' 'P<150' AND THEN
CLLCD TN "PRESS OK" 1 'P' STO W ELSE CLLCD TN "CHECK PRESS" W END IF
'T<100' 'T>250' OR THEN TN "CHECK TEMP" W ELSE "TEMP OK" 1 'T' STO W END
IF 'P'=1' 'T'=1' AND THEN "CONTINUE OPERATION" W ELSE "FIX PROBLEM"
W END >>
  
```

Step

Press

1. LS, << >>, LS, CLEAR, α, α, T, N, SPC, α, 0 (zero), SPC, ', α, T, RC, STO, 0 (zero), SPC, ', α, P, RC, STO, RS, "", α, α, E, N, T, E, R, SPC, P, R, E, S, S, SPC, A, N, D, SPC, T, E, M, P, α, RC, SPC, LS, { }, RS, " ", RS, ::, α, α, P, R, E, S, S, LS, (), P, S, I, α, RC, RC, RS, ←, RS, ::, α, α, T, E, M, P, LS, (), F, α, RC, RC, RC, SPC, LS, { }, 1, SPC, 0 (zero), RC, α, V, RC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ→, SPC, ', α, T, RC, STO, ', α, P, RC, STO, PRG, BRCH, IF, IF, ', α, P, PRG, TEST, >, 50, RC, SPC, ', α, P, <, 150, RC, SPC, NXT, AND, PRG, BRCH, IF, THEN, PRG, NXT, OUT, CLLCD, α, α, T, N, SPC, RS, " ", P, R, E, S, S, SPC, O, K, RS, " ", SPC, 1, SPC, α, ', α, P, RC, STO, α, W, PRG, BRCH, IF, ELSE, PRG, NXT, OUT, CLLCD, α, α, T, N, SPC, RS, " ", C, H, E, C, K, SPC, P, R, E, S, S, RS, " ", α, RC, α, W, PRG, BRCH, IF, END, IF, ', α, T, PRG, TEST, <, 100, RC, SPC, ', α, T, >, 250, RC, SPC, PRG, TEST, NXT, OR, PRG, BRCH, IF, THEN, α, α, T, N, SPC, RS, " ", C, H, E, C, K, SPC, T, E, M, P, RS, " ", SPC, W, α, PRG, BRCH, IF, ELSE, RS, " ", α, α, T, E, M, P, SPC, O, K, α, RC, SPC, 1, SPC, α, ', α, T, RC, STO, α, W, PRG, BRCH, IF, END, IF, ', α, P, PRG, TEST, ==, 1, RC, ', α, T, ==, 1, RC, SPC, NXT, AND, PRG, BRCH, IF, THEN, RS, " ", α, α, C, O, N, T, I, N, U, E, SPC, O, P, E, R, A, T, I, O, N, α, RC, SPC, α, W, ELSE, RS, " ", α, α, F, I, X, SPC, P, R, O, B, L, E, M, α, RC, α, W, END, ENTER
2. ', α, α, P, R, T, P, α, STO
3. LS, << >>, 3, PRG, NXT, OUT, DISP, 1, PRG, NXT, IN, WAIT, PRGM, NXT, OUT, CLLCD, α, α, T, N, ENTER
4. ', α, W, STO

The program keyed at Step 3 will appear as:

```
<< 3 DISP 1 WAIT CLLCD TN >>
```

5. LS, << >>, 1000, SPC, .2, PRG, NXT, OUT, NXT, BEEP, ENTER
6. ', α, α, T, N, α, STO

The program keyed at Step 5 will appear as: << 1000 .2 BEEP >>

B. Use data in Table 6.4. to test the program.

Table 6.4. Pressures and Temperatures		
Test No.	Pressure (PSI)	Temp (deg.F)
1	75	150
2	180	110
3	83	275
4	30	265
5	192	87
6	100	203

<u>Step</u>	<u>Press</u>
1.	VAR, PRTP, 75, DC, 150, ENTER (read: ""PRESS OK"/"TEMP OK"/CONTINUE OPERATION")
2.	PRTP, 180, DC, 110, ENTER (read: "CHECK PRESS"/"TEMP OK"/"FIX PROBLEM")
3.	PRTP, 83, DC, 275, ENTER (read: "PRESS OK"/"CHECK TEMP"/"FIX PROBLEM")
4.	PRTP, 30, DC, 265, ENTER (read: "CHECK PRESS"/"CHECK TEMP"/"FIX PROBLEM")
5.	PRTP, 100, DC, 203, ENTER (read: "PRESS OK"/"TEMPOK"/"CONTINUE OPERATION")

There are two main purposes for presenting Exercise 6.7. One is to demonstrate how two different conditions are tested together with the IF/THEN/ELSE/END construction. The other is to show yet another style for data entry. From the beginning of the program, VAR menus "T" and "P" are created and have zeros stored in them, then data for pressure and temperature are requested. An advantage of this data entry style is that both pieces of information remain displayed until ENTER is pressed. Only a slight modification of the display instructions in the program accommodates more than two data entries.

The first decision the program makes is to test entered data for pressure to determine if they are within the stipulated operating limits of 50 to 150 PSI. The construction " 'P>50' 'P<150' AND" allows the program to display "PRESS OK" only if both tests are true. The general test is: (condition 1) (condition 2) AND. Otherwise, the phrase following ELSE is activated. Limits for entered temperature data are tested in an analogous way, but one which allows the phrase following THEN to be displayed only if one of the two tests is true. The general construction "(condition 1) (condition 2) OR" controls flow in this part of the program.

The program has a third conditional test to make. The number "1" is stored in either "P" or "T" if "CHECK ..." is the correct option to be displayed earlier in the program. The phrase "CONTINUE OPERATION" is selected only if both "P" and "T" contain the number "1". If neither does, or if only one of them does, then the phrase "FIX PROBLEM" is selected. This makes sense in the context of the problem because "CONTINUE OPERATION" is reasonable only if both pressure and temperature are tested to be within operating limits of the system that Harvey is monitoring. Also, he will have to "FIX PROBLEM" if there is a discrepancy in either the pressure or the temperature controls. Testing the status of a VAR menu by seeing what number is stored there was a convenience for deciding which phrase to display at the end of the program in Exercise 6.6. A similar and more powerful test, but one which is much simpler and easier to use, is available by using FLAGS. Their operation is described in the next chapter.

Certainly, flexibility and power are salient features of the HP48G. The more you

discover of its characteristics, the more options you will have for making it do precisely and quickly what you want. The conditional tests of one kind or another reviewed so far allow you to control one or more operations depending on a single condition (Exercise 6.4.), 2 conditions (Exercise 6.5.) and on a variety of relationships between 2 conditions (Exercise 6.6.)

There is one more kind of conditional statement for which you are sure to find many applications. It is one in which a series of cases is tested within the logic structure of a single nested set of conditions. The logic involves the sequence: CASE/THEN/END/END. As for so many of the HP48G's features, a simple demonstration is valuable. Exercise 6.8. introduces a technique for sequentially testing a series of conditional statements of your own choice within a single command sequence. The flow diagram for the example is shown in Figure 6.6.

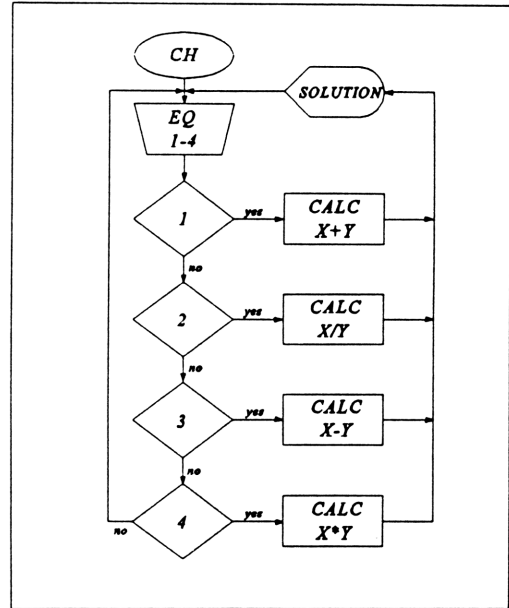


Figure 6.6. Exercise 6.8.

Exercise 6.8. Tests with Many Conditions

Problem Statement: Write a program using a "CASE/THEN/END/END" construction that allows the user to select a particular equation to be solved, then asks for data. The selected equations will either add, divide, subtract or multiply the entered data.

Solution:

A. Write the programs

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, <<>>, LS, CLEAR, RS, " ", α, α, E, Q, LS, DROP, LS, (), 1, -, 4, α, RC, RC, PRG, NXT, IN, NXT, PROM, RS, →, α, Z, LS, <<>>, PRG, BRCH, CASE, CASE, α, Z, SPC, 1, PRG, TEST, NXT, SAME, PRG, BRCH, CASE, THEN, α, A, END, α, Z, SPC, 2, PRG, TEST, NXT, SAME, PRG, BRCH, CASE, THEN, α, B, END, α, Z, SPC, 3, PRG, TEST, NXT, SAME, PRG, BRCH, CASE, THEN, α, C, END, α, Z, SPC, 4, PRG, TEST, NXT, SAME, PRG, BRCH, CASE, THEN, α, D, END, END, PRG, NXT, OUT, CLLCD, 4, DISP, 2, PRG, NXT, IN, WAIT, α, α, S, E, L, ENTER |
| 2. | ' , α, α, S, E, L, α, STO |

The program keyed at Step 1 will appear as:

```
<<CLEAR "EQ?(1-4)" PROMPT - Z << CASE Z 1 SAME THEN A END Z 2 SAME  
THEN B END Z 3 SAME THEN C END Z 4 SAME THEN D END END CLLCD 4 DISP  
2 WAIT SEL >> >>
```

3. LS, <<>>, RS, " ", α , α , X, LS, DROP, α , RC, PRG, NXT, IN, NXT, PROM, ', α ,
X, RC, STO, RS, " ", α , α , Y, LS, DROP, α , RC, PROM, ', α , Y, RC, STO,
ENTER
4. ', α , V, STO

The program keyed at Step 3 will appear as:

```
<< "X?" PROMPT 'X' STO "Y?" PROMPT 'Y' STO >>
```

5. LS, <<>>, LS, CLEAR, α , V, SPC, ', α , α , X, +, Y, α , RC, LS, \rightarrow NUM, ENTER
6. ', α , A, STO

The program keyed at Step 5 will appear as: <<CLEAR V 'X+Y' \rightarrow NUM>>

7. LS, <<>>, LS, CLEAR, α , V, SPC, ', α , α , X, \div , Y, α , RC, LS, \rightarrow NUM, ENTER
8. ', α , B, STO

The program keyed at Step 7 will appear as: <<CLEAR V 'X/Y' \rightarrow NUM>>

9. LS, <<>>, LS, CLEAR, α , V, SPC, ', α , α , X, -, Y, α , RC, LS, \rightarrow NUM, ENTER
10. ', α , C, STO

The program keyed at Step 9 will appear as: <<CLEAR V 'X-Y' \rightarrow NUM>>

11. LS, <<>>, LS, CLEAR, α , V, SPC, ', α , α , X, *, Y, α , RC, LS, \rightarrow NUM, ENTER
12. ', α , D, STO

The program keyed at Step 11 will appear as: <<CLEAR 'X*Y' \rightarrow NUM>>

B. Use the programs to solve for X when X=19 and Y=2.56.

<u>Step</u>	<u>Press</u>
1.	VAR, SEL, 1, LS, CONT, 19, LS, CONT, 2.56, LS, CONT (read: "21.56")
2.	2, LS, CONT, 19, LS, CONT, 2.56, LS, CONT (read: "7.42")
3.	3, LS, CONT, 19, LS, CONT, 2.56, LS, CONT (read: "16.44")
4.	4, LS, CONT, 19, LS, CONT, 2.56, LS, CONT (read: "48.64")

There is nothing at all fancy about how data are requested or entered in the program in Exercise 6.8. All techniques have been demonstrated in earlier exercises. What is new is the structure and function of the conditional phrases. The construction "CASE ... END" surrounds a series of conditional tests each of which selects an appropriate equation based on the number entered at the beginning of the program.

Analogous to the "IF/THEN/END" statement, at whatever level in the conditional tests the entered number (stored as Z) is the SAME as the test number (1 to 4), the instruction immediately following the next THEN statement is activated. For example, if the phrase "Z 3 SAME" is true, then the program branches to use the equation in C to subtract value Y from value X. If the phrase is not true, program flow continues beyond the next END to test the next conditional statement.

There are many built-in tests in the HP48G. In addition to the decision structures introduced in this section, you are able to construct several different types of controlled loops to activate selected sections of your program. A couple of them are introduced in the next section.

Section 6.4. Loop Structures

It will be important at times to have operations in a program repeated during a calculation. For example, if an error is encountered in a data entry for a program, it would be useful to flash several times "ERROR", or some similar message, perhaps accompanied by a repetitive warning tone. One way to do this, of course, is just to rewrite the appropriate lines so they appear as successive strings of instructions as the program runs. A more efficient program writing strategy is to construct commands in the program that control automatic reactivation of selected program lines in a looping fashion.

There are 2 built-in logic structures in the HP48G to do this. One is used to construct a "definite loop" and the other is used in an "indefinite loop" structure. "Definite loop" operations are designed to reactivate a specific sequence of program lines a specified number of times. This is done as a one-at-a-time operation using a "START/NEXT" format. A typical instruction would be read as: "complete (designated steps) 6 times, then go on to something else". Another technique is to use a step interval other than 1 in a "START/STEP" instruction.

In a typical application, such an instruction might be read as: "increase a number by its square root 3 times, then go on to something else". In either case, the basic structure of this function is the same, for example: (number to start at) (number to end at) START (repetitive operation to be performed) NEXT (operation to be continued). Exercise 6.9. provides an example of a controlled loop. Figure 6.7. shows the flow diagram for the calculation.

Exercise 6.9. Controlled Looping

Problem Statement:

Write a program that tests an entered number and will accept it as a legitimate data entry only if it is less than 10. If the number is equal to or greater than 10, display "ERROR" and sound a tone 3 times. If the tested number is less than 10, indicate that it is acceptable as a data input.

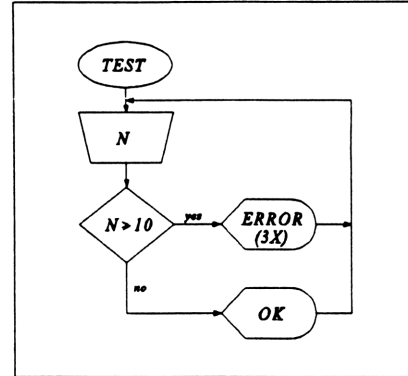


Figure 6.7. Exercise 6.9.

A. Write the program.

The program will appear as:

```
<<"ENTER TEST NUMBER " ":N:" INPUT OBJ-> ->X <<IF 'X ≥ 10' THEN CLEAR 1 3
START 900 .1 BEEP "ERROR" 3 DISP .2 WAIT NEXT "TRY AGAIN" 3 DISP 1 WAIT
CLEAR TEST ELSE 1000 .2 BEEP "NUMBER OK" 3 DISP 1 WAIT CLEAR TEST
END>> >>
```

Step

Press

1. LS, <<>>, RS, " ", α, α, E, N, T, E, R, SPC, T, E, S, T, SPC, N, U, M, B, E, R, α, RC, SPC, RS, " ", RS, :, α, N, RC, RC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ->, RS, ->, α, X, LS, <<>>, PRG, BRCH, IF, IF, ', α, X, PRG, TEST, ≥, 1 0, RC, PRG, BRCH, IF, THEN, LS, CLEAR, 1, SPC, 3, PRG, BRCH, START, START, 900, SPC, .1, PRG, NXT, OUT, NXT, BEEP, RS, " ", α, α, E, R, R, O, R, α, RC, 3, PRG, NXT, OUT, DISP, .2, PRG, NXT, IN, WAIT, PRG, BRCH, START, NEXT, RS, " ", α, α, T, R, Y, SPC, A, G, A, I, N, α, RC, SPC, 3, PRG, NXT, OUT, DISP, 1, PRG, NXT, IN, WAIT, LS, CLEAR, α, α, T, E, S, T, α, PRG, BRCH, IF, ELSE, 1000, SPC, .2, PRG, NXT, OUT, NXT, BEEP, SPC, RS, " ", α, α, N, U, M, B, E, R, SPC, O, K, α, RC, SPC, 3, PRG, NXT, OUT, DISP, 1, PRG, NXT, IN, WAIT, LS, CLEAR, α, α, T, E, S, T, α, PRG, BRCH, IF, END, ENTER
2. ', α, α, T, E, S, T, α, STO

B. Run the program. Test the following numbers: 2, 12, 4, 15.

Step

Press

1. VAR, TEST, 2, ENTER (read: "NUMBER OK")
 2. 12, ENTER (read: "ERROR/TRY AGAIN")
 3. 4, ENTER (read: "NUMBER OK")
 4. 15, ENTER (read: "ERROR/TRY AGAIN")
-

Exercise 6.9. demonstrates how the instructions "1 3 START ...NEXT" controls the sequence of tones and the display of "ERROR", after which program flow continues beyond "NEXT" to display "TRY AGAIN", then restart the program for the next number to test. This "definite loop" structure is contained within a controlled branch statement, but can exist just as well at any other place in the program, depending on the requirements for choices to be made.

Controlled loops function not only to repeat a section of a program a designated number of times, but may also be used in an "indefinite loop" structure to repeat program lines until some variable has reached a designated level. An example is shown in Exercise 6.10. for which Figure 6.8. presents a flow diagram.

Exercise 6.10. "DO/UNTIL"

Problem Statement: Write a program that increments a number by a designated interval until it has become greater than 10.

A. Write the program.

The program will appear as:

```
<< DUP → X << 0 'N' STO DO CLLCD X 1+
'N' STO+ N 3 DISP .5 WAIT UNTIL N 10 >
END CLEAR "OVER" >> >>
```

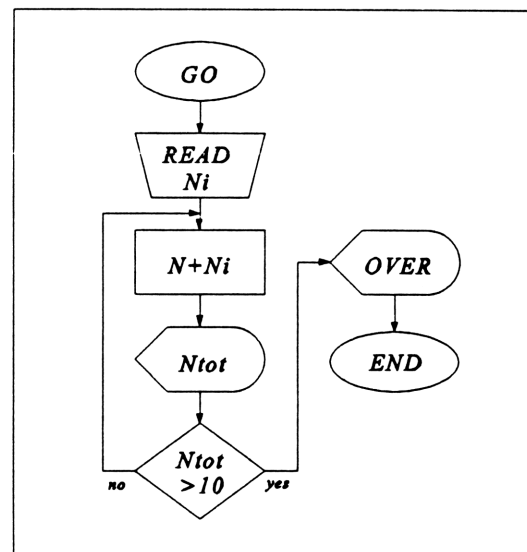


Figure 6.8. Exercise 6.10.

Step

Press

1. LS; <<>>, LS, STACK, NXT, DUP, RS, →, α, X, SPC, LS, <<>>, 0 (zero), SPC, ', α, N, RC, STO, PRG, BRCH, DO, DO, PRG, NXT, OUT, CLLCD, α, X, SPC, 1, SPC, +, SPC, ', α, N, RC, LS, MEMORY, ARITH, STO+, α, N, SPC, 3, PRG, NXT, OUT, DISP, .5, PRG, NXT, IN, WAIT, PRG, BRCH, DO, UNTIL, α, N, SPC, 10, SPC, PRG, TEST, >, PRG, BRCH, DO, END, LS, CLEAR, RS, " ", α, α, O, V, E, R, α, RC, ENTER
2. ', α, α, G, O, α, STO

B. Run the program

Step

Press

1. VAR, 0 (zero), GO (read: "1, 2, 3 ... 11"/"OVER")
 2. 1, GO (read: "2, 4, 6 ... 12"/"OVER")
 3. 2, GO (read: "3, 6, 9, 12"/"OVER")
 4. .25, GO (read: "1.25, 2.50, ... 11.25"/"OVER")
-

The first instructions in the program in Exercise 6.10. duplicate (DUP) the number at Line 1 in the view window then define it as "X". VAR menu "N" is then created to contain a zero. The "DO/UNTIL" loop starts with the instructions to clear the display (CLLCD), add 1 to whatever number was entered at the beginning of the program, store the sum in "N", then display that sum at Line 3. Program flow is delayed for 0.2 seconds, then the "DO" loop is reactivated "UNTIL" the number in "N" is greater than 10. When this limit is exceeded, program flow goes beyond the END statement to clear the screen and display "OVER".

Whatever message is displayed last, just before "OVER" is shown, depends, of course, on the interval of the count entered at the beginning of the program. The instruction to discontinue activation of the loop guards only when the sum in "N" is greater than 10. It does not control how much greater than 10, or how much greater than 10 it needs to be.

For those preferring a different notation style, the program in Exercise 6.8. runs just as well if it is constructed, for example, as:

```
<< DUP →X << 0 'N' STO DO CLLCD 'X+1' →NUM 'N' STO+ N 3 DISP .5 WAIT
UNTIL 'N'>10' END CLEAR "OVER" >>>>
```

Similar alternatives can be constructed for any other programs in which there are conditional statements or which contain statements to direct arithmetic operations. Looping in a program is particularly valuable when it is used interactively with manual data entries. Properly constructed, these program statements automatically keep track of one or more variables, either singly or in combination, as in Exercise 6.8., where the program continues to run and as you continue to enter data for it to use.

Exercise 6.11. demonstrates this feature as it is employed with "WHILE/REPEAT/END" control statements. Similar to the "DO/UNTIL" strategy, they will monitor one or more limits set in a program, continue to REPEAT specified program lines WHILE these limits have not yet been met, but like the other control statements, follow program instructions beyond the END statement when the internal test of the limits indicate they have been exceeded. The flow diagram for Exercise 6.11. is shown in Figure 6.9.

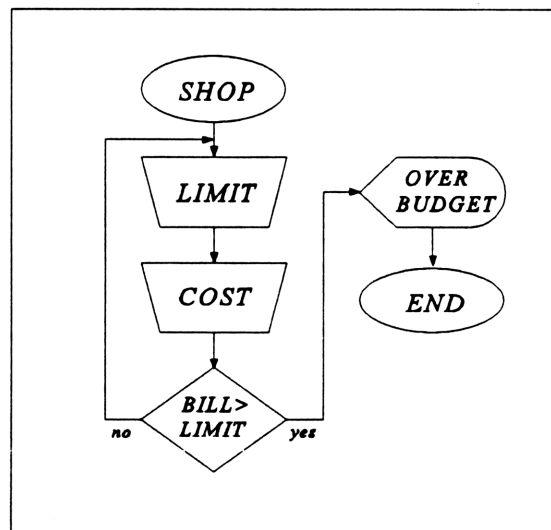


Figure 6.9. Exercise 6.11.

Exercise 6.11. Keeping in the Budget**Problem Statement:**

George enjoys buying outdoor equipment and sports clothes by shopping from the catalog supplied by a national distributor. Write a program that will allow him to enter each purchase he wants to make, then signal when he is over the budget limit he has set for himself.

Solution:**A. Write the program.**

The program will appear as:

```
<< CLEAR 0 'TEST' STO T "LIMIT?" PROMPT 'L' STO WHILE 'TEST<L' REPEAT T  
"COST?" PROMPT 'TEST' STO+ END T T "OVER BUDGET" >>
```

Step**Press**

1. LS, <<>>, LS, CLEAR, 0 (zero), SPC, ', α , α , T, E, S, T, α , RC, STO, α , T, SPC, RS, " ", α , α , L, I, M, I, T, LS, DROP, α , RC, PRG, NXT, IN, NXT, PROM, ', α , L, RC, STO, PRG, BRCH, WHILE, WHILE, ', α , α , T, E, S, T, α , PRG, TEST, <, α , L, RC, PRG, BRCH, WHILE, REPEA, α , T, RS, " ", α , α , C, O, S, T, LS, DROP, α , RC, PRG, NXT, IN, NXT, PROM, ', α , α , T, E, S, T, α , RC, LS, MEMORY, ARITH, STO+, PRG, BRCH, WHILE, END, α , α , T, SPC, T, α , RS, " ", α , α , O, V, E, R, SPC, B, U, D, G, E, T, ENTER
2. ', α , α , S, H, O, P, α , STO
3. LS, <<>>, 2000, SPC, .1, PRG, NXT, OUT, NXT, BEEP, ENTER
4. ', α , T, STO

The program keyed at Step 3 will appear as: << 2000 .1 BEEP >>

B. Use the program.

George has \$200 in his budget for catalog purchases. He selects the items in Table 6.5. What's the last purchase he can afford?

Item	Cost (\$)
hat	19.98
gloves	25.50
boots	73.19
sweater	35.02
coat	40.50
socks	9.35

Step

Press

1. VAR, SHOP, 200, LS, CONT, 19.98, LS, CONT, 25.50, LS, CONT, 73.19, LS, CONT, 35.02, LS, CONT, 40.50, LS, CONT, 9.35, LS, CONT (read: "OVER BUDGET". George will have to buy his socks some other time.)

It would have been a lot easier for George just to buy a \$2.00 calculator (or even better, a 50 cent pencil) to keep track of his purchases, but then you'd never have had such an easy-to-understand example for how to use the "WHILE/REPEAT/END" operations in a program. Once the limit for his shopping adventure is entered using LS, CONT, the cost for each item he wishes to buy is entered in the same way. The accumulated bill (stored in "TEST") is compared to the budget limit (stored in "L") repetitively (by REPEAT) WHILE the conditional test "TEST<L" is true. When it is no longer true, the phrase "OVER BUDGET" is displayed as the program automatically branches beyond its END statement.

As good a way as any to end this chapter is to present a guessing game that demonstrates controlled looping and shows a couple of new programming techniques in the bargain. Some would argue that learning to use the HP48G is a good enough guessing game all by itself, but the demonstration will serve other purposes too. Exercise 6.12. uses instructions that trigger the HP48G's built-in "random number generator", uses its operations for rounding a number and shows how to decrement automatically by a unit of 1 a number stored in a VAR memory location. Figure 6.10. shows the flow diagram.

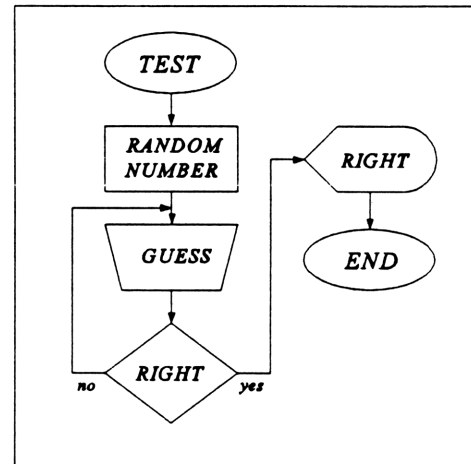


Figure 6.10. Exercise 6.12.

Exercise 6.12. Guess Again

Problem Statement: Write a program that controls a guessing game in which a number between 0 and 10 is entered in an attempt to match a hidden "randomly generated" number. The program recognizes when the correct match has been made and displays a score (maximum of 10) based on the number of required trials.

A. Write the programs.

Step

Press

1. LS, <<>>, MTH, NXT, PROB, RAND, 10, *, 0 (zero), MTH, REAL, NXT, NXT, RND, ', α , X, RC, STO, 11, SPC, ', α , S, RC, STO, α , α , G, S, ENTER
2. ', α , α , T, E, S, T, α , STO

The program keyed at Step 1 will appear as:

```
<<RAND 10 * 0 RND 'X' STO 11 'S' STO GS>>
```

3. LS, <<>>, PRG, BRCH, DO, DO, ', α, S, RC, LS, MEMORY, ARITH, DECR, α, α, T, N, α, LS, CLEAR, RS, " ", α, α, G, U, E, S, S, SPC, N, LS, (), 0 (zero), -, 10, α, RC, RC, SPC, RS, " ", RS, ::, α, N, RC, RC, PRG, NXT, IN, INPUT, PRG, TYPE, OBJ→, ', α, N, RC, STO, PRG, BRCH, DO, UNTIL, α, α, N, SPC, X, α, PRG, TEST, NXT, SAME, PRG, BRCH, DO, END, α, α, T, N, SPC, T, N, SPC, α, RS, " ", α, α, R, I, G, H, T, -, N, SPC, W, A, S, SPC, α, RC, SPC, α, N, +, RS, " ", α, α, S, C, O, R, E, LS, =, α, RC, SPC, α, S, +, ENTER
4. ', α, α, G, S, α, STO

Program keyed at Step 3 will appear as:

```
<<DO 'S' DECR TN CLEAR "GUESS N(0-10)" ":N:" INPUT OBJ→ 'N' STO UNTIL N X  
SAME END TN TN "RIGHT-N WAS " N + "SCORE=" S +>>
```

5. LS, <<>>, 1500, SPC, .2, PRG, NXT, OUT NXT, BEEP, ENTER
6. ', α, α, T, N, α, STO

Program keyed at Step 5 will appear as: <<1500 .2 BEEP>>

B. Run the program

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | VAR, TEST, (key a number between 0 and 10), ENTER, (if guess is incorrect, key another number and press ENTER. If guess is correct, read: "RIGHT-N WAS (correct number)" "SCORE=(score)"). |
| 2. | TEST (for next game) or CANCEL, CANCEL to discontinue the program. |

The program in Exercise 6.12. first generates a "randomly selected" number between 0 and 1 which is then multiplied by 10 and rounded (by "0 RND") to have no digits to the right of the decimal (the instruction "2 RND" would have rounded to 2 digits to the right of the decimal, etc.). Using the "INPUT" instruction provides a particularly useful way to accept the user's guess of the test number. The VAR menu "X" in which the test number is stored cannot be easily seen without interrupting the game.

The "DO/UNTIL/END" sequence first compares the entered number with the test one and continues the "DO" loop (which controls the request for another number) until they are tested to be the same. With a correct guess, the test number is revealed along with the user's score. One point is automatically subtracted (by "DECR"; the instruction "INCR" would

increase a number stored in a designated VAR menu by one) from the VAR menu "S" where the score is kept each time the "DO" loop is activated. More than 10 incorrect guesses made by a confused and unlucky user who enters the same number(s) more than once in a game, earns well-deserved negative points. What game strategy guarantees at least breaking even with the laws of chance? What would be a convincingly consistent score to document "extra-sensory perception"?

The branching and looping statements introduced in this section may be a little difficult to build into your own programs at first. Just a little practice, though, will no doubt soon make them clear and easy to use. They add considerable power and flexibility to programs and are well worthwhile learning.

The next chapter describes an additionally powerful programming tool, the use of FLAGS. How you take advantage of the ones dedicated to machine use, as well as those which are "user-defined" will soon become clear.

Chapter 7

FLAGS

Section 7.1. The Basic Concept

The last chapter gave several examples of the HP48G's flexibility when you take advantage of its branching and looping structures in combination with its conditional tests. All these tests, though, depended on evaluating a specific condition. It might have been the equality of two numbers or words, as in the constructions "X Y = =", "X = = Y", "X Y SAME", or in comparing the magnitude of two numbers, as in the statements "A B <", "A>B", or in some other similar test. Common to all of them, you tested the relationship between two explicitly defined entities. When these tests were used within the logic structures of "IF/THEN/END", "IF/THEN/ELSE/END" or other program devices, they gave considerable freedom for making a calculation or solving a problem. But that's only part of it.

Despite the advantage, you were limited to options in conditional tests which gave choices only in "closed-ended" comparisons. Only very specific values for A, B, X, Y and other variables could be compared. The HP48G allows you to do much more by allowing for program branching and looping on the basis of "open-ended" comparisons. You are not limited to comparing just stored numbers, words and phrases. Using FLAGS, you can compare the status of any situation you wish to define for your program. It need not be numerical, nor must it have an alphanumeric symbol designed for it. Too good to be true? Not at all.

Exploring these potentially unlimited opportunities to control program flow requires understanding the basic concepts behind FLAGS and knowing how to use them. Fortunately, both tasks are easy. Also making it easy, you have ready access to the different ways to test and use FLAGS through the 6 statements listed on "page" 3 of the menu targeted by "PRG, TEST, NXT, NXT. These functions are SF, CF, FS?, FC?, FS?C, and FC?C.

Calling a signaling device in the HP48G a "FLAG" makes a great deal of sense. Everyone is familiar with the more general use of flags as devices to convey information. National flags designate a country's identity, the checkered flag at the race track signals the winner of a competition, and the red flag at a construction site shows a dangerous situation. Each of these signaling instruments tells us something about what's going on in our world. In the same way, each FLAG in a program tells the computer what's going on in its world as the program makes its way through its many steps.

Flags familiar to us in everyday use, though, sometimes convey additional information than just "something is going on". Quite often, they also tell us what to do depending on a particular condition at any one time. For example, if you were driving down a road and saw a construction worker with a flag, not only would you suspect that "something is going on", but also you'd be aware from the position of the flag to know what to do next. If the flag were lowered to the worker's side, most of us would interpret the signal to mean "proceed with caution". If the flag were raised, most of us would recognize the flag's status means to stop or go in some designated direction.

FLAGS in a computer program provide similar signals. There are two instructions that work for all FLAGS. One is to indicate whether the FLAG is SET (FS?) or whether it is CLEAR (FC?). There are also two instructions to change a FLAG's status. One will SET it (FS) and one will CLEAR it (FC). There are two others which first check the FLAG's status and then clear it (FS?C and FC?C). But what FLAGS are they controlling and what is the function of any FLAG no matter what its status? The next section helps explain it all.

Section 7.2. Where FLAGS Are And What They Do

There are two categories of FLAGS in the HP48G. You can test and change the status of any of the FLAGS in either of them. Sixty four of them, designated "-1 to -64", are called "system-defined FLAGS". Another 64, designated "1 to 64" are called "user-defined FLAGS". Although you can manipulate the status (either SET or CLEAR) of the "system-defined FLAGS", each has a predetermined and specific meaning in controlling a built-in function of the computer. For example, FLAG -41 when SET displays the clock in a 24 hour format. When this FLAG is CLEAR, a 12 hour clock display is shown. The HP48G handbook gives a full listing of the "system-defined FLAGS". The "user-defined FLAGS" 1 to 64 have no intrinsic meaning for computer function and you can use any of them in any way you want. This chapter gives several demonstrations.

Determining whether a FLAG is SET or CLEAR (using FS? or FC?) requires understanding a simple code. The answer "yes" or "no" to the questions about FLAG status will be displayed as a number at Line 1 in the view window. If the answer to FS? (or to FC?) is true (that is, "the (designated) FLAG is SET (or CLEAR)"), the number 1 will be shown. Zero will appear if the answer is "no" either to the question FS? or FC?. For example, if FLAG 10 were set, "1" would appear in the view window after keying "10, FS?". Were it clear, "0" would be shown. Exercise 7.1. gives an example of how to test and control FLAG status. Its simple lessons are applicable to the use of all FLAGS, both system-defined and user-defined.

Exercise 7.1. FLAG Test and Manipulation: The Basics

(Hint: The "system-defined" FLAG -51 used in this exercise controls whether a period (when CLEAR) or a comma (when SET) serves as a decimal indicator. To get started, first press: 51, +/-, PRG, TEST, NXT, NXT, CF so that FLAG -51 is clear to begin the exercise.)

Problem No. 1: Test the status of FLAG -51.

<u>Step</u>	<u>Press</u>
1.	51, +/-, PRG, TEST, NXT, NXT, FS? (read: "0", indicating "No, FLAG -51 is not SET")
2.	RS, ARG, FC? (read: "1", indicating "Yes, FLAG -51 is CLEAR")

(**Hint:** FLAG -51 status must have been CLEAR if it was tested at Step 1 not to be SET, since SET or CLEAR are the only statuses any FLAG can have. The operation "RS, ARG" in Step 2 brings to Line 1 the last "argument" that was the number "-51.")

Problem No. 2: Seeing the effect of having Flag -51 clear and changing its status.

<u>Step</u>	<u>Press</u>
1.	RS, MODES, RC, 4, OK, OK, 1, 2, 3, 4, (decimal), 5, 6, 7, 8, ENTER (read: "1,234.5678")
2.	51, +/-, PRG, TEST, NXT, NXT, SF (read: "1.234,5678")
3.	RS, ARG, CF (set the original status for FLAG -51 and read: "1,234.5678")

Not much of a problem to see how changing the status of a "system-defined FLAG" affects computer operation, as illustrated in Exercise 7.1. But what sense is there to make of the "user-defined FLAGS" for which there is no proscribed functions? How do you use them if they can mean anything you want them to? Section 7.3. gives an explanation and some examples.

Section 7.3. "User-defined FLAGS"

By the same token that the "system-defined FLAGS" have functions for the operating system of the HP48G, the "user-defined FLAGS" have functions defined by you. They can be used to designate any condition you want. One could, for example, use a FLAG to let the computer know certain data have been entered (or not entered), whether a particular calculation has been made (or not yet made) as program flow continues, or to signal any other circumstance you define. You need to define what numbered FLAG (from 1 to 64) you want to use to signal a particular condition, and you need to designate what specific meaning does the FLAG have when it is SET and what meaning does it have when it is CLEAR.

The simplicity and ease of use, as well as the power of FLAGS is most easily seen with an example, like the one presented in Exercise 7.2. The flow diagram is in Figure 7.1. In view of the considerable experience gained so far with key entry skills, the "Step ... Press" instructions for keying program statements are omitted in the exercises in this chapter. Those for problem solutions, however will be kept.

Exercise 7.2. FLAGS: A Simple Application

Problem Statement:

Mary Worsted runs a retail fabric store which has a liberal policy about returned purchases. She'll accept any returned item in good condition, but requires that the customer may have to pay some of the paperwork expenses for processing the refund. She knows, for example, it'll cost on the average \$1.50 of employee time to locate the record of a sale if the customer doesn't have a receipt. Also, it costs her 75 cents if the payment was made by check. She needs to pass these expenses on to the customer. If the person has a receipt and didn't pay by check, they, of course, get a full refund. Write a program that will calculate how much of the original purchase the customer gets in refund.

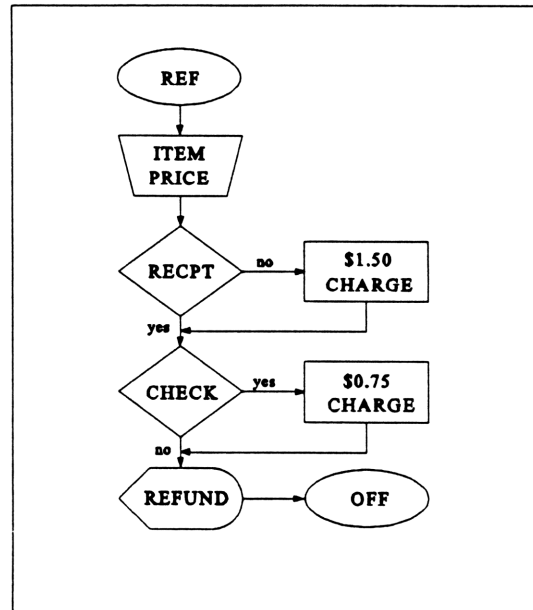


Figure 7.1. Exercise 7.2.

Solution:

A. Define FLAG status

Table 7.1. FLAG Status		
FLAG	SET	CLEAR
1	no receipt	receipt
2	check	no check

B. Write the program. (Hint: To key "\$": RS, CHARS, +64, +64, (highlight "\$"), ECHO,CANCEL)

Program: REF

```

<<TN 1 CF 2 CF 2
FIX "ITEM PRICE?"
":Price $:" INPUT
OBJ-> 'P' STO TN
"RECEIPT?(Y or N)"
"" INPUT
IF "N" SAME
THEN 1 SF
END TN

"CHECK?(Y or N)" ""
INPUT
IF "Y" SAME
THEN 2 SF
END
IF 1 FS?
THEN 'P' 1.5 STO-
END
IF 2 FS?
THEN 'P' .75 STO-
END TN TN CLLCD
"REFUND=$" P + 3
DISP 3 WAIT TN TN
TN OFF>>
-----
Program:TN
<< 900 .2 BEEP>>
  
```

C. Run the program.

Determine the correct refund for each of the following customers:

Table 7.2. Customer Refunds			
Customer	Item Price	Receipt	Check
M. Teresa	\$187.32	yes	no
Z. Gabor	\$519.50	no	yes
M. Jackson	\$324.19	yes	yes
W. Clinton	\$132.00	no	no

- | Step | Press |
|------|---|
| 1. | VAR, REF, 187.32, ENTER, α , Y, ENTER, α , N, ENTER (read: "REFUND=\$187.32") |
| 2. | ON, REF, 519.50, ENTER, α , N, ENTER, α , Y, ENTER (read: "REFUND=\$517.25") |
| 3. | ON, REF, 324.19, ENTER, α , Y, ENTER, α , Y, ENTER, (read: "REFUND=\$323.44") |
| 4. | ON, REF, 132, ENTER, α , N, ENTER, α , N, ENTER (read:"REFUND=\$130.50") |

Exercise 7.2. gives a little more practice for key entry skills and shows a couple of more programming techniques. Most important, though, it introduces the basic ideas of how "user-defined FLAGS" are used. If you saw in this exercise how program flow is directed depending on the SET or CLEAR status of a particular FLAG, you have a more than good enough grasp of their operating principle. That's an important step because all FLAG use is no more than a variation on the central theme demonstrated in this simple exercise.

There are a few features of the program that deserve additional comment. Because the status of FLAGS 1 and 2 in Exercise 7.2. are going to have special meaning in the program (Table 7.1.), it's important they be controlled right from the beginning. The instructions "1 CF 2 CF" initially place both FLAGS to CLEAR. The next program instruction (2 FIX) controls the view window's display to be two digits to the right of the decimal, an appropriate choice since data are processed in the program as dollars and cents.

If a customer doesn't have a receipt, FLAG 1 is SET (by "1 SF"), and FLAG 2 is SET if they paid by check. The customer's bill is calculated by subtracting \$1.50 from their purchase price if they've lost the receipt and subtracting an additional 75 cents if they paid by check. The calculations are done automatically as program flow goes through "IF/THEN/END" statements controlled by whether FLAGS 1 or 2 are SET ("1 FS?" and "2 FS?"). If the customer had a receipt and didn't pay by check, they receive a full refund.

There are a couple of extra features to watch for as the program runs. You may have noticed the numbered indicators appear at the top of the view window when either FLAG 1 or FLAG 2 became SET, and how they disappeared at the beginning of the program as they were automatically cleared. Setting FLAGS 3 to 5 would also be signaled by the appearance of appropriate numbers in the view window. None of the other FLAGS ("user-defined" 6 to 64 or "system-defined" -1 to -64" will have their SET status indicated in this way. A short program in Section 7.5. is designed to disclose FLAG status. As a last point, you were no doubt aware of the action produced by the instruction "OFF" at the end of program.

Section 7.4. Complex Choices with FLAGS

Exercise 7.2. showed how the numerical contents of a VAR menu was modified based on decisions made in the program and signaled by FLAGS. Useful though it may be, only the target of the menu itself was affected by conditions of the program. Exercise 7.3. demonstrates a more involved and practical use of FLAGS. The equations used in this example are purposely simple so attention is not diverted from how FLAGS direct program execution. The flow diagram for the exercise is shown in Figure 7.2.

Exercise 7.3. Weight Predictions

Problem Statement:

A zoologist needs to make a series of calculations using two different equations that predict body weight changes for two species of small animal based on their nutrition and environment. Calculations will be made either in metric or in English units. The equations are:

$$A=(T+U)/V$$

$$B=(T/W)-X$$

where:

A,B = species

T,U,V,W,X = environmental and nutritional data

Comment:

Values for T, U and W are different each time the program is run from the beginning, but they are constants in any uninterrupted series of calculations. This means that when the program is run for the first time, data will be requested for T, U and W. If a second, third or more calculations are made without starting the program over, the computer needs to "remember" that if it already has these data, it won't ask for them again. Values for V and X will be different for each calculation and have to be requested each time a calculation is made whether the program is started at the beginning or not. A test is necessary also for whether solutions need to be displayed in metric or in English units.

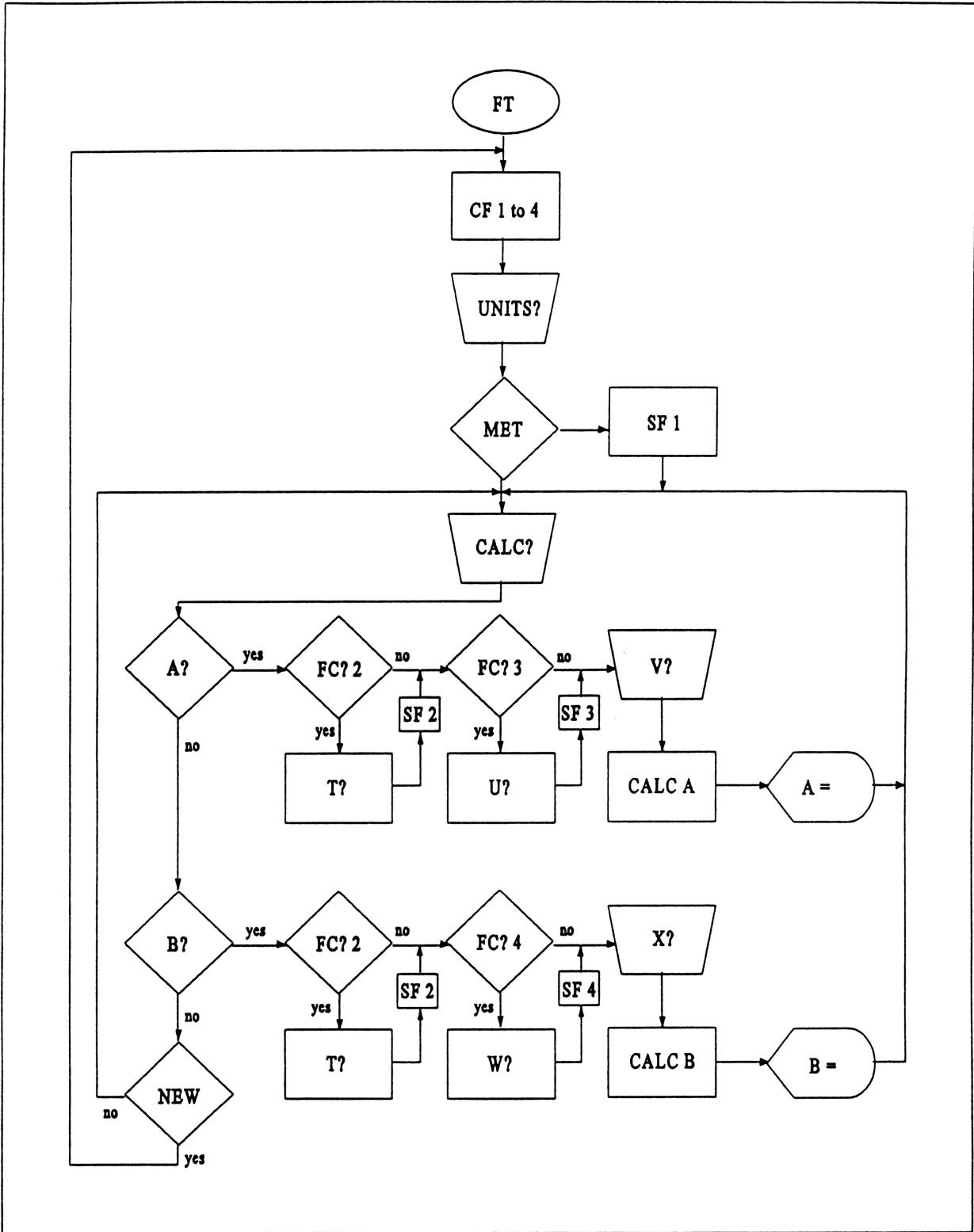


Figure 7.2. Exercise 7.3.

Solution:

A. Define FLAG status.

Four FLAGS will be used to keep track of what data are required and how calculations are to be made. FLAG status is described in Table 7.3.

Table 7.3. FLAG Status		
FLAG	SET	CLEAR
1	metric	English
2	T stored	no data
3	U stored	no data
4	W stored	no data

B. Write the program.

Program:FT

```
<< 4 'F' STO
DO F CF 'F' DECR
UNTIL F 0 ==
END TN
"MET or ENG?" ""
INPUT
IF "MET" ==
THEN 1 SF
END CAL >>
```

Program:CAL

```
<< TN
"CALC?(A,B or NEW)"
"" INPUT 'N' STO
CASE N "A" SAME
THEN A
END N "B" SAME
THEN B
END N "NEW"
SAME
THEN FT
END CAL
END >>
```

Program:A

```
<< TR
IF 3 FC?
THEN UA
END T U + TN
"ENTER V" ":V:"
INPUT OBJ→ / D >>
```

Program:B

```
<< TR
IF 4 FC?
THEN WA
END T W / TN
"ENTER X" ":X:"
INPUT OBJ→ - D >>
```

Program:TR

```
<< IF 2 FC?
THEN TA
END >>
```

Program:TA

```
<< TN "ENTER T"
":T:" INPUT OBJ→
'T' STO 2 SF >>
```

Program:UA

```
<< TN "ENTER U"
":U:" INPUT OBJ→
'U' STO 3 SF >>
```

Program:WA

```
<< TN "ENTER W"
":W:" INPUT OBJ→
'W' STO 4 SF >>
```

Program:TN

```
<< 900 .2 BEEP >>
```

Program:D

```
<< TN TN "WGT=" SWAP
+
IF 1 FS?
THEN "GM."
ELSE "OZ."
END + CLLCD 4
DISP 3 WAIT CAL >>
```

C. Run the program.

Table 7.4. Body Weight Data					
	T	U	V	W	X
Species A	19.3	7.2	10.1	6.4	0.2
Species B	4.8	9.7	1.8	1.1	5.6

Problem No. 1:

Step 1. Calculate A in metric units using data for Animal No. 1.

Press: VAR, NXT, FT, α , α , M, E, T, ENTER, α , A, ENTER, 19.3, ENTER, 7.2, ENTER, 10.1, ENTER (read: "WGT=2.62 GM.")

Problem No. 2: Calculate A again when V=15.3. Press: α , A, ENTER, 15.3, ENTER (read: "WGT=1.73 GM.")

Problem No. 3: Calculate B using data for Species A. Press: α , B, ENTER, 6.4, ENTER, .2, ENTER (read: "WGT=2.82 GM.")

Problem No. 4: Start a new calculation for B in English units using data for Species B. Press: α , α , N, E, W, ENTER (read "MET or ENG?"). Press: α , α , E, N, G, ENTER, α , B, ENTER, 4.8, ENTER, 1.1, ENTER, 5.6, ENTER (read: "WGT=-1.24 OZ.")

Problem No. 5: Calculate B again when X=0.8. Press: α , B, ENTER, .8, ENTER (read: "WGT=3.56 OZ.")

Problem No. 6: Calculate A using data for \species B. Press: α , A, ENTER, 9.7, ENTER, 1.8, ENTER (read: "WGT=8.06 OZ.")

(Hint: Press CANCEL to discontinue program.)

Although a little long and somewhat convoluted, Exercise 7.3. is a good example not only of how to use FLAGS, but also of how to structure subroutines to make a program run efficiently. The program's use of FLAGS demonstrates how different user-defined conditions are tested and how program flow is controlled as a function of the result of each test. For example, FLAGS 2, 3 and 4 keep track of whether data required for a calculation have been

entered. As shown in the program's flow diagram (Figure 7.2.), they direct program flow to subroutines to request appropriate data, if necessary. Also, FLAG 1 determines the selection of the correct units for displaying data. Although there are other programming strategies to provide similar control in a program, using FLAGS is the easiest.

Since FLAGS will be used in Exercise 7.3., it is good practice to control their initial status at the very beginning of the program. The first few instructions CLEAR FLAGS 1 to 4. This could be done by a series of instructions similar to "1 CF 2 CF etc." that would serve the purpose, but it might be impractical were many FLAGS used in a program. A more direct way is, as shown at the beginning of the program, to create a "DO/UNTIL/END" loop in which FLAGS 1 to 4 are cleared one at a time in the loop until the counter in "F" indicates no additional FLAGS remain to be cleared.

The first time calculations are made in Exercise 7.3. (at Steps 1 and 5), data are requested for T, then either for U and V (for calculating A) or for W and X (for calculating B). The second time a calculation is made (at Steps 2 and 6), data are not requested for T because FLAG 2 has now been SET. The rule established at the very beginning of the program (Table 7.3.) was that data for T had already been entered when FLAG 2 was set. The request for T is bypassed when "2 FS?" is tested to be true. Similarly, data requests for U and W are activated only the first time the program is run because the SET status of FLAGS 3 and 4 respectively, now indicate new data are not required.

Data for V and X are always requested because they remain variables no matter how many times the program is used. When "NEW" is entered in response to "CALC?(A, B or NEW)", the program is begun again and will ignore any previously entered data for T, U and W, even though "T", "U" and "W" remain as options in the VAR menu. The status of FLAG 1 determines whether answers will be displayed in metric or in English units by taking advantage of the "IF/THEN/ELSE/END" construction at the end of the program.

There are a couple of new programming features in Exercise 7.3. DECR automatically decreases by one whatever number is in a designated storage location, which for this program was the number stored in "F". Even though this number was originally "4" (placed there by: "4 'F' STO"), it soon became "3" by the steps "'F' DECR" the first time the "DO...UNTIL" loop was activated. It became "2", then "1" and finally "0" under the repetitive activation of "'F' DECR". Similar control is provided by the instruction "INCR" which increments by one the number in a designated storage location. Exercise 7.4. gives an example for the use of "INCR".

There is also a new application of the program structure: " "(statement)" " " INPUT" which appears in the subroutine "CAL". Not just one alpha entry is evaluated, but several of them are tested within the "CASE N "A" SAME ... END" subroutine. This is a convenient and easy way to allow the program to search among several possible data inputs in order to find the correct subroutine in which to make a particular calculation.

Another new feature is built into the display subroutine, "D", to append a displayed answer with an appropriate set of units. Program statements determine both the units and the instructions for how to show them. "'WGT=' SWAP +" indicates that the answer display

will be prefixed by "WGT=" and followed by either "GM." or "OZ.", depending on the status of FLAG 1. "4 DISPLAY" indicates which line in the view window to use to display the constructed answer phrase. "3 WAIT" controls how long it will be displayed.

It's easy enough to know the status of FLAGS 1, 2, 3, 4 and 5 because of the annunciators shown at the top of the view window when each is set. It's a little less easy to know the status of the other "user-defined FLAGS". Although there's always the "N FS?" or "N FC?" controls, with "N" designating the FLAG to be tested. Programs in the next two sections have general use. Consider placing them in a subdirectory called, for example, "FLG". The first program is a simple one for testing the status of any "user-defined FLAGS" you want. The second one automatically clears FLAGS in a sequence you define, or one-by-one as you designate them individually.

Section 7.5. Keeping Track

An obvious advantage of having so many "user-defined FLAGS" is that there are many circumstances, events and program conditions you can encode with them. A disadvantage is that they provide just something else to have to keep track of. Program flow could be in serious trouble if a FLAG or two were inadvertently addressed whose identity had been overlooked and whose status had been ignored. One way to avoid errors like this is to establish an initial condition for all FLAGS used in a program at its very beginning. This was the plan used in Exercises 7.2. and 7.3. For the conservative user, this strategy is probably the best. It is an attractively explicit way to start a new program with a completely clean blackboard, and no chance of initial FLAG status sabotaging program flow. An alternative is simply to reset all FLAGS at the end of the program. This procedure is analogous to using PURG to eliminate VAR menus specific to a program.

Another way is to review FLAG status from time-to-time outside of the program being developed. Exercise 7.4. presents a program that allows reviewing the "user-defined FLAGS" over a designated range. The program is constructed so that "system-defined FLAGS" are excluded from review and FLAGS in the chosen range have to be checked in an ascending order. Starting with this basic idea, it would not be difficult to redesign the program so that it reviews "system-defined FLAGS" also, displays their status in some other way, or includes additional features attractive for a specific application. Figure 7.3. shows the flow diagram for the next exercise.

Exercise 7.4. Controlled FLAG Review

Problem Statement:

Write a program that will review the "user-defined FLAGS" over a specified range.

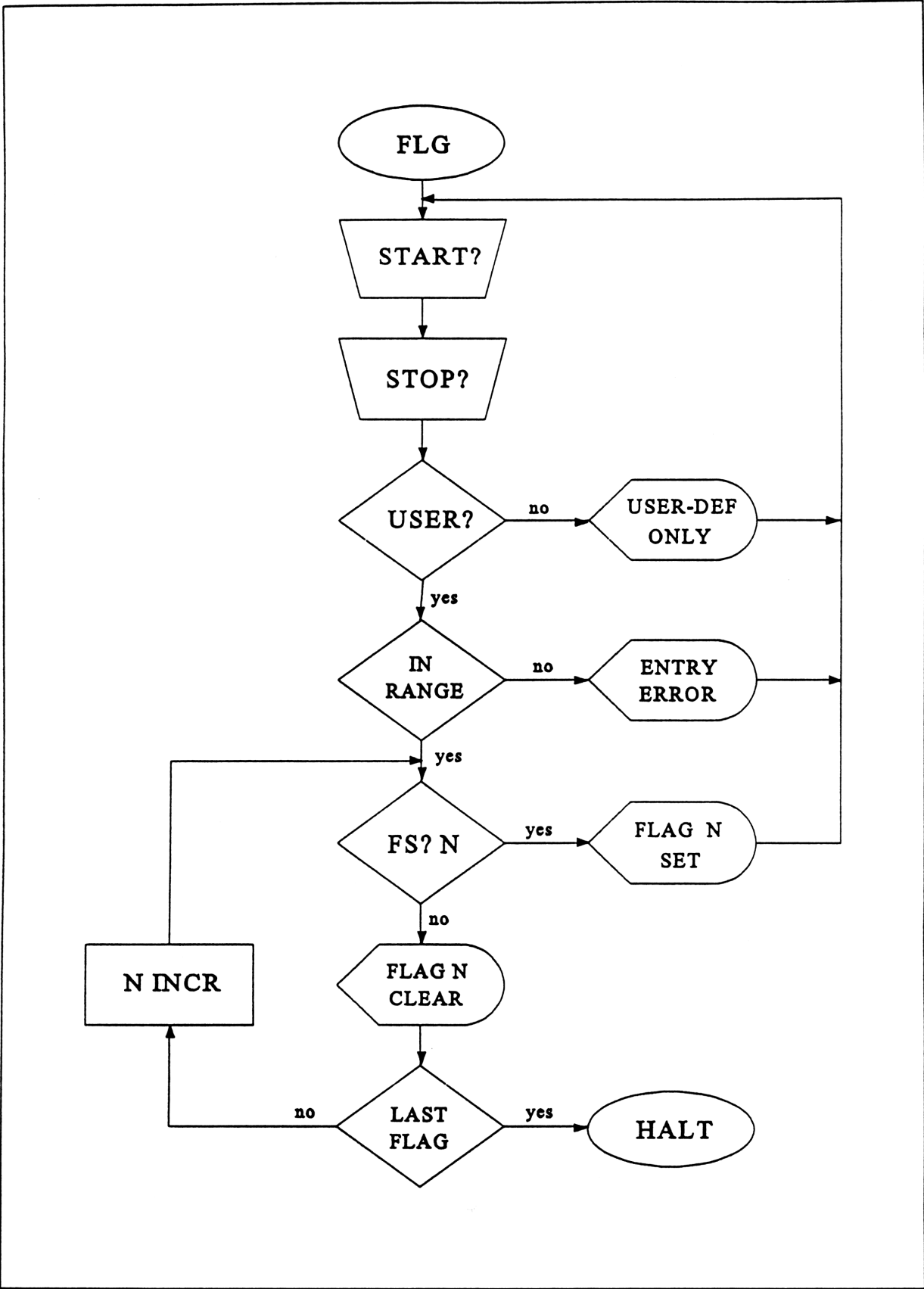


Figure 7.3. Exercise 7.4.

Solution:

A. Write the program

```
Program:FLG
<< 0 FIX TN "START?"
":First FLAG:"
INPUT OBJ → STO
TN "STOP?"
":Last FLAG:" INPUT
OBJ→ 'E' STO
IF B 0 ≤ E 0 ≤ OR
THEN TN TN TN
CLLCD
"User-Defined ONLY"
4 DISP 1 WAIT FLG
END

IF B E ≥ E 64 >OR
THEN CLLCD TN TN
TN
"Data Entry Error"
3 DISP 1 WAIT FLG
END
DO
IF B FS?
THEN CLLCD
"FLAG " B + " SET"
+
ELSE CLLCD
"FLAG " B + "CLEAR"
+
END TN 3 DISP 1
WAIT 'B' INCR
UNTIL B E >
END TN TN CLLCD
"FLAG TEST OVER" 4
DISP 1 WAIT CLEAR
HALT >>
-----
Program:TN
<< 900 .2 BEEP >>
-----
```

B. Run the program.

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | VAR, FLG, (key first FLAG to be checked; "1" is used as an example),
ENTER, (key last FLAG to be checked; "6" is used as an example), ENTER (read
"FLAG 1. SET", (pause), "FLAG 2. CLEAR", ... "FLAG TEST OVER") |

Only a few program statements are required in Exercise 7.4. to assure that only "user-defined FLAGS" will be reviewed. The first check is by "IF B 0 ≤ E 0 ≤ OR" to be sure the beginning and end FLAGS are not negative, that is, to be sure they are not "system-defined FLAGS". Another one (IF "B E ≤ E 64 > OR") determines that the beginning and ending numbered FLAGS are in the correct order, and that neither is greater than the highest numbered "user-defined FLAG". If either test is failed, the program displays an appropriate message and the user has to start over. If both tests are passed, each FLAG is automatically reviewed in order.

Section 7.6. Controlled FLAG Clearing

A few modifications to the basic structure of the program in Exercise 7.4. gives a new and useful one for clearing designated FLAGS. There are several applications for such a program. For example, you may want to be sure a certain range of FLAGS has a status of "clear" before you begin writing a new program. Or, you may want to clear just certain FLAGS before you start editing and testing an old program. The program described in

Exercise 7.5. allows you to act on these choices. The flow diagram is shown in Figure 7.4.

Exercise 7.5. Programmed Clearing

Problem Statement:

Write a program that will either clear FLAGS over a specified range, or will clear just those designated one-at-a-time.

Solution:

A. Write the program

<pre> Program:FLC << 0 FIX TN "SEQUENCE?(Y/N)" "" INPUT IF "Y" SAME THEN S END DO TN "FLAG TO CLEAR?" ":FLAG:" INPUT OBJ→ 'F' STO T F CF TN 1 'A' STO "MORE?(Y/N)" "" INPUT IF "N" SAME THEN 0 'A' STO END UNTIL 'A' == 0' END CLLCD "FLAG CLEAR OVER" 3 </pre>	<pre> DISP 2 WAIT CLLCD HALT >> ----- Program:S << TN "START?" ":FIRST FLAG:" INPUT OBJ→ 'F' STO T F 'ST' STO TN "END?" ":LAST FLAG:" INPUT OBJ→ 'F' STO T F 'LS' STO DO CLLCD "WORKING" 2 DISP ST CF 'ST' INCR UNTIL 'ST>LS' END CLLCD TN 1 3 START "FLAGS NOW CLEAR" 3 DISP .5 WAIT CLLCD </pre>	<pre> 5 WAIT NEXT CLLCD CLEAR HALT >> ----- Program:T << IF 'F ≤ 0' 'F>64' OR THEN TN CLLCD "OUT OF RANGE" 4 DISP 2 WAIT FLC END >> ----- Program:TN << 1 5 START RAND 2500 * 'P' STO RAND 3 / 'D' STO P D BEEP NEXT >> ----- </pre>
--	--	---

B. Run the Program:

(**Hint:** To see that the program does what it's supposed to, first set FLAGS 1 to 5 by pressing: PRG, TEST, NXT, NXT, 1, SF, 2, SF, 3, SF, 4, SF, 5, SF. The operation "SF" for these FLAGS not only places them in a "set" status, it also displays annunciators for each of them at the top of the view window. Watch for them to disappear as the program runs.)

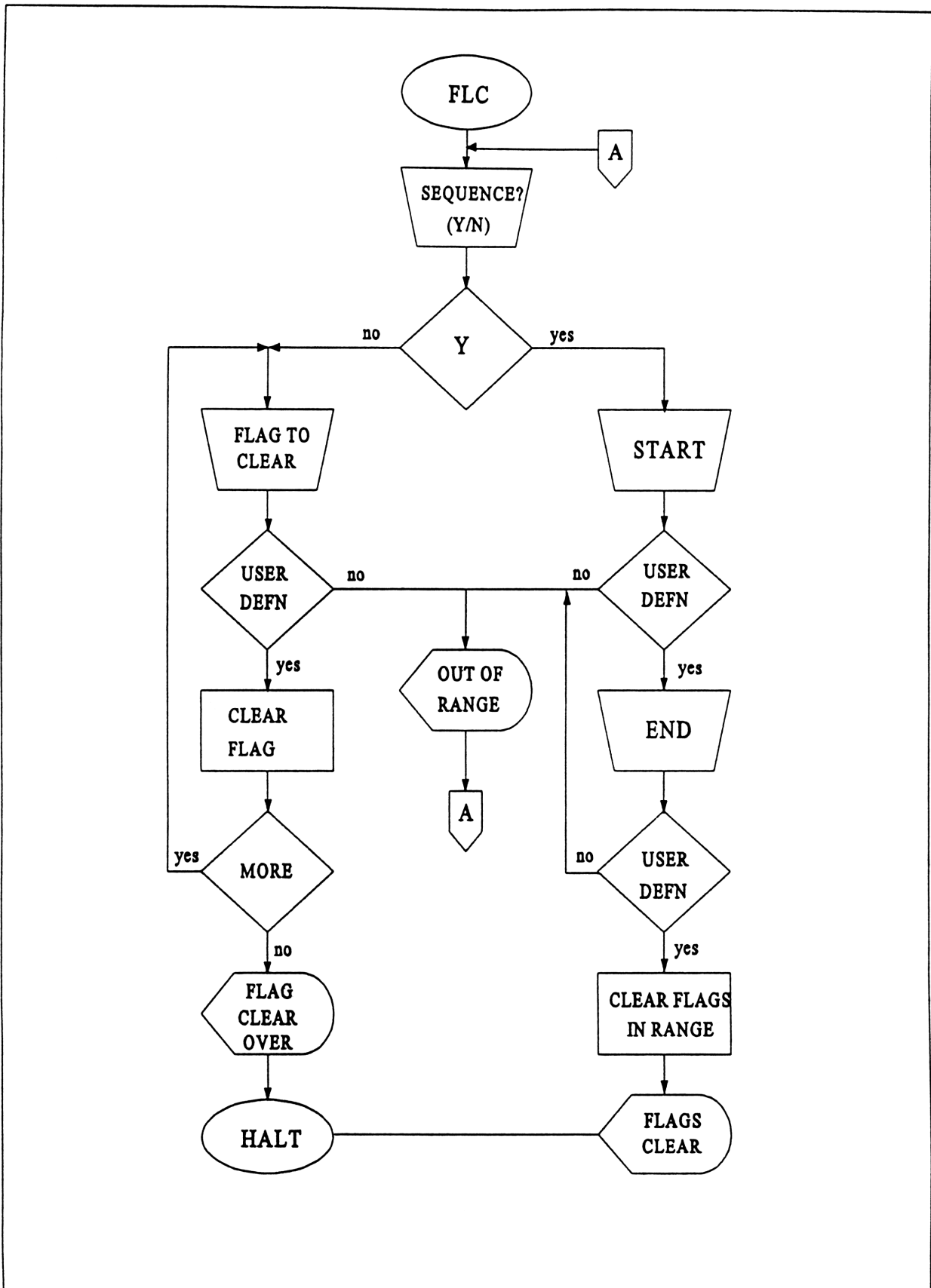


Figure 7.4. Exercise 7.5.

Problem No. 1 Clear selected FLAGS

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | VAR, NXT, FLC, α , N, ENTER, 1, ENTER (see annunciator for FLAG 1 disappear; read: "MORE?(Y/N)") |
| 2. | α , Y, ENTER, 5, ENTER (see annunciator for FLAG 5 disappear; read: "MORE?(Y/N)") |
| 3. | α , N, ENTER (read: "FLAG CLEAR OVER") |

Problem No. 2 Clear a sequence of FLAGS

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | FLC, α , Y, ENTER, 2, ENTER, 4, ENTER (read: "WORKING"; see annunciators for FLAGS 2, 3 and 4 disappear) |

(Hint: Press CANCEL, CANCEL to terminate the program at any stage.)

The first operation in running the program in Exercise 7.5. is to set the number display to show no digits to the right of the decimal. This makes sense because FLAGS are, of course, designated only by whole numbers. If the choice is to clear a sequence of FLAGS, program flow continues to the subroutine "S". Numbers for the first and last FLAGS in the sequence are requested, then tested to assure they are not "system-defined FLAGS" or "user-defined FLAGS" greater than 64. If they are in an appropriate range, they are then cleared sequentially by instructions in the "DO...UNTIL...END" structure of the subroutine, and program flow is terminated by HALT. If FLAGS are to be cleared one-at-a-time, the number for each is requested in FLC until "N" is entered in response to "MORE?(Y/N)".

The design of the subroutine "TN" provides an interesting structure for generating tones. First, a random number is generated by RAND, then constructed and stored to be the pitch of a tone which is stored as "P". Similarly, the duration of the tone is constructed by RAND and stored as "D". The instructions "P D BEEP" sound the tone which, because it has a randomly generated pitch and duration, is likely never to be heard again for a long time. Program instructions "1 5 START...NEXT" control for the tone to be heard 5 times whenever "TN" is encountered in the program.

Section 7.7. Wrapping It Up

Having read through each chapter to get this far, it's a good bet you now know more about how to use your HP48G than you did when you first brought it home from the store. **Congratulations!!** You have every reason to consider yourself an expert. Your time and

effort have been well-invested and you can look forward to your newly-found mathematical proficiencies paying off for you in many years to come.

Chapter 8

Sample Problems

Reviewing exercises and practicing different programming and calculation strategies are valuable, but sometimes just seeing an example helps too. Even though the content or subject of the example itself may not be of interest, seeing how a mathematical operation is performed and understanding how a programming procedure was accomplished can be useful and important ways to improve one's own skills. This chapter presents representative problems and typical solutions for the HP48G. Examples are selected because of their variety of application, types of calculation, styles of analyses, and different ways of constructing programs and solving problems.

These examples have special significance for people trying to improve math and programming skills, as well as for those interested in learning the complexities of the HP48G. A good case can be made for the idea that mathematics, statistics, computer programming and similar skills have their greatest value for the general user when they are applied in problem solving. The intricacies of a calculation, the significance of a mathematical procedure and the power of a programming strategy are most often remembered best when it is seen how they apply in practical applications. The general educational principle is not much different from learning to tie a set of shoe laces. Being told how to do it is one thing, but just seeing it done a couple of times makes the process easier to learn and remember.

Learning mathematical and computing skills in the context of practical application and example has another value. There is always value, of course, in being able to come up with the "correct answer" for a problem. There is much greater benefit, though, when the "correct answer" is only the first step in a series of analyses involving the questioning process of "let's see what happens when..." For example, calculating current flow in an electrical circuit requires not much more than the accurate application of existing equations. Understanding the electrical principles involved, however, can be learned much better by using "what if..?" reasoning and recalculating current flow when one or more elements in the circuit are changed. The HP48G makes such a "what if..?" approach to problem solving easy, practical and extremely valuable.

Even though some of the examples presented in this chapter may not be of specific interest, examining how each is presented and solved will give important experience in using the HP48G and in writing programs for it. Because the greatest value in presenting these examples is to show program structure clearly, their physical and biological realities are often compromised for the purpose of simplicity.

Sample Problem: Population Growth

Background

Population growth for many plants and animals typically follows an "S-shaped" pattern. Starting from a small number of individuals, the population grows slowly at first, then rises at a more rapid rate and finally slows its growth rate so that the population approaches a maximum. The increase in the beginning is because of the growing number of individuals that can produce offspring. The rate at which the population grows depends also on the rate of reproduction, among other factors. Slowed growth near the point of maximum population may be because of limited food supply, predation, and many other factors.

The "S-shaped" growth pattern for a population is as applicable as a model for a colony of bacteria, a group of fish in a pond and deer in a forest as it is for people in a country. The example presented here demonstrates how the beginning number in a population, its growth rate and its final count uniquely affect how a group of animals becomes established in a particular environment. It also demonstrates how expediently the HP48G solves a complex calculation and how conveniently data requests are constructed in a program.

Problem Statement

Four species of insects have invaded 100 acres of recently drained wet land. How successfully each is in this new environment depends on food availability, predation, temperature, humidity, wind speed, egg survival and many other factors. The population density for each species at any time is described by:

$$N_t = \frac{(N_0)(N_{\max})}{N_0 + (N_{\max} - N_0)e^{-(N_{\max})(K)(t)}}$$

where:

N_t = number of insects at time, t

N_0 = number of insects at $t = 0$

N_{\max} = maximum number of insects

K = population growth rate constant; number per unit time

t = time; weeks

Questions

Use data in Table 1 to determine:

1. When is each species growing at the fastest rate?
2. When does each species reach 85% of its final population density?
3. What is the population density for each species after 20 weeks?
4. A migratory bird, the *Blue-beaked Flicker*, feeds predominantly on insect species B. After what period of time would this bird find the drained wet land especially attractive?

Table 1: Population Growth Determinants				
	Species A	Species B	Species C	Species D
N_0 ($\times 10^3$)	0	0	10	2
N_{\max} ($\times 10^3$)	20	18	25	30
K	0.02	0.03	0.01	0.008

Hints:

1. Write the program so that $N_0 \neq 0$. Use a conditional test so that if a zero is entered in response to the N_0 data request, a small number, like 0.001, is entered.
2. For calculating data, enter numbers as shown in the table.

Solution

A. Figure 1 presents solutions graphically for data in Table 2.

B. Sample Solution:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | VAR, POP, 0 (zero), ENTER, 20 ENTER, .02 ENTER, 4 ENTER (read: "at 4 wks: 5), 8, ENTER (read: at 8 weeks: 25), 12, ENTER, (at 12 weeks: 121), etc. |

(Hint: Press CANCEL, CANCEL to discontinue program. Press POP to begin again.)

Program: POP

```
<< TN "Beginning N?"
":No.:" I 'B' STO
IF B 0 ==
THEN .001 'B' STO
END TN
"Maximum N?"
":Nmax:" I 'M' STO
TN "Growth Rate?"
":K:" I 'K' STO C >>
```

Program: C

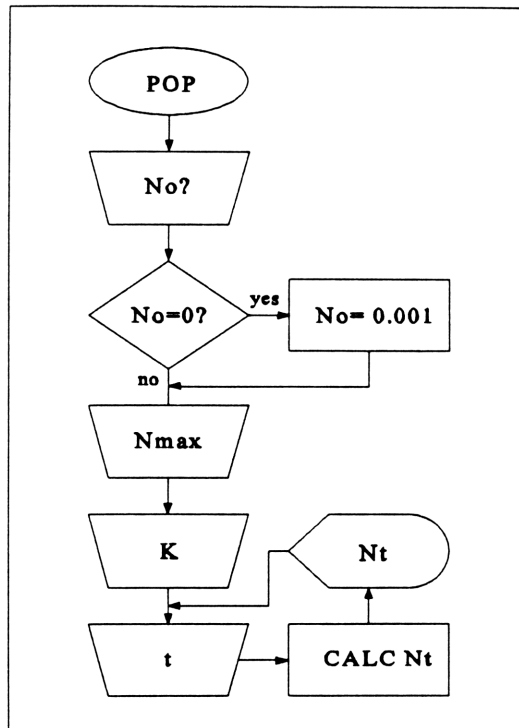
```
<< TN "Time?(WEEKS)"
":T:" I 'T' STO
'1000*B*M/(B+(M-B)*
EXP(-(M*K*T)))'
→NUM 'P' STO TN TN
0 FIX CLLCD 0 FIX
"at " T + " wks:" +
P + 3 DISP 3 WAIT C >>
```

Program: I

```
<< INPUT OBJ→ >>
```

Program: TN

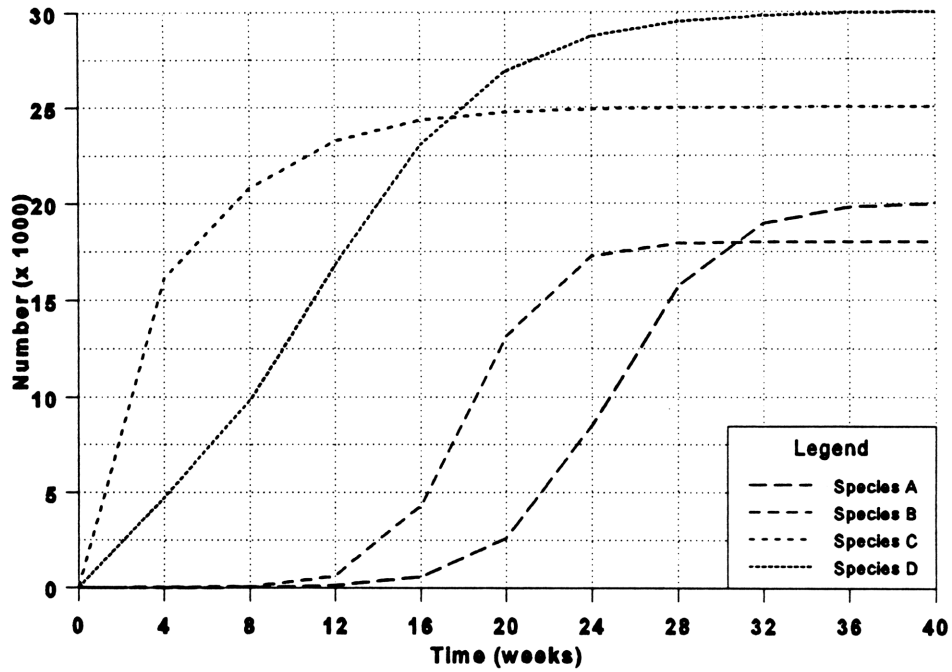
```
<< 2000 .25 BEEP >>
```



Flow Diagram

Table 2: Insect Populations (x 1,000)				
Week	Species A	Species B	Species C	Species D
4	0.01	0.01	16.11	4.72
8	0.03	0.08	20.78	9.83
12	0.12	0.63	23.26	16.80
16	0.58	4.30	24.33	23.06
20	2.60	13.17	24.75	26.90
24	8.49	17.27	24.91	28.73
28	15.71	17.91	24.97	29.50
32	18.95	17.99	24.99	29.81
36	19.78	18.00	25.00	29.93
40	19.96	18.00	25.00	29.97

Figure 1.
Insect Populations



Answers to Questions:

Table 3: Answers				
Question	Species A	Species B	Species C	Species D
1	24 weeks	18 weeks	2 weeks	11 weeks
2	28 weeks	20 weeks	8 weeks	16 weeks
3	2.6×10^3	13.2×10^3	24.8×10^3	26.9×10^3

Question 4: After about 24 weeks

Sample Problem: Relative Humidity

Background

Absolute humidity indicates the mass of water vapor in a unit volume of air. It is often expressed in units of grams per cubic meter, or as a partial pressure (torr). **Relative humidity** (RH) designates the amount of water vapor present in the air compared to how much it could hold were it saturated. For example, were RH calculated to be 50%, the air could hold twice as much water vapor. Were RH 90%, the air could hold only 10% more water vapor.

Because the water saturation point of air changes with its temperature, so does the air's relative humidity. The warmer the air, the more water vapor it can hold. For example, the same amount of water vapor in air at a low temperature might produce a RH of 85%, but only a RH of 40% were the air to warm. Also, even though the absolute humidity remains the same, relative humidity could increase, for example, from 50% to 90% were the air just to cool.

Relative humidity plays an important role in human health and comfort. In summer, relative humidity must remain less than 100% so that the water in sweat can evaporate to provide body cooling. Sweating itself doesn't cool at all, but the evaporation of the water it brings to the skin surface does. It is more than a matter of comfort. It can be lethal if the heat produced by physical activity and that gained from the environment is not dissipated. Exercising or working hard in hot and humid weather can lead to heat stroke and death if heat loss by evaporation doesn't keep pace to hold body temperature in a narrow range. They can be just as life-threatening if the weather is cool and dry, but the environment of the work setting isn't. On the other hand, the same level of activity and body heat production presents no danger at all, and often no discomfort, if relative humidity is low and there is good air flow over the skin, even though air temperature itself is 100°F or more.

Ambient relative humidity is important in the winter also. Even with the same absolute humidity, the air's relative humidity may be near 100% outside on a cold winter day, but lower than that in the Sahara desert (less than about 20%) in a house heated to 75°F. The total amount of water vapor in the air is the same, but the air can hold much more of it when it is warmed, so its relative humidity falls. The dry and itchy skin, chapped lips, and more frequent nose bleeds common in the wintertime all testify to the lower relative humidity inside heated dwellings in cold weather. It's hard sometimes to consider that room air is relatively very dry, even though it's snowing outside and the weather report indicates that relative humidity is 100%. The reference is, of course, to the outside air and to its low temperature.

Because water evaporates more rapidly the lower the relative humidity of the surrounding air, the rate at which heat is lost from a wetted surface gives a convenient basis for measuring ambient relative humidity. Were a wetted surface exposed to an environment

with 100% relative humidity, there would be no water vapor pressure gradient, no water evaporation and no evaporative heat loss. The wetted object would have the same temperature as one that was dry. The lower the relative humidity, however, the greater the water vapor pressure gradient between the wetted surface and the surrounding air, the greater the rate of evaporation, the greater the evaporative heat loss rate, and the further the temperature of the wetted surface would be below that of a dry one exposed to the same environment.

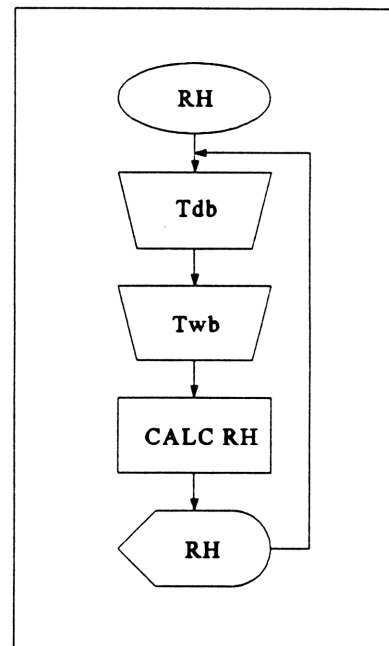
For more than a century, ambient relative humidity has been calculated based on the difference in temperature measured by two thermometers (or thermistors, thermocouples etc.), one that is dry (so-called "dry-bulb temperature") and one whose end is covered by a wetted wick (so-called "wet-bulb temperature"). The technique is called "psychrometry" and a device employing two such thermometers is called a "psychrometer". Typically, air flow is forced over the two thermometers to facilitate evaporation by swinging them in the air (using a "sling psychrometer") or by exposing them to a fan. The lower the ambient relative humidity, the greater the difference in temperatures measured, and *vice versa*. Relative humidity is calculated from these data by referring to appropriate tables or graphs. The easiest way, though, is to write a short computer program to make the calculations, as described in this exercise. Approximations made by the equations and calculations are assumed to be within measurement errors.

Problem Statement:

Develop an HP48G program to estimate ambient relative humidity based on dry-bulb temperature and wet-bulb temperature expressed on a Celsius scale. Design the program so that no calculations will be made if dry-bulb temperature is less than wet-bulb, or if dry-bulb temperature is less than -10°C or greater than 40°C.

Symbol Definitions:

- RH = relative humidity; dimensionless; %
- P_{H2O} = water vapor partial pressure; mbar
- P_{db} = water vapor partial pressure at Tdb; mbar
- P_{wb} = water vapor partial pressure at Twb; mbar
- TK_{db} = dry bulb temperature; Kelvin = $TC+273.16$
- TK_{wb} = wet bulb temperature; K; = $TC+273.16$
- TC_{db} = wet bulb temperature; Celsius
- TC_{wb} = wet bulb temperature; Celsius



Flow Diagram

Equations:

$RH = 100 \left(\frac{P_{H2O}}{P_{db}} \right)$	$P_{H2O} = P_{wb} - 0.674825 (TC_{db} - TC_{wb})$
--	---

$P_{db} = \text{antilog} \left(28.59051 - 8.2 \text{Log} TK_{db} + 0.00248 TK_{db} - \frac{314231}{TK_{db}} \right)$

$P_{wb} = \text{antilog} \left(28.59051 - 8.2 \text{Log} TK_{wb} + 0.00248 TK_{wb} - \frac{314231}{TK_{wb}} \right)$

by substitution:

$$RH = \frac{100 \left(\text{antilog} \left(28.59051 - 8.2 \text{log} TK_{wb} + 0.0248 TK_{wb} - \frac{314231}{TK_{wb}} \right) 10^3 - 0.674825 (TC_{db} - TC_{wb}) \right)}{\text{antilog} \left(28.59051 - 8.2 \text{log} TK_{db} + 0.00248 TK_{db} - \frac{314231}{TK_{db}} \right) 10^3}$$

Solution:

```

Program: RH
<< 1 FIX TN
"ENTER Tdb:Twb
(degC)"
{ ":Tdb:Twb:" { 1
0 } V } INPUT
OBJ-> DUP2 'TW'
STO 'TD' STO 'TWI'
STO 'TDI' STO
273.16 DUP 'TDI'
STO+ 'TWI' STO+
IF 'TW>TD'
THEN R
END
IF 'TD<-10' 'TD'>
40' OR
THEN R
    
```

```

END '100*(1000*
ALOG(28.59051-8.2*
LOG(TWI)+.00248*TWI
-3142.31/TWI)
-.674825*(TD-TW))/(
1000*ALOG(28.59051-
8.2*LOG(TDI)+.00248
*TDI-3142.3/TDI))'
->NUM TN TN CLLCD
"RH= " SWAP + "%" +
2 DISP 2 WAIT CLEAR
TN "ANOTHER?" ""
INPUT
IF "Y" SAME
THEN RH
END { TW TD TWI
TDI } PURGE HALT>>
    
```

```

-----
Program: R
<< TN TN CLLCD
"OUT OF RANGE" 2
DISP 2 WAIT CLLCD
RH>>
-----
Program: TN
<< 1 2 START 2000 .1
BEEP NEXT>>
-----
    
```

Run the Program

What is the approximate relative humidity when dry-bulb temperature is 25.2°C and wet-bulb temperature is 18.9°C? What is it when dry-bulb temperature is 15.1°C and wet-bulb temperature is 13.6°C?

Step

Press

1. VAR, RH, 25.2, DC, 18.9, ENTER (read: "RH=54.8%"), α , Y, ENTER, 15.1, DC, 13.6, ENTER (read: "RH=84.8%"), α , N

Sample Problem: Checks

Background

There are few jobs that provide so much aggravation as does the month-to-month chore of balancing the household checkbook. Simple mistakes in recording checks and their amounts in the register, forgotten entries and bad handwriting all add to the task. No wonder so many people don't bother to do the arithmetic every pay period and just take their chances with the bank. Nothing could make the job trouble-free and recreational, but the HP48G makes it easier.

The program described here is designed first to ask for information in the check statement and in the check register. In response to each on-screen prompt, each amount is first keyed, then entered using ENTER. A zero is keyed and entered after the last entry has been made for each category of information. This takes the program to the next set of data entry requests. When all information has been entered, the program calculates important features of the account, including the new balance, balance errors and other pertinent information.

The program first asks for the bank's balance of the account. This amount is commonly called the "current balance" or the "end balance". It is shown in the account statement that comes with the package of returned checks at the end of the month. The program next asks for each deposit for the period, then each returned check. The "returned checks" are those your check register indicates you've written and which have been returned by the bank. The program then asks for each outstanding check in this period, then each outstanding checks from past balance periods. The "outstanding checks" are those your check register indicates you've written, but which have not yet been returned by the bank. The program asks finally for you to enter any service charges.

Problem Statement:

Construct a program to help reconcile the monthly checking account.

Symbol Definitions:

B = bank statement's "current balance"
or "end balance"
C = credit
CB = calculated account balance
for this period
E = balance error
NB = new check book balance
OC = sum of outstanding checks
in this period

PO = sum of past outstanding checks
RC = sum of checks returned in this period
PB = prior balance (account balance
at beginning of period)
S = sum of service charges
SD = sum of deposits in this period
SC = sum of checks in this period
SO = sum of outstanding checks

Equations:

$$C=SD+PB$$

$$NB=C-RC-OC-S$$

$$SC=RC+OC+S$$

$$SO=OC+PO$$

$$CB=PB+SD-RC+PO-S$$

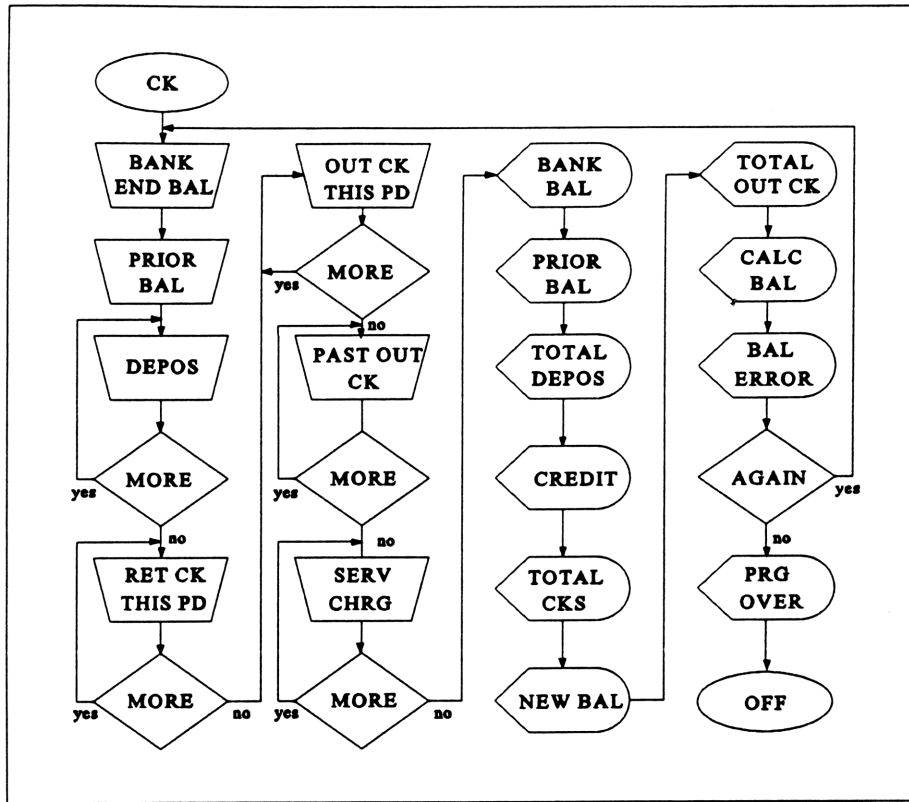
$$E=B-CB$$

Solution:

This solution uses data from the sample check register and bank statement shown on the next page.

Step**Press**

1. VAR, CK, 328.12, ENTER, 150.37, ENTER, 12.16, ENTER, 29.34, ENTER, 126.95, ENTER, 714, ENTER, 0, ENTER, 10.23, ENTER, 54.16, ENTER, 87.56, ENTER, 487.10, ENTER, 5.85, ENTER, 13.13, ENTER, 2.45, ENTER, 12.09, ENTER, 50.45, ENTER, 0, ENTER, 14.78, ENTER, 19.12, ENTER, 2.45, ENTER, 50, ENTER, 60.17, ENTER, 0, ENTER, 10.08, ENTER, 5.92, ENTER, 2.32, ENTER, 0, ENTER, 5, ENTER, 4, ENTER, 0, ENTER (read: "Bank Bal.=\$328.12"), LS CONT (read: "Pr.Bal.=\$150.37"), LS CONT (read: "Tot.Dep.=\$882.45"), LS CONT (read: "Cred=\$1032.82"), LS CONT (read: "Tot.Cks=\$869.54"), LS CONT (read: "NewBal.=\$163.28"), LS CONT (read: "Tot.OC=\$164.84"), LS CONT (read: "Cal.Bal=\$328.12"), LS CONT (read: "Bal.Err=\$0.00"), LS CONT (read: "Begin Again?(Y/N)", α , N, ENTER (read: "Program Over")



Flow Diagram

```

Prog:CK
<<TN 2 FIX "Bank's
End Balance?" ".:$.:"
I'B' STO TN "Prior
Balance?" ".:$.:" I
'PB' STO TN 0 'SD'
STO
DO TN "Deposit?"
":$.:" I DUP 'N' STO
'SD' STO+
UNTIL 'N'=0'
END TN 0 'RC'
STO
DO TN
"Returned Check?"
":$.:" I DUP 'N' STO
'RC' STO+
UNTIL 'N'=0'
END TN 0 'OC'
STO
DO TN
"Outst.Cks This
Pd?"
":$.:" I DUP 'N' STO

```

```

'OC' STO+
UNTIL 'N'=0'
END TN 0 'PO'
STO
DO TN
"Past Outstanding
Checks?"
":$.:" I DUP 'N' STO
'PO' STO+
UNTIL 'N'=0'
END TN 0 'S' STO
DO TN
"Service Charges?"
":$.:" I DUP 'N' STO
'S' STO+
UNTIL 'N'=0'
END TN TN
CLEAR
CLLCD "BkBal.=$"
B D "Pr.Bal.=$" PB
D "Tot.Dep.=$" SD
D
"Cred=$" 'SD+PB'
(NUM DUP 'C'

```

```

STO D
"Tot.Cks=$"
'RC+OC'(NUM
D "NewBal.=$"
'C-RC-OC'(NUM
D "Tot.OC=$" 'OC+
PO'(NUM D
"Cal.Bal=$" 'PB+
SD-RC+PO'(NUM
DUP 'CB' STO D
"Bal.Err=$" 'B-CB
'(NUM D TN
"Begin
Again?(Y/N)"
"" INPUT
IF "Y" SAME
THEN CK
END TN CLLCD
CLEAR
"PROGRAM
OVER" 2 DISP 1
WAIT { B PB SD
RC OC PO SC CB
C S N } PURGE

```

```

OFF>>
-----
Program: D
<<+ HALT TN TN
CLEAR>>
-----
Program: I
<<INPUT OBJ(>>
-----
Program: TN
<<2500 .15
BEEP>>
-----
Note: "S" is
included in "RC"
for "SC", "NB" and
"CB".

```

BNB

Budget National Bank

Checking Statement for Account No. 123456

Period: 6/12 to 7/13

Deposits 4

Checks 9

Begin. Bal.	+	Deposits	-	Tot. Checks	-	Serv. Chrg.	=	Balance
168.69		882.45		714.02		9.00		328.12

Jane and John Dough
 123 Fourth Street
 Anywhere, USA 98765

Low Balance	
Date:	Amount
6/15	1967.54

Average Balance
2312.83

RECORD ALL CHARGES OR CREDITS THAT AFFECT YOUR ACCOUNT							
Number	Date	Description of Transaction	Payment/Debit	✓	fee	Deposit	Balance
482	5/28	Tim's Party Store	10 08				
483	5/29	Central Grocery	17 09	X			
484	5/30	Book Store	5 92				
485	5/30	City News	2 32				
486	5/31	Electric Power Co.	79 13	X			
- - -	- - -	- - - - - (balance 6/13) - - -	- - - - -	- - -	- - -	- - - - -	150 37
487	6/13	ABC Grocery	10 23	X		12 16	
488	6/13	State Police	14 78				
489	6/14	J. Doloff (law firm)	54 16	X			
490	6/14	National Gasoline Co.	87 56	X			
491	6/15	BOJCKA Realty	478 10	X		29 34	
492	6/15	City Telephone Co.	19 12				
493	6/17	Joe's Drill	5 85	X		126 95	
494	6/18	Sam's Tire Repair	13 13	X			
495	6/25	Purity Cleaners	2 45				
- - -	6/30	(service charge)	5 00				
496	7/1	Bill's Service Station	2 45	X		714 00	
497	7/3	Travel Agency	50 00				
- - -	7/15	(service charge)	4 00				
498	9/18	Discount Center	12 09	X			
499	7/10	Kendall-Packard Co.	60 17				
500	7/14	Downtown Grocery	50 45	X			
- - -	- - -	- - - - - (balance 7/14) - - -	- - - - -	- - -	- - -	- - - - -	163 28

Sample Problem: Plant Density

Background

Plant populations, as well as those for animals, become established in a new environment because of a balance between their rate of reproduction and the rate at which individuals die. Evaluating the dynamics of the population requires considering each of these rate functions and the differences between them. Although there are several ways to approach problems like these, one is to integrate numerically the known functions for gains and losses in the population. This is a cumbersome and time consuming process with paper-and-pencil, but direct, quick and simple with the HP48G, as demonstrated in this sample problem for an exponential equation and for one with a power function.

Problem Statement

A species of annual plant was introduced to a test area of virgin forest to which it was especially well-suited. The optimum rate at which individual plants will become established is predicted by:

$$N_{opt} = 2,000 e^{0.5T}$$

where:

N_{opt} = optimum number of plants each year

T = time; years

It is unlikely, however, that the number of plants would become established at the optimum rate. Some die each year from disease, others from insects, climatic damage and related causes. How many plants actually survive is best estimated on the basis of samples taken from the test area. Data in Table 1 indicate the estimated population of plants based on sample data.

Table 1: Number Based on Sample Data	
Time (years)	Population
1	2797
2	3696
3	5351
4	8715
5	15305

Questions

1. What equation best predicts plant loss each year?
2. What would the population have been in 5 years had there been no losses?
3. How many plants were lost in the first 5 years?
4. Even with some losses, how many plants were there in 5 years?
5. How many plants were there during the period between 3 to 4 years?
6. How many plants were initially introduced to the forest?

Solutions

I. Numerical Solutions (results shown in Table 2 and Figure 1)

A. Question 1

The first step in determining the best-fit equation for how many plants were lost each year is to determine how many were lost in the 5 year period for which there are data. The number of plants lost each year is, of course, the difference between the optimum number and the number that were actually in the test area, based on sample data (Table 1). Calculating the number expected with an optimum growth rate could be done in several ways with the HP48G. One way is to take advantage of the LS DEF operation, by:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, MODES, FMT, 0 (zero), FIX, VAR, LS, EQUATION, α , α , N, O, P, T, α , LS, (), α , T, RC, LS, =, 2000, *, LS, e^x , .5, *, α , T, RC, ENTER, LS, DEF |
| 2. | 1, NOPT (read: "3,297"), 2, NOPT (read: "5,437"), 3, NOPT (read:"8,963"), 4, NOPT (read: "14,778"), 5, NOPT (read: "24,365") |

Data are now available to determine how many plants were lost each year (Table 2).

Table 2: Plant Population Summary			
time (years)	Population		
	Optimum	Sampled	Lost
1	3297	2797	500
2	5437	3696	1741
3	8963	5351	3612
4	14778	8715	6063
5	24365	15305	9060

Using data in Table 2, the HP48G's curve-fitting functions can now be used to determine the best-fit equation for predicting how many plants were lost each year:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, MODES, FMT, 1 FIX, RS, STAT, (highlight "Fit data..."), OK, EDIT, 1, ENTER, 500, ENTER, DC, 2, ENTER, 1741, ENTER, 3, ENTER, 3612, ENTER, 4, ENTER, 6063, ENTER, 5, ENTER, 9060, ENTER, ENTER, (highlight "MODEL"), CHOOS, (highlight "Best Fit"), OK, OK, (read: "500.0*X^1.8"; Correlation 1.0") |

The answer to Question 1 is the equation that best predicts the number of plants lost each year (N_{lst}) is:

$$N_{lst} = 500 T^{1.8}$$

B. Question 2

Obtaining an answer to Question 2 requires integrating the equation that predicts optimum plant population growth rate as a function of time for the period from when the plants were initially introduced until the end of the test period (5 years), that is:

$$N_{tot} = \int_0^5 (2000 e^{0.5 T}) dT$$

One way to do this is:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, SYMBOLIC, ("Integrate..." is highlighted), OK, EDIT, 2000, *, LS, e^x , .5, *, α , T, ENTER, α , T, ENTER, 0 (zero), ENTER, 5, ENTER, CHOOS, DC (to highlight "Numeric"), OK, DC, RC, (set "Fix 0"), OK, (read: "44,732") |

An alternative strategy is:

(Hint: The symbol "f" is keyed using the RS function of the 2nd key, 3rd row.)

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, MODES, FMT, 0 (zero), FIX, LS, EQUATION, RS, f, 0 (zero), RC, 5, RC, LS, (), 2000, *, LS, e^x , .5, *, α , T, RC, RC, RC, α , T, ENTER, LS, →NUM (read: "44,730") |

C. Question 3

Getting the answer to Question 3 requires integrating as a function of time from zero to 5 years the derived equation that best predicts plant loss rate, that is:

$$N_{lost} = \int_0^5 (500 T^{1.8}) dT$$

One way to do this is:

Step

Press

1. RS, SYMBOLIC, ("Integrate..." is highlighted), OK, NXT, RESET, DC, OK, NXT, EDIT, 500, *, α , T, y^x , 1.8, OK, α , T, ENTER, 0 (zero), ENTER, 5, ENTER, CHOOS, DC, OK, DC, CHOOS, (highlight "Fixed"), OK, OK (read: "16,173")

An alternative strategy is:

Step

Press

1. LS EQUATION, RS, \int , 0 (zero), RC, 5, RC, LS, (), 500, *, α , T, y^x , 1.8, RC, RC, RC, α , T, ENTER, LS, \rightarrow NUM (read: "16, 173")

D. Question 4

The answer to Question 4 is obtained by integrating as a function of time from zero to 5 years the difference between the equation that predicts the optimum number of plants and the one that predicts plant loss. To determine the number of surviving plants (N_{surv}):

$$N_{surv} = \int_0^5 (2000 e^{0.5 T} - 500 T^{1.8}) dT$$

One way to do this is to follow analogous Step/Press operations described for the solutions to Questions 2 and 3. Another way, of course, is to follow analogous operations described for the alternative solutions to these problems. For Questions 4, the solution is: 28,559.

E. Question 5

The answer to Question 5 comes from integrating as a function of time from 3 to 4 years the difference between the equation that predicts optimum plant population growth and

the one that predicts plant loss, that is:

$$N_{surv} = \int_3^4 (2000 e^{0.5 T} - 500 T^{1.8}) dT$$

One way to do this is to follow analogous Step/Press operations described for the solutions to Questions 2 and 3. Another way, of course, is to follow analogous operations described for the alternative solutions to these problems. For Questions 5, the solution is: 6,838.

F. Question 6

The answer to Question 6 is revealed in the equation that predicts optimum plant population growth. It is the number by which the exponential function is multiplied, 2,000.

Data for this sample problem are shown graphically in Figure 1.

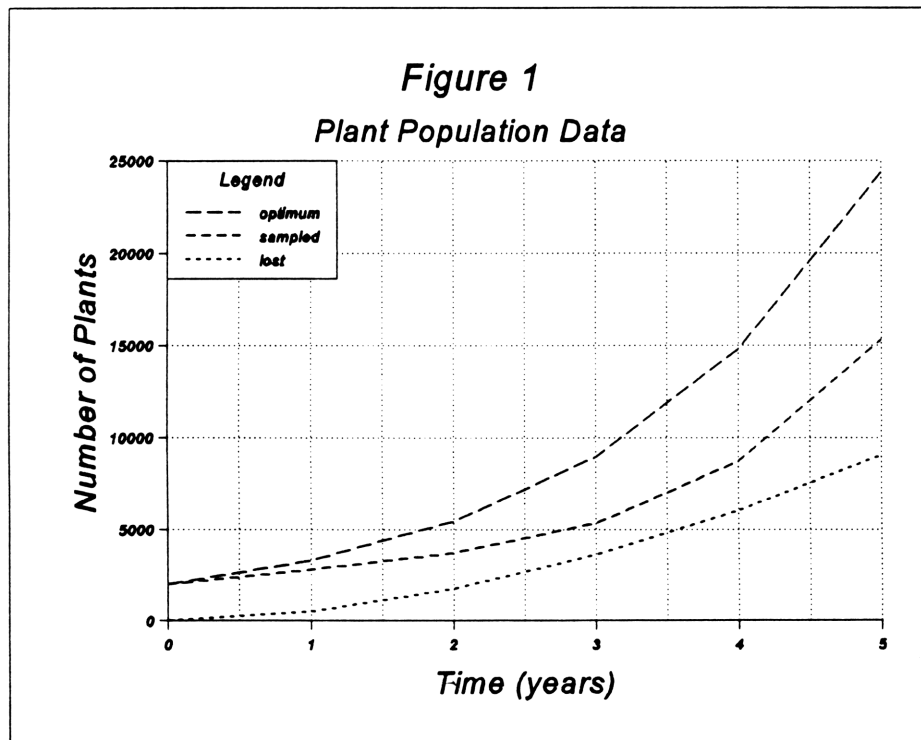


Figure 1

II. Graphical Solutions

(Reminder: The function "CANCL" is activated by the key "F" under the view window. The function "CANCEL" is activated by pressing the "ON" key. Differences between answers obtained numerically and graphically are due to imprecision in coordinate selection and rounding.)

For Question No. 2:

A. Write the Equation

Step

Press

1. LS EQUATION, 2000, *, LS, e^x , .5, *, α , T, RC, ENTER
2. ', α , T, STO

B. Construct the Graphic

Step

Press

1. LS, MODES, FMT, 2, FIX, RS, PLOT ("EQ:" is highlighted), CHOOS (highlight "T"), OK, DC (to highlight "INDEP:"), α , T, ENTER (:H-view" is highlighted), 0 (zero), ENTER, 6, ENTER, RC, 0 (zero), ENTER, 30000, ENTER (see Figure 2), ERASE, DRAW, DC (until abscissa is seen; see Figure 3)



Figure 2

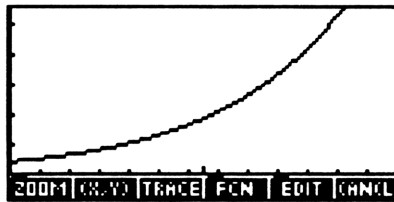


Figure 3



Figure 4

2. TRACE, (X,Y), LC (until "T:0"), NXT, FCN, AREA, NXT, PICT, TRACE. (X,Y), RC (until "T:4.98"), NXT, FCN, AREA (read: "AREA:44,356.57")
3. (to see area of integration), NXT, SHADE, DC (until abscissa is seen; see Figure 4), CANCL, ERASE, CANCEL (read: "Area: 44356.57")

For Question No. 3:

A. Write the Equation

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS EQUATION, 500, *, α , T, y^x , 1.8, RC, ENTER |
| 2. | ', α , L, STO |

B. Construct the Graphic

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, MODES, FMT, 2, FIX, RS, PLOT ("EQ:" is highlighted), CHOOS (highlight "L"), OK, DC (to highlight "INDEP:"), α , T, ENTER (:H-view" is highlighted), 0 (zero), ENTER, 6, ENTER, RC, 0 (zero), ENTER, 30000, ENTER (see Figure 5), ERASE, DRAW, DC (until abscissa is seen; see Figure 6) |

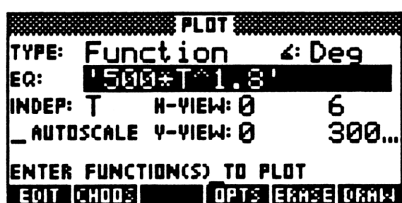


Figure 5

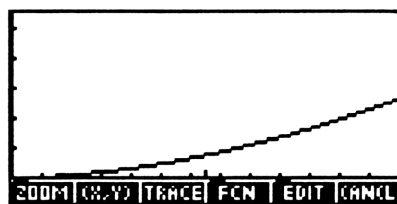


Figure 6

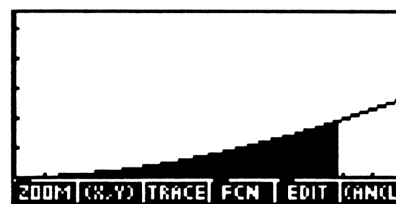


Figure 7

- TRACE, (X,Y), LC (until "T:0"), NXT, FCN, AREA, NXT, PICT, TRACE, (X,Y), RC (until "T:4.98"), NXT, FCN, AREA (read: "AREA:16,039.12")
- (to see area of integration), NXT, SHADE, DC (until abscissa is seen; see Figure 7), CANCL, ERASE, CANCEL (read: "Area: 16039.12")

For Question No. 4

A. Write the Equation

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS EQUATION, 2000, *, LS, e^x , .5, *, α , T, RC, -, 500, *, α , T, y^x , 1.8, RC, ENTER |
| 2. | ', α , α , TL, α , STO |

B. Construct the Graphic

Step

Press

1. LS, MODES, FMT, 2, FIX, RS, PLOT ("EQ:" is highlighted), CHOOS (highlight "TL"), OK, DC (to highlight "INDEP:"), α , T, ENTER (:H-view" is highlighted), 0 (zero), ENTER, 6, ENTER, RC, 0 (zero), ENTER, 30000, ENTER (see Figure 8), ERASE, DRAW, DC (until abscissa is seen; see Figure 9)



Figure 8

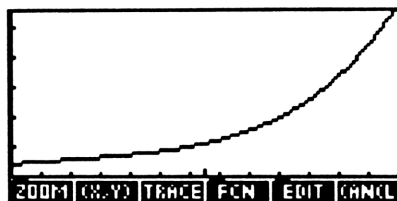


Figure 9



Figure 10

2. TRACE, (X,Y), LC (until "T:0"), NXT, FCN, AREA, NXT, PICT, TRACE, (X,Y), RC (until "T:4.98"), NXT, FCN, AREA (read: "AREA:28,317.45")
3. (to see area of integration), NXT, SHADE, DC (until abscissa is seen; see Figure 10), CANCL, ERASE, CANCEL (read: "Area: 28317.45")

For Question No. 5

Step

Press

1. LS, MODES, FMT, 2, FIX, RS, PLOT ("EQ:" is highlighted), CHOOS (highlight "TL"), OK, DC (to highlight "INDEP:"), α , T, ENTER (:H-view" is highlighted), 0 (zero), ENTER, 6, ENTER, RC, 0 (zero), ENTER, 30000, ENTER (see Figure 8), ERASE, DRAW, DC (until abscissa is seen; see Figure 9)
2. TRACE, (X,Y), LC (until "T:3.00"), NXT, FCN, AREA, NXT, PICT, TRACE, (X,Y), RC (until "T:3.97"), NXT, FCN, AREA (read: "AREA:6,572.62")
3. (to see area of integration), NXT, SHADE, DC (until abscissa is seen; see Figure 11), CANCL, ERASE, CANCEL (read: "Area: 6572.62")

To see area of integration in a different way:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, MODES, FMT, 2, FIX, RS, PLOT ("EQ:" is highlighted), CHOOS (highlight "T"), ✓CHK, (highlight "L"), ✓CHK, OK, DC (to highlight "INDEP:"), α , T, ENTER (:H-view" is highlighted), 0 (zero), ENTER, 6, ENTER, RC, 0 (zero), ENTER, 30000, ENTER (see Figure 2), ERASE, DRAW, DC (until abscissa is seen), TRACE, (X,Y), LC (until "T:3.00"), NXT, FCN, AREA, NXT, PICT, TRACE, (X,Y), RC (until "T:3.97"), NXT, FCN, AREA, NXT, SHADE (see Figure 12), CANCL, ERASE, CANCEL. |

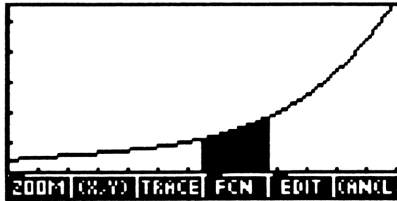


Figure 11

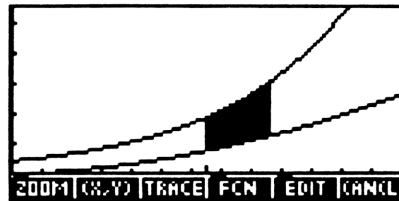


Figure 12

Sample Problem: Tumor Treatment

Background

Analyzing rates at which phenomena change is a common requirement for many problems in engineering, medicine, business and other enterprises. Although valuable, differentiating a function to evaluate its rate of change all too often requires drawing from techniques only temporarily learned in half-remembered calculus lessons. The HP48G can help. Its built-in rules handle differentiation for many functions, one of which is demonstrated in this sample problem.

Problem Statement

An otherwise healthy person was found to have a benign tumor. Its mass was about 100 grams when it was first detected. Biopsy data showed the growth to be a type known to respond well to a combination of radiation and chemotherapeutic treatments. Its change in size with treatment is predicted by:

$$M_T = \frac{(30-T)^2}{8}$$

where:

M_T = tumor mass at time, T; grams

T = time; weeks

Questions

1. Construct a table that shows how tumor mass changes as a function of time after treatment begins.
2. Derive an equation that predicts for any time during treatment the rate at which the tumor is expected to shrink.
3. When after treatment begins with the tumor be approximately one half its mass when it was first detected?
4. Treatments can be reduced one the rate of tumor shrinkage had fallen below about 3 grams per week. When can the person look forward to receiving reduced treatment?
5. When can the person be predicted to be free of the tumor assuming the response to treatment has been as effective as in the past?

Solutions

I. Numerical Solutions

(Hint: Solutions are shown in Table 1 and Figure 1)

A. Question 1

To answer Question 1, solve the equation that predicts tumor size with treatment for different periods of time. Working in a new (empty) subdirectory, one way to do this is:

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, MODES, RC, 1, ENTER, OK, RS, SOLVE, ("Solve equation..." is highlighted), OK, EDIT, α , α , M, T, LS, =, LS, (), 30, -, T, α , RC, y^x , 2, \div , 8, OK, DC, 0 (zero), ENTER, DC, SOLVE (read: "112.5"), DC, 3, ENTER, DC, SOLVE (read: "91.1"), DC, 6, ENTER, DC, SOLVE (read: "72"), (continue solutions at 3 week intervals; see Table 1), CANCEL |

An alternative solution is:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, α , α , M, T, LS, (), T, α , RC, LS, =, LS, (), 30 -, α , T, RC, y^x , 2, RC, \div , 8, ENTER, LS, DEF |
| 2. | 0 (zero), MT (read: "112.5"), 3, MT (read: "91.1"), 6, MT (read: "72.0"), (continue solutions at 3 week intervals; see Table 1) |

B. Question 2

The answer to Question 2 is obtained by differentiating the function that predicts tumor mass with treatment for different periods of time. One way to do this is:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, MODES, FMT, 2, FIX, RS, SYMBOLIC, DC (to highlight "Differentiate..."), OK, EDIT, α , α , M, T, LS, =, LS, (), 30, -, T, α , RC, y^x , 2, RC, \div , 8, OK ("VAR" is highlighted), α , T, ENTER, ("Symbolic" is highlighted), CHOOS, DC, OK, DC, 0 (zero), ENTER, OK (read: "7.50"), RS, SYMBOLIC, DC, OK, DC, α , T, ENTER, DC, DC, DC, CHOOS, DC, OK, DC, 3, ENTER, OK (read: "6.75"), (continue solutions at 3 week intervals; see Table 1) |

An alternative solution is first to expand the equation by:

$$\begin{aligned}
 M_T &= (30-T)^2/8 \\
 &= (30-T)(30-T)/8 \\
 &= (900-60T+T^2)/8 \\
 &= 112.5-7.5T+0.125T^2
 \end{aligned}$$

then differentiate the last expression by (working in a new (empty) subdirectory with no variable defined as "T" in the HOME directory):

(Hint: The symbol "δ" is obtained by the RS function of the 1st key, 2nd row. To see the full expression after each of the following steps, press LS DEF, then ENTER before executing the next step.)

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | LS, MODES, FMT, 3, FIX, LS, EQUATION, RS, δ, α, T, RC, 112.5, -, 7.5, *, α, T, +, .125, *, α, T, y ^x , 2, RC, ENTER (read: " 'δT(112.500-7.500*T+.125*T^2)' ") |
| 2. | EVAL (read: " 'δT(112.500-7.500*T)+δT(0.125*T^2)' ") |
| 3. | EVAL (read: " ' -δT(7.500*T)+0.125*δT(T^2)' ") |
| 4. | EVAL (read: " ' -(7.500*δT(T))+0.125*(δT(T)*2*T^(2-1))' ") |
| 5. | EVAL (read: " ' -7.500+0.125*(2*T)' ") |

The last expression is equivalent to: "0.25T-7.5". This reveals that the rate of change of the tumor (S; grams per week) with treatment is predicted by:

$$S = 0.250T - 7.5$$

and is solved by:

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, α, S, LS, (), α, T, RC, LS, =, .25, *, α, T, -, 7.5, ENTER, LS, DEF |
| 2. | 0 (zero), S (read: "-7.500"), 3, S (read: "-6.750"), (continue calculating "S" at 3 week intervals; see Table 1) |

(Hint: The minus sign indicates that "S" is decreasing.)

C. Question 3 Calculated data show the tumor will be half its original size just before about 9 weeks of treatment.

D. Question 4: Calculated data show that therapy can be reduced after about 18 weeks.

E. Question 5: Calculated data predict that the tumor will be gone after about 30 weeks of treatment.

Table 1: Tumor Statistics		
Time (weeks)	Tumor Size (grams)	Growth Rate (g/wk)
0	112.5	-7.50
3	91.1	-6.75
6	72.0	-6.00
9	55.1	-5.25
12	40.5	-4.50
15	28.1	-3.75
18	18.0	-3.00
21	10.1	-2.25
24	4.5	-1.50
27	1.1	-0.75
30	0.0	0.0

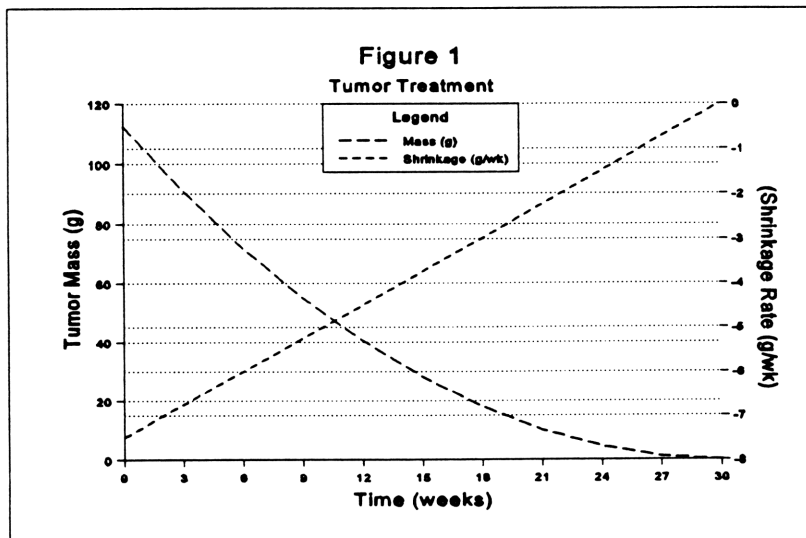


Figure 1

II. Graphical Solutions

A. Construct Equation

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | LS, EQUATION, LS, (), 30, -, α , T, RC, y^x , 2, RC, \div , 8, ENTER |
| 2. | , α , α , M, T, α , STO |

B. Plot Data for Tumor Mass

- | <u>Step</u> | <u>Press</u> |
|-------------|---|
| 1. | RS, PLOT ("EQ:" is highlighted), CHOOS, (highlight "MT:'(30-T)^2/8'"), OK, DC (to highlight: "INDEP:"), α , T, ENTER ("H-view" is highlighted), 0 (zero), ENTER, 30, ENTER, RC (to highlight: "V-view"), 0 (zero), ENTER, 120, ENTER (see Figure 2), ERASE, DRAW |
| 2. | TRACE, (X,Y) (see Figure 3) |

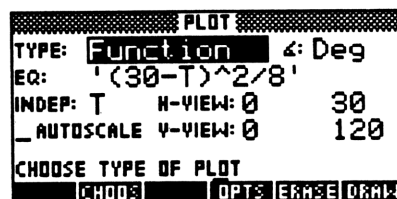


Figure 2

(Hint: Use LC or RC to move cursor to a position on the curve to show corresponding X (time; weeks) and Y (tumor size; grams) coordinates. For example, when "T:9.00", read: "Y:55.13", and when "T:15.00", read: "Y:28.13". Check graphically derived data with those in Table 1.)



Figure 3

- CANCEL, CANCEL

C. Plot Data for Tumor Growth Rate

(Hint: The function "F" shows the first derivative of the selected equation.)

- | <u>Step</u> | <u>Press</u> |
|-------------|--|
| 1. | RS, PLOT (set "EQ:'(30-T)^2/8'"), DRAW, FCN, NXT, F', (use the key "F" under the view window for:) CANCL (set "V-view: 0 (zero) -8"; (use "8, +/-, ENTER to construct "-8"), (do not press ERASE), DRAW (see derivative function), TRACE, (X,Y) (see Figure 4) |

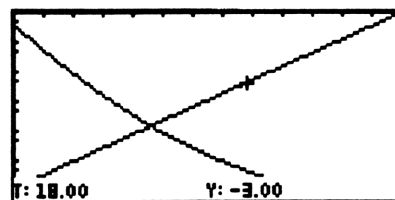


Figure 4

(Hint: Use UC, if necessary, to position the cursor on the straight line representing the derivative function. To determine when the tumor growth rate has fallen to below about 3 grams per week (Question 4), use LC and/or RC to position cursor on "Y: - 3.00" (read: "T: 18.00"))

2. CANCEL, CANCEL (to return to view window)

Sample Problem: Useful Statistics

Background

Calculating basic statistics for a set of data is a commonplace enough operation, but it is sometimes cumbersome and time consuming. The HP48G has a number of facilities to make such calculations quickly and display them in a customized fashion. It is always possible just to use the RS STAT operations, but it is more convenient to write a program to calculate the most often used statistics and conveniently display their solutions.

The sample program described here first asks how many digits to the right of the decimal the user requires, then asks for data entries one at a time in an indefinite series. After the last entry has been made, entering a zero initializes data calculation and display. There are a couple of ways to avoid problems with the program if one or more data to be evaluated are, in fact, equal to zero. One solution is to enter a number that is small enough not to introduce an error into the statistical calculation. For example, if a data series is, "...25.4, 56.2, 10.5, 0.0...", entering "0.0001" instead of zero allows continuing with data entry. Another solution is to rewrite the program so that some other entry signals the end of the data entry phase.

Problem Statement

Write a program that automatically calculates and displays the total, average, standard deviation, standard error and the number of entries in a series of data of indefinite length. Configure the program so that an entry of zero ends the data entry phase and begins calculations and display.

Solution

Program:STATS << CLΣ TN "decimal?" ":digits to right:" INPUT OBJ→ FIX CLEAR REQ >> ----- Program REQ << TN1 "Entry?" ":N:" INPUT OBJ→ N' STO IF 'N≠0' THEN N Σ+ REQ ELSE TN TN CLLCD	TOT "Total=" SWAP + 2 DISP MEAN "Mean=" SWAP + 3 DISP SDEV DUP 'SD' STO "SDev=" SWAP + 4 DISP NΣ 'T' STO ' SD/√(T-1)' →NUM "SErr=" SWAP + 5 DISP 0 FIX NΣ "N=" SWAP + 6 DISP 0 WAIT CLEAR { SD T N ΣDAT } PURGE 2 FIX END >>	----- Program: TN1 << 1500 .1 BEEP >> ----- Program TN << 1000 .1 BEEP>>
--	--	---

Run the Program

Calculate the total, mean, standard deviation, standard error and N for the following data: 24.52, 12.85, 45.87, 19.64, 22.71, 17.00, and 12.19. Show answers with two digits to the right of the decimal.

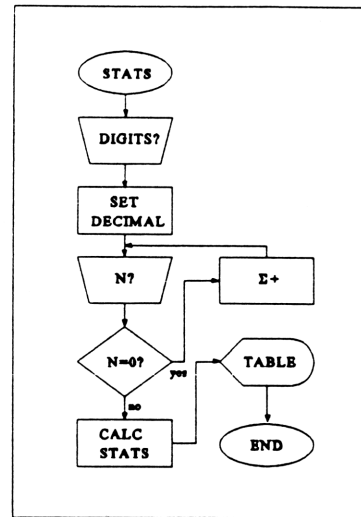
Step

Press

1. VAR, STATS, 2, ENTER, 24.52, ENTER, 12.85, ENTER, 45.87, ENTER, 19.64, ENTER, 22.71, ENTER, 17, ENTER, 12.19, ENTER, 0 (zero), ENTER

read:

```
Total=154.78
Mean= 22.11
SDev= 11.45
SErr= 4.68
N=7.
Σ0NT | N | SUM | REQ | TN1 | TN
```



Flow Diagram

Sample Problem: Automatic Curve Fitting

Background

Determining which one of a series of equations best-fits a set of data can be an important first step in analyzing functional relationships between independent and dependent variables, and for providing important insight into phenomena. Prior to technology like that available in the HP48G, curve-fitting was a laborious and time consuming process that often cost many hours in pursuing false leads. Even with the HP48G's impressive statistical library and built-in programs for curve-fitting data, it still takes time and a good memory to remember which buttons to press first in the complicated sequence required for data entry, model choice and final calculation. Unless these operations are performed frequently, it's easy to forget what comes next in the calculation process.

The program described in this section is designed to make curve-fitting with the HP48G rapid, accurate and easy for trying different data models. It also demonstrates some useful programming strategies, like automatic entry of data into the statistical matrix table, and constructing "keyboard-sensitive" operations so that just pressing a selected key triggers the next series of operations. The program also shows how data entry is automatically tallied and how program flow is controlled by information generated within the program itself. Most importantly, though, the program is practical, easy-to-use and flexible for the user who needs to curve-fit different sets of data. Instructions for data entry are explicit so that even the infrequent user can operate the program with ease.

Problem Statement

Construct a program for the HP48G that provides automatic curve-fitting of entered data. Design the program so that it:

1. requests the number of digits to the right of the decimal for answer displays,
2. requests the number of X,Y pairs in the data set,
3. requests data with identifying pairs of numbers, for example "...X1... Y1... X2... Y2..." etc.
4. automatically constructs arrays in the statistical registers,
5. calculates and displays the intercept, slope and correlation coefficient for a selected model once all data pairs have been entered,
6. allows additional model testing with the same data, and
7. allows entry of new data for additional tests.

```

Program: CURV
<< 1 'T' STO CLΣ
CLLCD TN
"No. of digits for
answer display?"
":N:" INPUT OBJ→
'D' STO TN
"No. of X,Y pairs??"
":N:" INPUT OBJ→
'N' STO REQ >>

```

```

-----
Program: REQ
<< IF 'N≠T-1'
  THEN 0 FIX TN
  "Enter X" T + ""
  INPUT OBJ→ TN
  "Enter Y" T + ""
  INPUT OBJ→ 2 →ARRAY
  Σ+ 1 'T' STO+ REQ
  ELSE CALC
  END >>

```

```

Program: CALC
<< D FIX CLLCD TN
"Key Number for model:
1. Linear
2. Logarithmic
3. Exponential
4. Power"
DUP 1 DISP 7 FREEZE
0 WAIT 'C' STO
CLLCD
CASE C 82.1 SAME
  THEN LINFIT
  END C 83.1 SAME
  THEN LOGFIT
  END C 84.1 SAME
  THEN EXPFIT
  END C 72.1 SAME
  THEN PWRFIT
  END
END TN TN LR 3
DISP 2 DISP CORR"r="
SWAP + 4 DISP
" (any key to con't)"

```

```

6 DISP 0 WAIT TN
"ANOTHER MODEL FOR
SAME DATA?(Y/N)"
"" INPUT OBJ→ 'A'
STO
IF A 'Y' SAME
  THEN CALC
  ELSE TN
"NEW DATA(Y/N)?" ""
INPUT OBJ→ 'A' STO
IF A 'Y' SAME
  THEN CURV
  END { ΣPARΣDAT
D N C T A } PURGE
TN CLLCD
" END OF CURVE FIT"
4 DISP 1 WAIT CLEAR
TN TN
END >>

```

```

-----
Program: TN
<< 1500 .1 BEEP >>

```

Run the Program

Determine the form of the equation that best-fits data in Table 1.

Table 1: Test Data		
Observation	X	Y
1	5.93	4.06
2	1.25	12.26
3	3.01	6.57
4	8.45	3.16
5	0.98	14.57

Solution

Step

Press

1. VAR, CURV, 4, ENTER, 5, ENTER, 5.93, ENTER, 4.06, ENTER, 1.25, ENTER, 12.26, ENTER, 3.01, ENTER, 6.57, ENTER, 8.45, ENTER, 3.16, ENTER, .98, ENTER, 14.57, ENTER (read: (see Figure 1)), 1 (read: (see Figure 2)), (any key), α , Y, ENTER, 2 (read: (see Figure 3)), (any key), α , Y, ENTER, 3 (read: (see Figure 4)), (any key), α , Y, ENTER, 4 (read: (see Figure 5)), α , N, ENTER, α , N, ENTER

```
Key Number for model:
1. Linear
2. Logarithmic
3. Exponential
4. Power
┌──┬──┬──┬──┬──┬──┬──┐
E D I T | N | O | T | CURV | REQ
```

Figure 1

```
: Intercept: 13.7711
: Slope: -1.4391
r=-0.9132
<any key to con't>
┌──┬──┬──┬──┬──┬──┬──┐
CURV | REQ | CALC | TN |
```

Figure 2

```
: Intercept: 13.6522
: Slope: -5.2961
r=-0.9855
<any key to con't>
┌──┬──┬──┬──┬──┬──┬──┐
CURV | REQ | CALC | TN |
```

Figure 3

```
: Intercept: 15.0945
: Slope: -0.2014
r=-0.9680
<any key to con't>
┌──┬──┬──┬──┬──┬──┬──┐
T | CURV | REQ | CALC | TN |
```

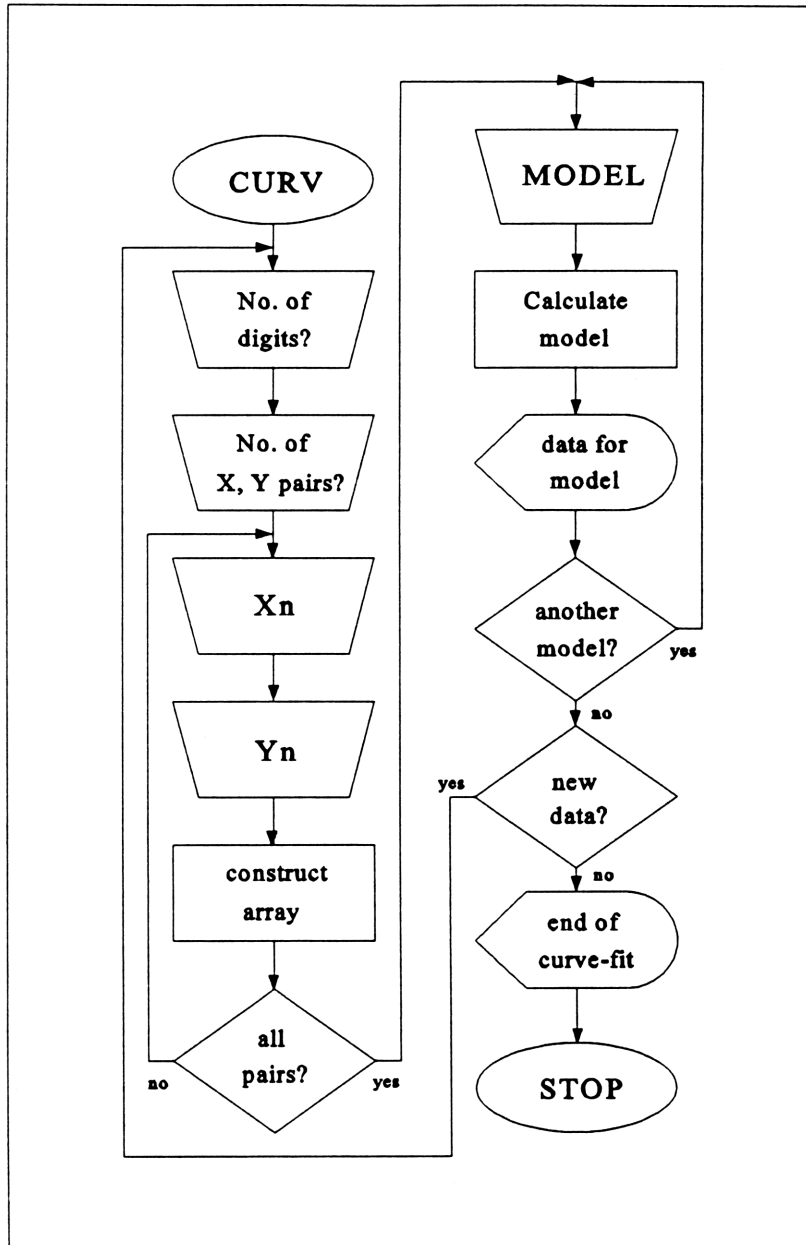
Figure 4

The best-fit model for the test data is the power function:

$$Y = 14.36X^{-0.71}$$

```
: Intercept: 14.3620
: Slope: -0.7096
r=-1.0000
<any key to con't>
┌──┬──┬──┬──┬──┬──┬──┐
T | CURV | REQ | CALC | TN |
```

Figure 5



Flow Diagram

Chapter 9

File Transfer and Printing

(Hint: The designation "HP48" in this section refers to the HP48S/SX and HP48G/GX machines. Transferring files out of and into an HP48 requires a "serial interface cable" and the software that comes with it in the model HP82208A or HP82208C Serial Interface Kit. Other programs in this software allow for graphic conversion, graphics printing, and provide a calendar function, stopwatch operations and other utilities. Complete instructions are in the "README.TXT" file of the program disks.)

The HP48 has comparatively large RAM storage. This allows keeping many programs in the machine, either in the HOME directory, or in any one of a number of subdirectories. There are times, however, when one or more programs are more conveniently stored outside the HP48 in the hard drive of a tabletop computer, on a 5¼" or 3½" disk, on tape, or on some other storage device. This requires initially transferring files from the HP48 through an IBM-compatible tabletop computer to whatever storage device is used. When the files need to be used again in the HP48, they must be transferred back through an IBM-compatible computer.

Besides transferring files to a storage device outside the HP48, it may also be necessary to transfer files between HP48 machines. This enables others to use programs you've developed, and gives you access to theirs. Fortunately, procedures for file transfer between storage media and between other HP48's are relatively straightforward, but a number of steps need to be taken in a precise order, and a number of conditions must be satisfied for both the HP48 and for the tabletop computer. This chapter describes the basics for file transfer and gives examples for them.

In addition to transferring files for storage or for use in other HP48's, it is often useful to get a print of the HP48 view window. Displays of graphs, interim calculations, program instructions and other view window configurations are easily transferred for printing using an IBM-compatible computer and the serial interface cable. The procedures described in this chapter for making prints of HP48 displays refer to tools available in the "HP Display Grabber" program that comes with the software for the model HP82208C serial interface cable.

The first step in the process of obtaining a print is to import a copy of the HP48 view window so it is seen on the monitor display of the IBM-compatible computer. The second step is to save the display either as a ".BMP" or ".TIFF" file. The last step is to use this file with appropriate word processing or other appropriate programs to make prints of it.

Instructions in this chapter describe how to obtain these prints using either the HP48G/GX, or HP48S/SX machines. Instructions are written with the assumption that the:

1. HP-48 series PC serial link for IBM-compatible PCs software (82208C) software has been installed,
2. the IBM-compatible computer is running under WINDOWS. ver. 3.1. with the "HP48 Display Grabber" installed.
2. serial interface cable is connected to an IBM-compatible computer and to an HP48G/GX or HP48S/SX.

(Hint: Instructions for the installations at Steps 1 and 2 are in the software that comes with the serial interface cable.)

Section 9.1. File Transfer Operations

Section 9.1.1. Installing the Software

Transferring HP48 files requires that the serial interface software be installed on the hard disk of the tabletop computer. After its installation, you provide the instructions so that these files can interact with those that are built-in to the HP48. To install the program that comes with the "serial interface kit" in the tabletop computer:

1. Insert the software disk (or a backup copy of it) into a drive with matching size in the tabletop computer.

(Hint: Complete the following steps from the DOS display of the computer's "C:" drive, through the "File Manager" display in WINDOWS (ver. 3.1) or through the "MS-DOS PROMPT" facility available in WINDOWS.)

2. If the DOS display is used, at the "C:\>" prompt, log to the drive that has the program disk inserted into it and type "SETUP C:". For example, if the "serial interface" software disk is in Drive A, read: "A:SETUP C:". Press ENTER and follow screen instructions. Files will be copied from the software disk to a subdirectory on the hard disk called, "SERIAL" if the HP82208A program is used, or "LINK48" if the HP82208C program is used.

Section 9.1.2. File Transfer from the HP48G to a Tabletop Computer

A. For the IBM-compatible computer:

- A.1. Turn off the tabletop computer and the HP48G. Connect the serial interface cable to the HP48G and to the serial connection on the tabletop computer.
- A.2. Turn on the tabletop computer. If the HP82208A program is used, go to the "C:\>" display on the monitor of the tabletop computer and type, "CD SERIAL" (read, "C:\>CD SERIAL") and press ENTER (read, "C:\SERIAL>"). If the HP82208C program is used, go to the "C:\>" display on the monitor of the tabletop computer and type, "CD LINK48" (read, "C:\>CD LINK48") and press ENTER (read, "C:\LINK48>")
- A.3. Type "KERMIT" (read, "C:\SERIAL>KERMIT" or read, "C:\LINK48>KERMIT"), press ENTER (read, "Kermit-MS>").
- A.4. Type "SET PORT 1" (read, "Kermit-MS>SET PORT 1"), press ENTER (read, "Kermit-MS>").
- A.5. Type "SET BAUD 9600" (read, "Kermit-MS>SET BAUD 9600"), press ENTER (read, "Kermit-MS>").
- A.6. Type "RECEIVE" (read, "Kermit-MS>RECEIVE), press ENTER and read:

File Name:
KBytes transferred:
Receiving: In progress
Number of packets: 0
Packet length:
Number of retries: 0
Last error:
Last message:

(Hint: The count for the "Number of retries" will automatically increase until successful file transfer begins. This phase may self-terminate to display "Kermit-MS" at the bottom of the monitor. If so, after completing Steps B.1. to B.7., repeat Step A.6., before continuing with Step B.8.)

B. For the HP48G:

B.1. Turn on the HP48G.

B.2. Press: RS, I/O.

B.3. Highlight "Transfer.." (see display below at left), press "OK" (see display below at right).



B.4. If necessary, set screen values to: PORT: Wire; TYPE: Kermit; FMT: ASC; XLAT: New1; CHK: 3; BAUD: 9600; Parity: None.

B.5. Select name(s) of files to be transferred by: highlight "NAME:", press CHOOS

B.6. Highlight "Local vars", press "OK".

B.7. Highlight directory/file name(s), press " ✓CHK" for each, press "OK".

B.8. If necessary, repeat Step A.6., then press "SEND". Wait for file transfer to be completed. Successful transfer places file(s) in either "C:\SERIAL" or in "C:\LINK48".

(Hints: 1. Continue at Step A.6. to transfer additional files. 2. To exit Kermit, at Step A.6. type "EXIT" (read, "Kermit-MS>EXIT"), then press ENTER (read, "C:\SERIAL>" or "C:\LINK48").

Keep a careful record of all files that were transferred from the HP48G to "C:\SERIAL" or "C:\LINK48" in the tabletop computer. They can now be moved or copied into most word processing programs to be configured for inclusion in a document, printed, or manipulated in other ways. Be careful, of course, not to remove any files from "C:\SERIAL" OR "C:\LINK48" other than the ones transferred from the HP48G. This would adversely affect the integrity of the SERIAL or LINK48 subdirectories. If this happens, the remedy is either to copy the missing files back again, or reload the "serial interface" software, as described in Section 9.1.1.

Not all characters used in HP48 programs will be transferred accurately. For example, symbols like "<< >>", "<=", ">=", among others will not transfer in these forms. The solution is easy. Use your word processing program to erase the incorrect symbols and figures, and insert the ones you need. Careful proof reading is essential, of course.

Section 9.1.3. File Transfer from a Tabletop Computer to the HP48G

One advantage of storing programs and data files produced by the HP48G on floppy disks is it provides a library of virtually unlimited size. Another advantage is that these files can now be more easily used by others, as well as inserted into documents and used in other similar ways. Not the least advantage is that RAM storage in the HP48G itself is available for other programs and use. Using the disk-stored programs once again in the HP48G, however, requires they be copied back into the machine through a tabletop computer. The procedure is just as straightforward as copying programs to the disk.

(Hint: Use all capital letters for entering file names)

A. For the IBM-compatible Computer

A.1. Complete steps A.1. to A.5. in Section 9.1.2.

A.2. At Step A.6. in Section 9.1.2., instead of "RECEIVE", type "SEND" (read, "Kermit-MS>SEND"), press ENTER (read, "Local Source File:")

A.3. Using all capital letters, type the name of the file in "C:\SERIAL" or "C:\LINK48" to be transferred, press ENTER (read, "Remote Destination File:").

A.4. Using all capital letters, type the name of the file to be transferred, press ENTER, read:

File Name:
KBytes transferred:
Sending: In progress
Number of packets: 0
Packet length: (read size of file)
Number of retries: 0
Last error:
Last message:

(Hint: The count for the "Number of retries" will automatically increase until successful file transfer begins. This phase may self-terminate to display "Kermit-MS" at the bottom of the monitor. If so, complete Steps B.1. to B.4. in this section, then begin again at Step A.1. in this section.)

B. For the HP48G

B.1. Turn on the HP48G, then go to the subdirectory into which transferred files will be stored.

B.2. Press: RS, I/O

B.3. Highlight "Transfer...", press "OK".

B.4. If necessary, set screen values to: PORT: Wire; TYPE: Kermit; FMT: ASC; XLAT: New1; CHK: 3; BAUD: 9600; Parity: None.

B.5. Press "RECV", wait for file transfer process to complete.

B.6. Press "CANCEL" to exit "I/O" (read name of transferred file in VAR menu).

(Hint: To transfer another file, begin at Step A.1. in this section.)

Section 9.1.4. File Transfer Between HP48Gs

The procedures for transferring files and programs from one HP48G to another is similar to those for sending information back and forth between an HP48G and a tabletop computer, but it is much simpler. The process still requires configuring and arranging both machines so they can communicate, but there are far fewer keystrokes.

A. For the HP48 Source

A.1. Turn on the HP48G and go to the subdirectory that contains the files to be transferred.

A.2. Press RS, I/O, highlight "Send to HP48G...", press "OK", then "CHOOS" to select files.

A.3. Highlight each file to be transferred and indicate its selection by pressing "✓CHK". Press "OK".

B. For the HP48G Recipient

- B.1. Create and enter a subdirectory into which imported files will be stored.
- B.2. Press RS, I/O and highlight "Get from HP48". Do not press "OK" at this time.

C. To Complete the File Transfer

- C.1. Place the source and recipient HP48Gs head-to-head about an inch or so apart, aligning them at their small arrows molded into the thin upper edge of each case.
- C.2. Press "OK" on the HP48G recipient, then press "SEND" on the HP48G source. Wait for the file transfer process to be completed. Seeing the file names appear in the originally empty subdirectory of the HP48G recipient indicates successful file transfer.

Section 9.1.5. File Transfer Between HP48S/SX and HP48G/GX

I. From an HP48S/SX to an HP48G

A. For the HP48S/SX

- A.1. Turn the HP48S/SX on and go to the subdirectory that contains the files to be transferred.
- A.2. Press LS, { }. Using the white keys under the view window, select the files to be transferred, then press ENTER.
- A.3. Press LS, I/O, SETUP. Using the white keys, configure the view window to indicate:

I/O setup menu	
IR/wire:	IR
ASCII/binary:	binary
baud:	9600
parity:	none 0
checksum type:	3
translate code:	1

- A.4. Press, LS, I/O

B. For the HP48G/GX

- B.1. Create and enter a subdirectory into which files will be stored.
- B.2. Press RS, I/O, highlight "Get from HP48". Do not press "OK" at this time.

C. To Complete File Transfer

- C.1. Place the source and recipient HP48Gs head-to-head about an inch or so apart, aligning them at their small arrows molded into the thin upper edge of each case.
- C.2. Press "SEND" on the HP48S/SX, then press "OK" on the HP48G/GX. Wait for the file transfer process to be completed.

II. File Transfer from an HP48G/GX to an HP48S/SX

A. For the HP48G/GX

- A.1. Turn on the HP48G/GX and go to the subdirectory that contains the files to be transferred.
- A.2. Press RS, I/O, highlight "Send to HP48G...", press "OK", then "CHOOS" to select files.
- A.3. Highlight each file to be transferred and indicate its selection by pressing "✓CHK". Press "OK".

B. For the HP48S/SX

- B.1. Create and enter a subdirectory into which files will be stored.
- B.2. Press LS, I/O, SETUP. Using the white keys, configure the view window to appear as shown at the right:

I/O setup menu	
IR/wire:	IR
ASCII/binary:	binary
baud:	9600
parity:	none 0
checksum type:	3
translate code:	1

C. To Complete the File Transfer

- C.1. Place the source and recipient HP48s head-to-head about an inch or so apart, aligning them at their small arrows molded into the thin upper edge of each case.
- C.2. Press "SEND" on the HP48G/GX, then press "RECV" on the HP48S/SX. Wait for file transfer process to be completed. Press VAR on the HP48S to see the new files.

Section 9.1.6. File Transfer Between HP48S/SXs

A. For the HP48S/SX Source:

- A.1. Turn on the HP48S/SX and go to the subdirectory that contains the files to be transferred.
- A.2. Press LS, I/O, Setup and configure the view window to appear as shown on the right, then press: LS, I/O.

B. For the HP48S/SX Recipient:

- B.1. Create and enter a subdirectory into which files will be stored.
- B.2. Press LS, I/O, SETUP. Using the white keys, configure the view window to appear as shown on right, then press: LS, I/O

I/O setup menu	
IR/wire:	IR
ASCII/binary:	binary
baud:	9600
parity:	none 0
checksum type:	3
translate code:	1

C. To Complete the File Transfer

- C.1. Place the source and recipient HP48 S/SXs head-to-head about an inch or so apart, aligning them at their small arrows molded into the thin upper edge of each case.
- C.2. Press "SEND" on the HP48S/SX source then press "RECV" on the HP48S/SX recipient. Wait for file transfer process to be completed. Press VAR on the HP48S/SX recipient to see the new files.

Section 9.2. Printing HP48 View Window Displays

Section 9.2.1. Obtaining Prints of the HP48G/GX View Window

A. For the IBM-compatible Computer:

- A.1. Turn the computer on, and at the WINDOWS display, double-click on the "Grab 48" icon.
- A.2.. Use buttons in the "HP48 Grabber" display on the IBM-compatible monitor first to set Port to "COM1" and Baud to "9600" (under "Options").

B. For the HP48G/GX:

- B.1. Turn on the HP48G/GX and set FLAG -34 by: VAR, 34, +/-, PRG, TEST, NXT, NXT, SF.
- B.2. Construct the view window of the HP48G/GX to appear as required for the print, then press: RS, I/O, highlight "Print display" as:



then press, OK.

- B.3. Save the screen display on a selected disk drive as either an appropriately titled ".BMP" or ".TIFF" file.

Section 9.2.2. Obtaining Prints of the HP48S/SX View Window

A. For the IBM-compatible Computer:

- A.1. Turn the computer on, and at the WINDOWS display, double-click on the "Grab 48" icon.

A.2.. Use buttons in the "HP48 Grabber" display on the IBM-compatible monitor first to set Port to "COM1" and Baud to "9600" (under "Options").

B. For the HP48S/SX:

B.1. Press LS, I/O, SETUP and configure the "I/O setup menu" to be as shown on right::

B.2. Set FLAG -34 by: VAR, 34, +/-, PRG, TEST, NXT, NXT, SF

B.3. Construct the view window of the HP48S/SX to appear as required for the print, then press: LS, PRINT, PRLCD

B.4. Save the screen display on a selected disk drive as either an appropriately titled ".BMP" or ".TIFF" file.

I/O setup menu	
IR/wire:	wire
ASCII/binary:	binary
baud:	9600
parity:	none 0
checksum type:	3
translate code:	1

Index

A

absolute humidity, 149
addition with conditional branching, 111
alarms, 9
algebraic solutions, 46
ALPHA characters, 4, 5
angl, 24
area, 23-25, 27-31, 70, 73, 92-94, 100
area of rectangle, 93
arithmetic with complex units, 32
array, 1, 56, 60-63
array functions, 61, 62
automatic curve-fitting, 174-177
automatic program execution, 105

B

Basic Operations, 11
 Number Control, 11
 UNDO, 16
 arithmetic, 12
 RPN, 11
 STACK Position, 14
 Tools for STACK Control, 16
 UC, DC, LC, and RC, 19
basic statistics, 56-58, 78
Brush Turkey, 51, 54
BTU, 24, 25, 27, 29-33

C

calculations of percent, 19

calculations with complex units, 32
calendar, 9, 178
cancl, 3, 7, 9, 73, 76, 77
CASE/THEN/END/END, 117
Checks, 153-156
clearing stack lines, 18
clock, 7, 9, 71, 128
Clock, Calendar and Alarms, 9
 Setting, 9
complex units, 29, 32
complex units conversions, 29
conditional branching, 111
controlled looping, 120, 124
conv, 27, 28, 30
correlation coefficient, 60, 65-69
course score, 87, 89-91, 101, 102
creating customized menus, 30
CST, 23, 29, 31-33
curve-fitting, 61, 65, 66
customized menus, 23, 29, 30, 32, 33, 51

D

data requests, 47, 92, 94, 135
DEBUG, 100
decision structures, 108, 119
differentiating, 167-169
directory structure, 56, 75
Directories1, 22, 23, 74, 75, 82
 HOME directory, 74
 LS MEMORY, 76
 RS MEMORY, 75
 Basic Concepts, 74

Directory Structure and
Construction, 75
Subdirectory Functions, 76
DO/UNTIL, 121, 122, 125, 135
double conditionals, 113
duplicating and counting the stack, 18

E

Editing Stored Data, 60
elec, 24, 25
energy, 25
enrg, 24, 30, 31
Equation Solutions with LS SOLVE, 39
Equation Solutions with RS SOLVE, 38
Equations, 35
 Complex Equation, 37
 LS SOLVE, 39
 more than 2 variables, 49
 RS SOLVE, 38
 two variables, 48
 Writing an Equation and Solving
 it, 35
Alternative for Algebraic
 Solutions, 46
Alternative Way to Solve, 43
Amending an Existing Equation,
 42
Display and Storage, 47
LS DEF Operation, 47
Making Changes in Stored
 Equations, 42
Simple Equation, 36
single variable, 48
Solving an Equation Algebraically,
 45, 46
Using EDIT 42

equation solutions, 6, 38, 39, 52, 70
errors, 41, 75, 80, 99, 100, 107, 136
 white letters, 3
examination score, 83, 85
exponential fit, 66, 67, 72

F

File Transfer and Printing, 178
 between HP48S/SXs, 186
 from an HP48G/GX to an
 HP48S/SX, 185
 between HP48S/SX and
 HP48G/GX, 184
 between HP48Gs, 183
 from the HP48G to a Tabletop
 Computer, 180
 HP48S/SX, 178
 Serial Interface, 178
 Tabletop Computer to the HP48G,
 182
file transfer and printing, 178-186
finding program errors, 99, 100
FLAGS, 127
 Basic Concept, 127
 Complex Choices, 132
 Controlled FLAG Clearing, 139
 Controlled FLAG Review, 137
 Keeping Track, 137
 system-defined FLAGS, 128
 user-defined FLAGS, 129
 What They Do, 128
 Where FLAGS Are, 128
flag clearing, 138
flag review, 136
flag status, 113, 128, 130, 132, 133, 137
flag test, 128, 139
force, 24, 25, 27, 28

G - K

graphical solution, 53, 55, 72, 73
home directory, 21, 22, 31, 32, 35, 40-42,
51, 55, 70, 74-77, 178
HP48S/SX, 179, 180, 185-189
IBM-compatible, 179-181, 183, 188, 189
IF/THEN/ELSE/END, 108, 110, 113,
116, 127, 135
IF/THEN/END, 108, 110, 113, 119, 127,
131
integration, 160-162
Kermit, 181-184
Keyboard, 1-4, 9, 15, 16, 19, 31, 39, 47,
77, 106
 primary function, 2
 white letters, 2
 alphanumeric, 2
 left-shift function, 2
 right-shift function, 2

L

leng, 2, 23-25
length, 2, 23, 25, 29, 93-96, 106, 111
light, 24, 25
linear fit, 66, 67
Link48, 179-182
Local Variables, 85
local source file, 182
local variables, 75, 85
logarithmic fit, 66, 67
loop structures, 119
LS DEF Operation, 47
LS UNITS Menu, 27
LS CLEAR, 5, 11, 14, 16, 18, 19, 40, 43,
56, 60
LS DEF 47, 48, 50, 51
LS MEMORY, 75, 76

LS SOLVE, 39, 70
LS STACK, 16-19
LS STAT, 57-62, 65, 66, 77
LS SWAP, 15, 25
LS UNITS, 27, 28

M - Q

making predictions, 69
many conditions, 117
mass, 23-25, 29
Menus for Program Writing, 82
mound temperature, 52-54
mystery solved, 70
number control, 11
number display, 8, 142
numerical solutions, 53, 84
percent calculations, 19-21, 68, 83, 86,
87, 89, 101, 110
Plant density, 157-165
Population Growth, 145
power fit, 66, 67
powr, 24, 30, 31
pressure, 25, 113, 115, 116
PRG menu, 81
primary function 2, 15
Prints of the HP48S/SX View Window,
187
programming options, 107
Programming, 79, 119
 Automatic Program Execution,
 105
 Finding Program Errors, 99
 Single-step Execution of a
 Program, 100
 Strategies, 101
 with RPN, 102
Addition with Conditional
 Branching, 111

Combining STACK and VAR
Data, 89

Controlled Choices, 108

Controlled Looping, 120

Data Entry with ENTER and
INPUT, 104

Data Requests, 92

Decision Structures, 108

DO/UNTIL, 121

Getting Started, 79

Local Variables, 85

Loop Structures, 119

Menus for Program Writing, 82

Options, 107

Other Ways to Enter Data, 103

PRG Menu, 81

Program Map, 79

Storing Equations and Data, 82

Tests with Double Conditionals,
113

Tests with Many Conditions, 117
with RPN, 101, 102

program map, 79

R

rad, 2, 24, 25

random number, 124, 142

relative humidity, 149-152

remote destination file, 182

reordering the stack, 18

Reverse Polish Notation, 11, 12

RPN, 11, 12, 15, 25, 56, 57, 101-103

RS Units Menu, 24

RS memory, 40, 42, 51, 75

RS SOLVE, 3, 38, 70

RS STACK, 16, 19

RS STAT, 57-61, 65, 77

RS UNITS, 2, 22-26, 28

S

sample problems, 50, 144

sample unit conversion, 28

serial, 178-182

Single Column, 54, 56, 58

Single-step Execution of a Program, 100

Solving Equations, 35

Special Cases, 51

Special Feature, 51

special symbols, 4

CALCULATOR MODES, 7

RS CHARS, 6

alphanumeric, 4

Generating Special Characters, 6

special characters, 4, 6

speed, 23-26, 29, 73

stack position, 14

STACK Control, 16

STACK REGISTER, 15

START/NEXT, 119

START/STEP, 119

statistics, 54, 56-58, 60, 78, 144

Array Functions, 61

Exponential Fit, 67

Linear Fit, 66, 67

Logarithmic Fit, 66

LS STAT Menu, 57

Power Fit, 66

Single Column of Numbers, 56

Basic Calculations, 60

correlation coefficient, 68

Curve-Fitting, 63, 65

Curve-fitting With Equation
Solutions, 70

Editing Stored Data, 60

Exponential Fit, 66

Linear Fit, 66

Logarithmic Fit, 67

- Making Predictions, 69
- Power Fit, 67
- Single Column of Numbers, 56
- Two Columns of Numbers, 60
- Using RS STAT, 58
- Subdirectory Functions, 76
 - Moving a File, 77
 - Change the Contents of a File, 77
 - Creating a Subdirectory, 76
 - Erase a Subdirectory File, 77
- system-defined FLAGS, 128

T - Z

- temp, 24-26, 59, 71, 72, 114-116
- temperature, 25, 26, 32, 52-54, 59, 70-73, 113, 116
- tumor treatment, 166-171
- two columns, 60
- undo, 16, 42
- Unit Conversions, 25
 - Complex Units Conversions, 29
 - Arithmetic with Complex Units, 32
 - Calculations with Complex Units, 32
 - Creating a Customized Menu, 31
 - CST, 29
 - Customized Menus, 29
 - Mixing Built-in Units, 28
 - Using Customized Menus, 30
- units menu, 24, 27, 29, 30
- units, 2, 10, 22-34, 72, 74, 93, 95, 132, 135-137
- unit conversions, 22, 25, 26, 27-29, 30, 51
- useful statistics, 78, 172-173
- user-defined FLAGS, 129
- Variables with Units, 22
 - LS UNITS Menu, 27

- RS UNITS, 22
- RS Units Menu, 24
- Unit Conversions, 25
- variables as numbers, 21
- variables with units, 22
- visc, 24
- vol, 24, 25, 28, 96-100
- volume, 25, 27-29, 95-98
- watts, 29-32, 52, 54
- WHILE/REPEAT/END, 122, 124
- Writing and Solving Equations, 35
- writing an equation, 35, 38
- X stack register, 15
- Y stack register, 15

MASTERING THE HP 48G/GX

A Step-By-Step, Easy to Read Introduction to
Operating and Programming the HP 48G/GX

Thomas Adams

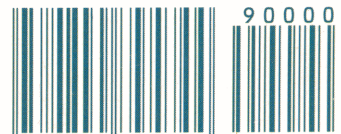
About the Author

Thomas Adams is a Professor of Physiology at Michigan State University in East Lansing, Michigan. He is the author of numerous publications in the areas of physiology and medicine, and has published three books on the operation and programming of handheld computers, including *Programming the HP-41C/CV/CX*, *Handheld Computers in Physiology and Medicine*, and *An Easy Guide to the HP48*. He currently teaches a course at MSU in the use of handheld computers in physiology and medicine.



KENDALL/HUNT PUBLISHING COMPANY
Dubuque, Iowa

ISBN 0-8403-9534-5



9 780840 395344