# Solving

# *Electronic Fundamentals: Circuits, Devices, and Applications, 8th Edition*
**By Thomas Floyd**

## The HP Way

A Beginning Tutorial for the HP-48GX, HP-48gII, and HP-49g+
by Gary D. Snyder

## Preface

The intent of this tutorial is to acquaint the reader with some of the more useful features of the HP-48 calculator as they relate to electronics technology and engineering. This tutorial assumes that the user is somewhat familiar with the operation of the calculator, but if not the examples should help the user to quickly familiarize himself or herself with how the calculator work. Unfortunately it is not possible for this short introduction to detail all the functions, features, and nuances of the HP-48. Considering that the calculator manuals alone are over 800 pages a short work of less than 70 pages can do little more than scratch the surface and hopefully encourage the reader to explore the uncovered features on his or her own.

This tutorial actually covers 3 calculators but collectively refers to them as the HP-48. This is because many features and operations of the HP-48gII and HP-49g+ are virtually identical to the HP-48GX. Consequently the sections that follow generally use “HP-48” to refer to all three calculators. If something is unique to a particular calculator, the discussion will use the calculator’s full model number to avoid confusion.

Although the operations of all three calculators are very smilar the actual keystrokes will vary somewhat. Consequently each detailed example will show two sets of keystrokes, designated as “HP-48” and “HP-49”. The “HP-48” keystrokes are for the HP-48GX only, while the keystrokes shown for the ”HP-49” are for both the HP-48gII and HP-49g+. This may seem confusing at first, but the design of the HP-48gII is much closer to that of the HP-49g+ than it is to the design of the older HP-48GX and the keystrokes for the HP-48gII and HP-49g+ are (at least for this tutorial) identical. If it helps to simplify things, you can consider the HP-48gII as more of an “HP-49 Lite” than a “HP-48GX Plus”.

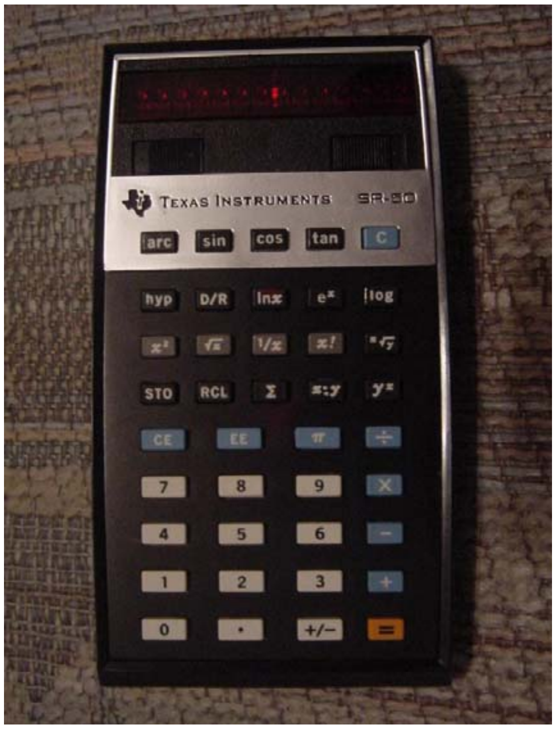
# 1. Introduction

For nearly as long as there have been calculations, humans have found ways to make calculating faster, more accurate, and, above all, easier. Fingers were probably the first devices used to perform simple computations with small numbers. In time the numbers became too large for fingers to be practical, and small stones arranged in rows on a flat surface were used. From the Latin words for the fingers (*digiti*) and small stones (*calculi*) used by these early methods came current English words that are often associated with numbers, such as *digit* and *calculate*.

Over the centuries calculations became more complex and computing devices became more sophisticated as well. For the most part these devices were purely mechanical in their operation, although in time some machines were powered by electricity. It wasn't until the 1940's that electricity itself became part of the computation process in electromechanical behemoths like ENIAC. In the 1960s desktop electronic calculators appeared, and handheld "four-bangers" capable of adding, subtracting, multiplying, and dividing were available by the early 1970s. Then, in 1972, Hewlett-Packard introduced the first scientific pocket calculator – arguably the original "PC".

## 1.1 The HP-35 Calculator

The HP-35 was a remarkable invention. While its capabilities seem rather limited by today's standards, it was the first handheld device that could calculate powers, roots, logarithms, and trigonometric values and do so with ten digits of accuracy. In fact, when developing the calculator, the designers were hard-pressed to find some device capable of checking the accuracy of their algorithms. It also offered floating point and scientific numeric formats, a dedicated key for $\pi$, a user memory for storing intermediate values during calculations, and a rechargeable battery pack that would become standard features on later scientific calculators, such as the Texas Instruments SR-50.



If you compare the keyboard of the HP-35 on the left with that of the SR-50 on the right, you will notice that the HP-35 has no "=" key but does have a large key marked "ENTER↑". This is because the HP-35 and its successors used a method of data entry referred to as "reverse Polish notation", or RPN. This distinctive feature was unlike that of most other calculators, but was so effective that virtually every HP calculator developed since the HP-35 (including the HP-48GX and HP-49g+) use it.

Despite its fairly high initial price of $345 (later reduced to $295 when the more advanced HP-45 was introduced) the HP-35 was a popular seller. Even when competitor products like the SR-50 provided more functions at a lower price, the high quality and rugged nature of the HP calculators had become almost legendary and made them the calculators of choice for those who could afford them.

## 1.2 Later Developments

Following the introduction of the HP-35, calculators rapidly advanced in capability and features. Only 18 months after the HP-35 was launched Hewlett-Packard marketed its first user-programmable pocket calculator, the HP-65. As remarkable a technical accomplishment as this was, even more astonishing was this calculator's ability to read and write miniature magnetic cards with an integrated card reader. Further calculator improvements over the years included more built-in functions, non-volatile data and program storage, greater numeric ranges, alphanumeric data entry and displays, symbolic solving, graphing features, use of objects, and serial communications capabilities. Today's high-end calculators (such as the Texas Instruments TI-89 and Hewlett-Packard HP-49g+) are far more than their simple number crunching predecessors of the 1970's, and are more correctly termed pocket computers than pocket calculators.

One drawback of this increased power is greatly increased complexity. The HP-49g+, for example, has over 300 built-in functions and equations and a user's manual well over 800 pages long. The purpose of this tutorial is to help familiarize you with the operation and features of the HP-48GX and HP-49g+ that directly apply to solving the types of problems found in *Electronic Circuits Fundamentals, 8th Edition*, by Thomas Floyd. In doing so, it will hopefully build a solid understanding of the calculator that will assist you in other problem-solving areas.

## 1.3 The HP-48GX, the HP-48gII, and the HP-49g+

As of this writing the HP-48GX and its successors, the HP-48gII and HP-49g+, are Hewlett-Packard's most advanced programmable graphing scientific calculators. Hewlett-Packard discontinued production of the HP-48GX in 2003 but this calculator can still be acquired at a reasonable price from various sources. The capabilities and operation of both calculators are quite similar and will both be covered in this tutorial. This tutorial will, where appropriate, also discuss the special enhancements of the HP-49g+ but either calculator is acceptable. In fact, as you will learn, the problems found in *Electronic Fundamentals: Circuits, Devices, and Applications, 8th Edition* require the use of only a small fraction of what these calculators can do.

A word of caution is in order here. Be aware that a calculator - any calculator - can only do what the user tells it to do. Using a calculator to solve a problem is no substitute for understanding how to approach a problem. You should always try to thoroughly understand the underlying concepts of each problem before attempting to work through it. There is no use in trying to obtain an answer if the answer or its associated problem have no meaning to you.

# 2. Pre-Flight

Although the HP-48GX, HP-48gII, and HP-49g+ are very similar and comparable in capability, they do differ somewhat in their specific features. As you become more comfortably with the operation of your calculator you will find that there various settings you can use to customize the calculator to suit your own preferences. For simplicity this tutorial will assume that your calculator is set to display values in ENG 2 (that is, engineering notation with up to 2 decimal places of accuracy). If you have an HP-49gII or HP-49g+ it also assumes that the calculator is set for RPN operation and APPROXIMATE rather than EXACT mode. Although you do not have to use these settings it will make it easier to verify that you have followed the instructions correctly. The following sections will explain how to specify these settings.

## 2.1 Setting the Calculator for ENG 2 Mode

To set the calculator's number format for ENG 2 enter the following keystrokes. "HP-48" indicates the keystrokes for the HP-48GX, whereas "HP-49" shows the keystrokes for the HP-48gII and HP-49g+.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MODES | MODE | Enter the CALCULATOR MODES screen of the calculator. The HP-48-GX will start in the NUMBER FORMAT menu. The HP-48gII and HP-49g+ will start in the OPERATING MODE menu. |
|  | ▼ | Cursor down to the NUMBER FORMAT menu in the HP-48gII and HP-49g+. The display will show "CHOOSE NUMBER DISPLAY FORMAT" (HP-48GX) or "Choose number display format" (HP-49gII and HP-49g+). |
| α E | ALPHA E | Enter "E" to specify ENGINEERING mode. The NUMBER FORMAT indicator should show "Eng". |
| ► | ► | Cursor over to the decimal places specifier. The display will show "ENTER DECIMAL PLACES TO DISPLAY" (HP-48GX) or "Choose decimal places to display" (HP-48gII and HP-49g+); |
| 2 ENTER | 2 | Specify 2 decimal places for the display. Note that the HP-48gII and HP-49g+ do not require you to press ENTER. |
| OK | OK | Confirm the setting. This will return you back to the standard display. |

## 2.2 Setting the Soft Menu Flag (HP-48gII and HP-49g+ Only)

You can configure HP-48gII and HP-49g+ menus in two different ways. The first way is to have the calculator display menu choices as items in a numbered list. With this option you would choose a menu item by pressing the number key that corresponds to the number of the item. The second way is to have the calculator display the items as "tabs" or "buttons" at the bottom of the display, which is the way that the HP-48GX displays them. In this configuration you would select a menu item by pressing the "soft key" (F1 through F6) directly under the desired item. Which method you choose is strictly a matter of preference, but for consistency between all three calculators this tutorial assumes that the calculator is configured for the second (tab) option. To set this particular option you set Flag 117 as follows:

| HP-49 | Comments |
|---|---|
| MODE | Enter the MODE screen of the calculator. The HP-48gII and HP-49g+ will start in the CALCULATOR MODES screen. The display will show “Choose calculator operating mode”. |
| FLAGS | Press the FLAGS softkey to access the calculator SYSTEM FLAGS screen. |
| ▲ | Press the “up” cursor key several times until the option for Flag 117 is highlighted. If the option is checked and says “Soft MENU” the calculator is already properly configured. Press CANCL to exit. Otherwise, proceed per the instructions below. |
| ✓CHK | If Flag 117 is not checked and says “CHOOSE boxes”, press the “✓CHK” button to select the flag and toggle the mode to “Soft MENU”. |
| OK | Confirm the Soft MENU setting. This will return you to the CALCULATOR MODES screen. |
| OK | Confirm the CALCULATOR MODES settings. This will return you back to the standard display. |

## 2.3 Setting the Calculator for RPN Operation (HP-48gII and HP-49g+ Only)

To set the calculator for RPN operation and APPROXIMATE mode enter the following keystrokes.

| HP-49 | Comments |
|---|---|
| MODE | Enter the MODE screen of the calculator. The HP-48gII and HP-49g+ will start in the CALCULATOR MODES screen. The display will show “Choose calculator operating mode”. |
| ALPHA<br>R | Enter “R” to specify RPN mode. The OPERATING MODE menus should show “RPN”. |
| CAS | Select CAS to enter the CAS MODES screen. The calculator will start in the INDEP VAR menu. The display will show “Enter independent variable name”. |
| ▼<br>▼ | Cursor down to the _APPROX selection. The display will show “Perform approx calculations?” |

| | |
|---|---|
| ✓CHK | Select the "✓CHK" button to set approximate mode |
| OK | Confirm the CAS MODES settings.  This will return you to the CALCULATOR MODES screen. |
| OK | Confirm the CALCULATOR MODES settings.  This will return you back to the standard display. |

This will standardize the settings for all three calculators.

# 3. Calculator Basics

There are three steps to using any calculator. These are

1. entering the information to be processed,
2. processing the information, and
3. obtaining the results of the calculation.

Manufacturers try to design their calculators to make these operations fairly intuitive (after all, calculators are meant to simplify calculations, not make them more difficult). In reality every calculator will require some time and practice before using it becomes second nature. Understanding how the HP-48[1] handles data is essential to using it effectively.

[1]For simplicity this tutorial will use “HP-48” when discussing topics common to both the HP-48GX and HP-49g+.

## 3.1 Objects

The first thing to understand is that the HP-48 works with objects. Unless you have studied object-oriented programming you may find the concept of objects difficult to understand, but only because objects are something that you’ve taken for granted your entire life. In the real world nearly everything – such as a stone, a tree, or a machine - is an object. Every object possesses certain features and properties that are characteristic of that type of object and to more general classes of objects to which they belong (for example, a *tree* is also a *plant*). The information with which the HP-48 deals are also objects. Those objects, like real-world objects, have features and properties that are specific to that type of object.

As a simple illustration of an object, consider the number 4. You may choose to write it as “four” or “IV” instead, but how you choose to express it is irrelevant. Regardless of how you represent it, 4 is 1 more than 3, 4 added to itself will give 8, and 4 is an integer with all the mathematical properties of an integer. When you think about 4 in terms of what it *is* rather than how it *looks*, you are thinking about 4 as an *object*. The HP-48 deals with objects, rather than numbers (although it might seem that way at times).

Why is this so important? There are two very good reasons.

First, objects allow the calculator to work with a variety of data types in a consistent manner. The earliest calculators were able to work with integer and floating point values in the same way so that calculating 3 x 2 was no different than calculating 3.141592654 x 2.718281828. Objects allow the HP-48 to extend this consistent approach to other data types such as complex numbers, matrices, and lists.

Second, objects allow the HP-48 to clearly differentiate between data and how that data is represented. This may not seem very important or even necessary at the moment, but you will find it quite useful in certain situations.

## 3.2 Data Entry

Over the years calculators have used two major types of data entry with some minor variations.

The first, and probably the most familiar, type of data entry is *algebraic entry,* or the *algebraic operating system* (or AOS). With this type of system users enter equations into the calculator they same way they appear on paper. For example, the user would solve the time-honored equation “1 + 2 = ?” with an AOS calculator by entering the following keystrokes:

$\boxed{1}$ $\boxed{+}$ $\boxed{2}$ $\boxed{=}$

When the user presses the “=” key the calculator would display the answer (namely 3). This is referred to as *infix* notation, as the operator is located between the operands.

The second type of data entry is *reverse Polish notation* (or RPN), which is a type of *postfix* notation. With postfix notation the user would specify the operands before the operator. To solve the above equation using an RPN calculator such as the HP-35, the user would enter the following keystrokes:

$\boxed{1}$ $\boxed{\text{ENTER}\uparrow}$ $\boxed{2}$ $\boxed{+}$

While the HP-48 is able to work with both algebraic entry as well as RPN, its “native” operating system is RPN. Because of this, most of the examples given in this tutorial will be shown using RPN.

## 3.3 The Memory Stack

Every calculator must have some place to store data while performing calculations. The HP-35 and later Hewlett-Packard models used a four-register stack consisting of X, Y, Z, and T (or “top”) registers as shown:

| |
|---|
| T |
| Z |
| Y |
| X |

The calculator displayed the contents of the X register, while the other registers were used to hold intermediate results during calculations. For the RPN addition example shown above the contents of the stack would change as indicated as the keys were pressed:

| | | | | |
|---|---|---|---|---|
| t | t | z | z | z |
| z | z | y | y | z |
| y | y | 1 | 1 | y |
| x | 1 | 1 | 2 | 3 |
| | $\boxed{1}$ | $\boxed{\text{ENTER}\uparrow}$ | $\boxed{2}$ | $\boxed{+}$ |

Note that x, y, z, and t are the original contents of the X, Y, Z, and T registers. The calculator automatically manages the values in the stack as calculations are performed.

The HP-48 uses a memory stack system of holding data during calculations which is similar to the register stack of earlier Hewlett-Packard calculators, but with two important differences. First, whereas the HP-35 registers could hold only numbers, the HP-48 can hold any object in any level of its memory stack. Second, while the register stacks in earlier Hewlett-Packard calculators were limited to 4 registers, the number of levels in the HP-48 memory stack is limited only by the amount of free memory.

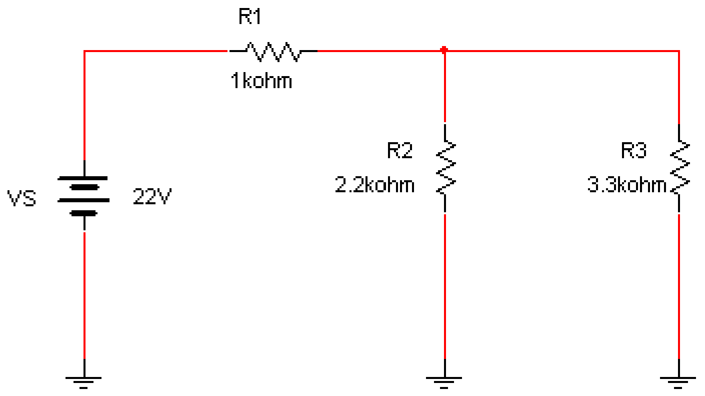
Because calculator operations almost always involve the memory stack, it is important for you to understand how the memory stack works and how to control its contents when using the HP-48. As you will see later, the HP-48 includes a number of stack operations (such as DUP, SWAP, and ROLL) that manipulate and affect the stack contents. In later sections you will learn how this can simplify using your calculator.

# 4. Basic Calculations

Despite the many advanced capabilities of the HP-48, you will probably find yourself using it to perform basic arithmetic most of the time. This shouldn't be surprising. Even operations like polynomial evaluation, matrix arithmetic, and convolution boil down to adding, subtracting, multiplying, and dividing. The calculator's job is to relieve you of the drudgery of crunching the numbers. Your job is to learn how to communicate the problem to the calculator. The following examples will step you through some basic electronics calculations to familiarize you with basic calculator operations.

Note: Before proceeding, be sure to set your calculator's number format to "Engineering 2" mode so that your answers agree with the values shown in the examples.

**Example 4-1:** Calculate the total resistance in the series-parallel circuit below.



**Solution:** Usually you would first enter in the values of R2 and R3, calculate the total parallel resistance, and then add in the series resistance of R1 to find the total resistance. Just for fun, however, let's enter in the resistor values in the order R1, R2, and R3 and process each value as needed.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1000 | 1000 | 4:<br>3:<br>2:<br>1: | Enter value of R1 |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 1.00E3 | Enter value of R1 into the stack |
| 2200 | 2200 | 4:<br>3:<br>2:<br>1: 1.00E3 | Enter value of R2 |
| 1/X | 1/X | 4:<br>3:<br>2: 1.00E3<br>1: 455.E-6 | Find the conductance of R2 for parallel calculation |

| | | | |
|---|---|---|---|
| 3300 | 3300 | 4:<br>3:<br>2: 1.00E3<br>1: 455.E-6 | Enter value of R3 |
| 1/X | 1/X | 4:<br>3: 1000<br>2: 455.E-6<br>1: 303.E-6 | Find the conductance of R3 for parallel calculation |
| + | + | 4:<br>3:<br>2: 1.00E3<br>1: 7.58E-6 | Find the total conductance of the parallel branches |
| 1/X | 1/X | 4:<br>3:<br>2: 1.00E3<br>1: 1.32E3 | Find the total resistance of the parallel branches |
| + | + | 4:<br>3:<br>2:<br>1: 2.32E3 | Find the total circuit resistance |

The final answer, while correct, was not really the point of this example. The example was instead intended to illustrate some important features of the calculator.

First, note that only first value required you to press ENTER to place it into the memory stack. R2 and R3 were automatically entered into Level 1 when you calculated their conductances by using the 1/X key. This is true in general. Pressing a function key will place the results of the function into Level 1 of the stack, whereas you must press the ENTER key to enter values into the stack.

Second, note that although you entered the resistor values as whole numbers (1000, 2200, and 3300) the display showed them in the Engineering format you specified (1.00E3, 2.20E3, and 3.30E3). It also displayed the intermediate and final results in the same Engineering format. While this is true for calculators in general, it underscores the point that the HP-48 is working with what the number is, rather than how it is represented. When you keyed in the sequence “1000” the calculator accepted it as the numeric object 1000 and displayed it as “1.00E3”. Had you specified another number format it would have represented it differently but the number itself would have been unaffected.

Finally, note how the contents of the memory stack changed during the calculations. As you entered data the contents of each level were shifted up (referred to as “stack lift”) and as the data was processed the contents of the stack dropped back down again. In this case only 3 levels of the memory stack were required, but more involved computations may take many more. The calculator manages the stack automatically, but in some cases you may wish to intercede and change the contents of the stack manually.

Consider the following example:

**Example 4-2:** What is the total current for the circuit in Example 4-1?

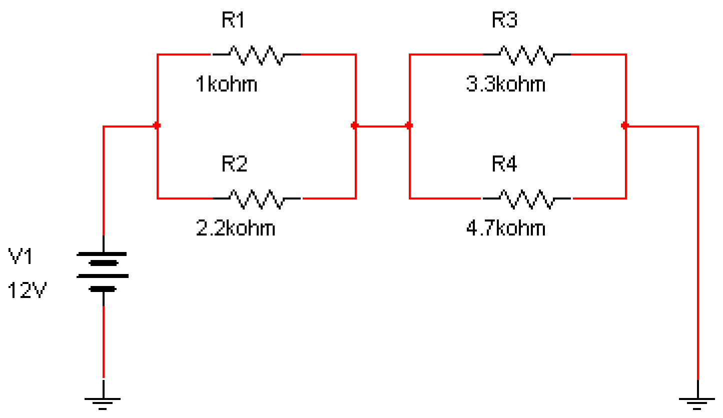
**Solution:** One way to solve this would be to take the reciprocal of the total resistance (i.e., the total conductance) and multiply it by the supply voltage. Another way would be to enter the supply voltage, somehow exchange the contents of Levels 1 and 2, and then divide the voltage in Level 2 by the total resistance in Level 1. Let’s do it that way.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 22 | 22 | 4:<br>3:<br>2:<br>1: 2.32E3 | Enter the value of VS |
| ENTER | ENTER | 4:<br>3:<br>2: 2.32E3<br>1: 22.0E0 | Enter the value of VS into the stack |
| ▶ | ▶ | 4:<br>3: 22.0E0<br>2: 22.0E0<br>1: 2.32E3 | Swap the values of Levels 1 and 2[2] |
| ÷ | ÷ | 4:<br>3:<br>2:<br>1: 9.48E-3 | Calculate the total current |

[2] Although the HP-48GX keyboard implies that SWAP is a right shift function and requires you to press the right shift (↱) key first this is actually not necessary in user mode, as you have just seen. In program mode you must use the shift key.

Swapping the values of Levels 1 and 2 happens quite frequently, so SWAP is a useful function to know. Let's try a slightly more extreme case of stack manipulation.

**Example 4-3:** Calculate the total resistance of the series-parallel circuit below.



**Solution:** Once again, let's enter all the resistor values into the calculator before proceeding and manipulate the stack contents as needed to complete the calculation. This time you'll use another stack operation to move the data to where you need it.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1000 | 1000 | 4:<br>3:<br>2:<br>1: | Enter value of R1 |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 1.00E3 | Enter value of R1 into the stack |
| 2200 | 2200 | 4:<br>3:<br>2:<br>1: 1.00E3 | Enter value of R2 |
| ENTER | ENTER | 4:<br>3:<br>2: 1.00E3<br>1: 2.00E3 | Enter the value of R2 into the stack |
| 3300 | 3300 | 4:<br>3:<br>2: 1.00E3<br>1: 2.00E3 | Enter value of R3 |
| ENTER | ENTER | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 3.30E3 | Enter the value of R3 into the stack |
| 4700 | 4700 | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 3.30E3 | Enter the value of R4 |
| ENTER | ENTER | 4: 1.00E3<br>3: 2.20E3<br>2: 3.30E3<br>1: 4.70E3 | Enter the value of R4 into the stack |
| 1/X | 1/X | 4: 1.00E3<br>3: 2.20E3<br>2: 3.30E3<br>1: 213.E-6 | Find the conductance of R4 for parallel calculation |
| ► | ► | 4: 1.00E3<br>3: 2.20E3<br>2: 213.E-6<br>1: 3.30E3 | Swap the contents of Levels 1 and 2 |
| 1/X | 1/X | 4: 1.00E3<br>3: 2.20E3<br>2: 213.E-6<br>1: 303.E-6 | Find the conductance of R3 for parallel calculation |
| + | + | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 516E-6 | Find the total conductance for R3 and R4 in parallel |
| 1/X | 1/X | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 1.94E3 | Find the equivalent resistance of R3 and R4 in parallel |

What now? It seems that the values of R1 and R2 are beyond the reach of the SWAP function, as indeed they are. Fortunately, there are other stack operations that can move the data where we want it. Two of these functions are called ROLL and ROLLD (or “roll down”). Unlike SWAP, which exchanges the contents of Levels 1 and 2, ROLL moves the value of a specified level into Level 1 and moves, or “rolls”, the contents of the lower levels up one level. Similarly, ROLLD moves the value of Level 1 into a specified level and rolls the contents of the lower levels down one level. In this case, we want to move the contents of Level 1 into Level 3, and move Levels 2 and 3 down so we will use ROLLD.

Note that we could use ROLL to roll the stack up just as well, but we would need to use it twice to move the contents of Level 1 to Level 3.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 3 | 3 | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 1.93E3 | Specify Level 3 as the target level for Level 1 |
| ENTER | ENTER | 4: 1.00E3<br>3: 2.20E3<br>2: 1.94E3<br>1: 3.00E0 | Enter value of ROLLD parameter into the stack |
| α<br>α | ALPHA<br>ALPHA | 4: 1.00E3<br>3: 2.20E3<br>2: 1.94E3<br>1: 3.00E0 | Lock the keyboard into alphabetic mode |
| ROLLD | ROLLD | 4: 1.00E3<br>3: 2.20E3<br>2: 1.94E3<br>1: 3.00E0 | Specify the ROLLD function (by typing in R O L L D) |
| α | ALPHA | 4: 1.00E3<br>3: 2.20E3<br>2: 1.94E3<br>1: 3.00E0 | Turn off the keyboard alphabetic mode |
| | | | |
| ENTER | ENTER | 4:<br>3: 1.94E3<br>2: 1.00E3<br>1: 2.20E3 | Execute the ROLLD function |
| 1/X | 1/X | 4:<br>3: 1.94E3<br>2: 1.00E3<br>1: 455.E-6 | Calculate the conductance of R2 |
| ▶ | ▶ | 4:<br>3: 1.94E3<br>2: 455.E-6<br>1: 1.00E3 | Swap the contents of Levels 1 and 2 |
| 1/X | 1/X | 4:<br>3: 1.94E3<br>2: 455.E-6<br>1: 1.00E-3 | Calculate the conductance of R1 |
| + | + | 4:<br>3:<br>2: 1.94E3<br>1: 1.45.E-3 | Find the total conductance of R1 and R2 in parallel |

| | | | |
|---|---|---|---|
| 1/X | 1/X | 4:<br>3:<br>2: 194E3<br>1: 688.E0 | Find the equivalent resistance of R1 and R2 in parallel |
| + | + | 4:<br>3:<br>2:<br>1: 2.63E3 | Find the total circuit resistance |

You may be wondering whether manipulating the stack like this is really necessary. After all, you could have calculated the total resistance just as easily – even more so – by using a procedure similar to the following:

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1000 | 1000 | 4:<br>3:<br>2:<br>1: | Enter value of R1 |
| 1/X | 1/X | 4:<br>3:<br>2:<br>1: 1.00E-3 | Calculate the conductance of R1 |
| 2200 | 2200 | 4:<br>3:<br>2:<br>1: 1.00E-3 | Enter value of R2 |
| 1/X | 1/X | 4:<br>3:<br>2: 1.00E-3<br>1: 455.E-6 | Calculate the conductance of R2 |
| + | + | 4:<br>3:<br>2:<br>1: 1.45E-3 | Find the total conductance of R1 and R2 in parallel |
| 1/X | 1/X | 4:<br>3:<br>2:<br>1: 688.E0 | Find the equivalent resistance of R1 and R2 in parallel |
| 3300 | 3300 | 4:<br>3:<br>2: 688.E0<br>1: 3.30E3 | Enter value of R3 |
| 1/X | 1/X | 4:<br>3:<br>2: 688.E0<br>1: 3.03E-6 | Calculate the conductance of R3 |
| 4700 | 4700 | 4:<br>3:<br>2: 688.E0<br>1: 3.03E-6 | Enter the value of R4 |
| 1/X | 1/X | 4:<br>3: 688.E0<br>2: 3.03E-6<br>1: 213.E-6 | Calculate the conductance of R4 |

| | | | |
|---|---|---|---|
| + | + | 4:<br>3:<br>2: 688.E0<br>1: 516.E-6 | Find the total conductance of R3 and R4 in parallel |
| 1/X | 1/X | 4:<br>3:<br>2: 688.E0<br>1: 1.94E3 | Find the equivalent resistance of R3 and R4 in parallel |
| + | + | 4:<br>3:<br>2:<br>1: 2.63E3 | Find the total circuit resistance |

The point is well-taken. RPN is easily able to handle equations of this type by processing data as you enter it and placing the intermediate results in the memory stack until they are needed. So why be concerned with manipulating data in the stack using SWAP, ROLL, and other stack functions?

The key to this question lies in the phrase “processing data as you enter it”. The second procedure is clearly simpler, shorter, and easier to understand than the first and would undoubtedly be the procedure of choice if you were manually entering data into the calculator. But suppose you didn’t enter the data into the memory stack one value at a time. Suppose you entered the data all at once, or (even more intriguing) something else loaded data into the stack for you? You would then be forced to work with data already on the stack, rather than enjoying the luxury of entering it at your leisure in whatever order you wished. In such a situation stack manipulation would be unavoidable.

In the next section you will examine just such a situation where this is the case.

# 5. Basic Programming

By now you have probably begun to appreciate the calculating power of your Hewlett-Packard calculator and feel that it is a vast improvement over calculating by hand. Even so, some calculations are still quite tedious to work through, especially if you had to work through them several times again. This is where another aspect of the HP-48 comes in quite useful. Rather than work the same calculation over and over again by hand, you can program the calculator to remember how to perform the calculation and have it repeat the sequence of keystrokes with the press of a single button. All you have to do is provide the new data for each calculation.

## 5.1 Your First Simple Program

As a demonstration, consider the case of calculating the length of the hypotenuse of a triangle using Pythagorean's Theorem. As you rememember, the hypotenuse C of a right triangle having legs of length A and B is

$$C = \sqrt{(A^2 + B^2)}$$

The normal RPN method of solving this equation would be to enter the values of A and B, square them, add the two results, and take the square root of the sum. For the classic Pythagorean triangle with A = 3 and B = 4, for example, the sequence could be:

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 3 | 3 | 4:<br>3:<br>2:<br>1: | Enter value of A |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 3.00E0 | Place the value of A into the stack |
| 4 | 4 | 4:<br>3:<br>2:<br>1: 3.00E0 | Enter value of B |
| ENTER | ENTER | 4:<br>3:<br>2: 3.00E0<br>1: 4.00E0 | Place the value of B into the stack |
| ↰<br>$X^2$ | ↰<br>$X^2$ | 4:<br>3:<br>2: 3.00E0<br>1: 16.0E0 | Calculate the value of $B^2$ |
| ▶ | ▶ | 4:<br>3:<br>2: 16.0E0<br>1: 3.00E0 | Swap the values of Levels 1 and 2 |
| ↰<br>$X^2$ | ↰<br>$X^2$ | 4:<br>3:<br>2: 16.0E0<br>1: 9.00E0 | Calculate the value of $A^2$ |

| | | | |
|---|---|---|---|
| + | + | 4:<br>3:<br>2:<br>1: 25.0E0 | Calculate the value of $A^2 + B^2$ |
| √X | √X | 4:<br>3:<br>2:<br>1: 5.00E0 | Find the square root |

As you can see the example once again places all the required values on the stack before performing the actual calculation. This is so that you will see that programming the HP-48 calculator is no different (in most ways) than working through the calculation by hand. The only difference is that you must specify that what you are entering is a program object so that the calculator does not try to execute each function as you enter them. You do so by enclosing the series of keystrokes in the double angle brackets (or “<< >>”) that define a program object.

To program the above example the keystrokes would be:

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↰<br><< >> | ↱<br><< >> | Specify a program object. Note that the HP-48 and HP-49 use different shift keys to do this. |
| ↰<br>$X^2$ | ↰<br>$X^2$ | Calculate the value of $B^{2.}$ Note that nothing is going onto the stack yet. Everything is being processed in the work area. |
| ↰<br>▶ | TOOL<br>STACK<br>SWAP | Swap the values of Levels 1 and 2. Recall that ▶ by itself only specifies SWAP in manual mode. For the HP-48 the left shift key must be used. For the HP-49 SWAP is accessed as a softkey through the STACK menu at the bottom of the display. |
| ↰<br>$X^2$ | ↰<br>$X^2$ | Calculate the value of $A^2$ |
| + | + | Calculate the value of $A^2 + B^2$ |
| √X | √X | Find the square root |
| ENTER | ENTER | Enter the program into the stack |

You should now see the program in Level 1 of the stack (the exact form will depend upon the calculator). The calculator knows that it is a program because of the “<< >>” enclosing the functions.

You could execute the program by pressing the EVAL key, but that would be pointless for two very good reasons. The first reason is that there is no data in the stack for the program to process. The second (and more important) reason is that, even if there were data to process, the program would disappear once the program is evaluated. You must first save the program so that you can run it over and over again. You do this by storing the program in a variable.

Before going on I'll take a moment to once again emphasize that you can store a program in a variable because the HP-48 deals with objects. A variable can hold not just numbers, but any type of calculator object that you can recall and use whenever you wish. You just need to remember what type of object you are storing. To keep things straight you can prefix the variable name with a lower case letter indicating what type of object is in the variable. For this program, give it the name "pPYTH" (for "program Pythagorus"). You do so by executing the following keystrokes.

| HP-48 | HP-49 | Comments |
|---|---|---|
| α<br>α | ALPHA<br>ALPHA | Lock the keyboard in alphabetic mode. |
| ↰<br>P | ↰<br>P | Use the left shift key to enter a lower-case "p" |
| PYTH | PYTH | Enter the rest of the program name in upper-case letters |
| α | ALPHA | Turn off the keyboard alphabetic mode |
| STO | STO | Store the program on Level 1 of the stack into the variable "pPYTH". |

As soon as you press the STO key the program in Level 1 should disappear from the stack, which is normal. But where did it go? The calculator has placed the program into memory as the variable "pPYTH". To find it again, simply press the VAR key to access the list of variables in memory. This will display a list of the variables currently in memory at the bottom of the screen. Just above the F1 key you should see a box labeled "pPYTH". To execute the program, you simply enter the variables into the memory stack and press the key below it as shown:

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 3 | 3 | 4:<br>3:<br>2:<br>1: | Enter value of A |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 3.00E0 | Place the value of A into the stack |
| 4 | 4 | 4:<br>3:<br>2:<br>1: 3.00E0 | Enter value of B |

| | | | |
|---|---|---|---|
| ENTER | ENTER | 4:<br>3:<br>2: 3.00E0<br>1: 4.00E0 | Place the value of B into the stack |
| VAR<br>pPYTH | VAR<br>pPYTH | 4:<br>3:<br>2:<br>1: 5.00E0 | Calculate the hypotenuse C for A = 3 and B = 4. Note that VAR is only necessary if the variables list is not visible. |

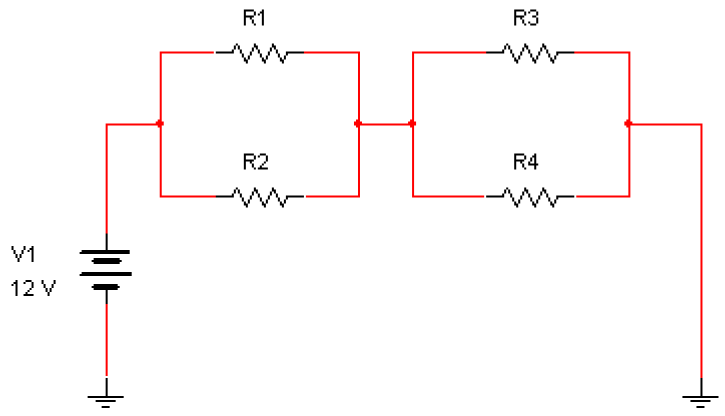
Now calculate the hypotenuse for a triangle with A = 12 and B = 5.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 12 | 12 | 4:<br>3:<br>2:<br>1: | Enter value of A |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 12.0E0 | Place the value of A into the stack |
| 5 | 5 | 4:<br>3:<br>2:<br>1: 12.0E0 | Enter value of B |
| ENTER | ENTER | 4:<br>3:<br>2: 12.0E0<br>1: 5.00E0 | Place the value of B into the stack |
| pPYTH | pPYTH | 4:<br>3:<br>2:<br>1: 13.0E0 | Calculate the hypotenuse C for A = 12 and B = 5. Pressing VAR is not necessary as you should be in the VAR menu from the last calculation. |

The point of all this should not be lost on you. Once you have entered a program, you can solve any number of similar problems with far less effort. In electronics, where you will need to solve many of the same basic circuit calculations over and over, programmability is a great time-saver.

You might be wondering at this point if it is possible to program the calculator so that you enter data as the program needs it, rather than entering all the data before you run the program. The answer is “yes”, but there are some reasons not to do so. One is that having the calculator pause to wait for you to enter data slows down program execution somewhat. Another is that there is always the chance of the user entering the wrong data at the wrong time unless you add more code to supply some sort of hint, or “prompt”. Yet another reason, and the most important, is that having all the data in memory beforehand allows the program to accept data generated by another program and proceed without user intervention. You saw, for example, that your program returned the answer on Level 1 of the stack. If you had another program called “pPOD” that required the value of a hypotenuse you could now run it without having to enter the data manually. Even better, you could have a program that executed first pPYTH and then pPOD as subroutines and not even see the intermediate result. This is called *structured programming* and it is a very powerful programming method. This tutorial will discuss more aspects of programming in a future section.

**Example 5-1:** Program the calculator to find the total resistance of the series-parallel circuit below.



**Solution:** Program the calculator using the problem solving method of Example 4-3 which assumes the stack contains the values of R1, R2, R3, and R4. Call the program “pPARINSER” (for program Parallel in Series”).

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↰ | ↱ | Enter programming mode |
| << >> | << >> | |
| 1/X | 1/X | Find the conductance of R4 for parallel calculation |
| ↰ | TOOL | Swap the contents of Levels 1 and 2 |
| ▶ | STACK | |
| | SWAP | |
| 1/X | 1/X | Find the conductance of R3 for parallel calculation |
| + | + | Find the total conductance for R3 and R4 in parallel |
| 1/X | 1/X | Find the equivalent resistance of R3 and R4 in parallel |

| | | |
|---|---|---|
| 3 | 3 | Specify Level 3 as the target level for Level 1 |
| SPC | SPC | Enter a space to separate the value 3 from the ROLLD command |
| α | ALPHA | Lock the keyboard into alphabetic mode |
| α | ALPHA | |
| ROLLD | ROLLD | Specify the ROLLD function (by typing in R O L L D) |
| α | ALPHA | Turn off the keyboard alphabetic mode |
| 1/X | 1/X | Calculate the conductance of R2 |
| ↰ | TOOL | Swap the contents of Levels 1 and 2 |
| ▶ | STACK | |
| | SWAP | |
| 1/X | 1/X | Calculate the conductance of R1 |
| + | + | Find the total conductance of R1 and R2 in parallel |
| 1/X | 1/X | Find the equivalent resistance of R1 and R2 in parallel |
| + | + | Find the total circuit resistance |
| ENTER | ENTER | Enter the program into Level 1 of the stack |
| α | ALPHA | Lock the keyboard into alphabetic mode |
| α | ALPHA | |

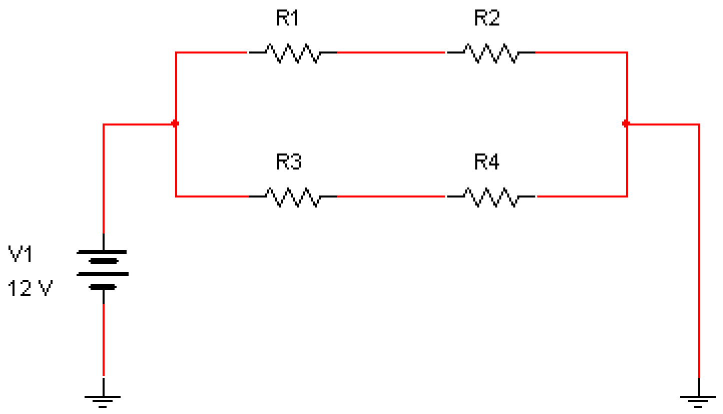
| | | |
|---|---|---|
| ↰<br>P | ↰<br>P | Use the left shift key to enter a lower-case “p” |
| PARINSER | PARINSER | Enter the rest of the program name in upper-case letters |
| α | ALPHA | Turn off the keyboard alphabetic mode |
| STO | STO | Store the program on Level 1 of the stack into the variable “pPARINSER” |

Now to take your program for a test ride. Enter in the values for the Example 3-2 and see whether the program gives the same results as before.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1000 | 1000 | 4:<br>3:<br>2:<br>1: | Enter value of R1 |
| ENTER | ENTER | 4:<br>3:<br>2:<br>1: 1.00E3 | Place the value of R1 into the stack |
| 2200 | 2200 | 4:<br>3:<br>2:<br>1: 2.20E3 | Enter value of R2 |
| ENTER | ENTER | 4:<br>3:<br>2: 1.00E3<br>1: 2.20E3 | Place the value of R2 into the stack |
| 3300 | 3300 | 4:<br>3:<br>2: 1.00E3<br>1: 2.20E3 | Enter the value of R3 |
| ENTER | ENTER | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 3.30E3 | Place the value of R3 into the stack |
| 4700 | 4700 | 4:<br>3: 1.00E3<br>2: 2.20E3<br>1: 3.30E3 | Enter value of R4 |
| ENTER | ENTER | 4: 1.00E3<br>3: 2.20E3<br>2: 3.30E3<br>1: 4.70E3 | Place the value of R4 into the stack |

| | | | |
|---|---|---|---|
| VAR | VAR | 4:<br>3:<br>2:<br>1: 2.63E3 | Calculate the value of R1 and R2 in parallel in series with R3 and R4 in parallel. |
| PPARI | pPARI | | |

It works! Now you can calculate the total resistance of similar series-parallel circuit in far less time than you could by working through the problem manually. For fun (and possibly to save time on some homework) you can try writing a similar program called "pSERINPAR" that calculates the total resistance of serial-parallel circuits similar to that shown below.



Just as an aside, you probably noticed that the calculator cannot display the entire name of "pPARINSER", although this is how it is actually stored. For now this is not an issue, as you (probably) don't have any other programs under the variable menu that share the same first characters. Later you will see how to avoid potential problems like this.

# 6. Working with Matrices

Two of the most powerful methods of circuit analysis are node-voltage and mesh current analysis. The main drawback of these methods is that they involve solving linear equations of the form

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$$

where **A** is an n x n matrix and **x** and **B** are column vectors. Not long ago students and engineers had to solve these equations by hand, which was a tedious and time-consuming process. Fortunately your HP-48 calculator is able to solve these equations with ease.

## 6.1 Some Background

As discussed in Principles of Electric Circuits, node-voltage and mesh current analysis yield a linear system of the form

$$\begin{array}{c} A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n = B_1 \\ A_{21}x_1 + A_{22}x_2 + \ldots + A_{2n}x_n = B_2 \\ . \\ . \\ . \\ A_{n1}x_1 + A_{n2}x_2 + \ldots + A_{nn}x_n = B_n \end{array}$$

where the values of $A_{jk}$ and $B_j$ are known. This system can be represented using matrices as

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$$

where

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1n} \\ A_{21} & A_{22} & \ldots & A_{2n} \\ \ldots & \ldots & & \ldots \\ A_{n1} & A_{n2} & \ldots & A_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}, \text{ and } \mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \\ \ldots \\ B_n \end{bmatrix}$$

Once the matrices **A** and **B** are known, finding **x** is a simple matter of dividing **B** by **A**.

## 6.2 A Moment on the Soapbox

Immediately after reading the last sentence of the previous section every instructor and student of linear algebra will no doubt leap to their feet and shriek in indignation that there is no such thing as matrix division. They will emphatically insist that dividing **B** by **A** has no meaning, and pull Massive Tomes of Ancient Mathematical Wisdom to support their claim. The only way to solve for B, they contend, is to use the time-honored method given by

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{B} \Rightarrow \mathbf{A}^{-1} \cdot (\mathbf{A} \cdot \mathbf{x}) = \mathbf{A}^{-1} \cdot \mathbf{B} \Rightarrow (\mathbf{A}^{-1} \cdot \mathbf{A}) \cdot \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{B} \Rightarrow \mathbf{I} \cdot \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{B} \Rightarrow \mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{B}$$

This means, they conclude triumphantly, that **x** can only be found by multiplying the inverse of **A** by **B**. Matrix division indeed! Harrumph, harrumph!

To such impeccable reasoning my only response can be to remove and clean my glasses while calmly replying, “What’s the point?”

This is not a flippant response. Recall that we are dealing with objects, and objects have defined methods for doing things. Someone at some time came up with the idea and notation of matrix multiplication to represent linear systems of equations in a convenient and compact form. What is so sacrilegious representing the *operation*

$$\mathbf{A^{-1} \cdot B}$$

with the *notation*

$$\mathbf{B / A}$$

and referring to it as matrix division? It gives exactly the same result as inverse multiplication because it is exactly the same thing as inverse multiplication. In addition, however, its notation is far more consistent with our ideas of working with multiplication and division. That really is the point of using objects. We can deal with mathematical operations consistently because each object understands what is meant by that operation. Furthermore, it knows how to obtain the correct result for that operation regardless of what it actually has to do to obtain that result.

This is exactly what the HP-48 has done. Just as you can add, subtract, multiply, and divide integers and floating point numbers, you can add, subtract, multiply, and divide any other object for which those operations are defined – and that includes matrices.

## 6.3 Using the Matrix Writer

The HP-48 offers the user two ways of entering matrices. One way is to enter the matrix from the command line, just as you have been doing with numbers and programs. The second and much better way is to use the Matrix Writer. This is a special editor built into the calculator that lets you enter the values of the matrix the way you would on paper. As an example, let's enter the matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

using the Matrix Writer. Once you know how to enter a 2 x 2 matrix, you'll know how to enter in any matrix.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MATRIX | ↰ MTRW | Start the Matrix Writer. The screen will change to a display that resembles a spreadsheet and start in the upper lefthand corner (Row 1, Column 1). The display shows the column numbers along the top and row numbers along the lefthand side. |
| 1 ENTER | 1 ENTER | Enter the value of $A_{11}$. |
| ▶ | | Cursor over to Row 1, Column2. The HP-48gII and HP-49g+ automatically moves the cursor over to the next column. |
| 2 ENTER | 2 ENTER | Enter the value of $A_{12}$. |

| | | |
|---|---|---|
| ▼ ◄ | ▼ ◄ ◄ | Cursor over to Row 2, Column 1. |
| 3 ENTER | 3 ENTER | Enter the value of $A_{21}$. Note that once ENTER is pressed the calculator automatically fills Row 2, Column 2 with the value 0.00E0. This is because it knows that since Row 1 has two columns that Row 2 must also have two columns. |
| ► | | Cursor over to Row 2, Column 2. |
| 4 ENTER | 4 ENTER | Enter the value of $A_{22}$. |
| ENTER | ENTER | Enter the matrix into the stack. |

At this point you will see one of two things in Level 1 of the stack. If you have an HP-48GX you will see the matrix in the form that you would have used to enter it on the command line, namely

```
[ [ 1.00E0  2.00E0 ]
  [ 3.00E0  4.00E0 ] ]
```

If you are using an HP-48gII or HP-49g+ you will see the matrix displayed the way you'd expect a matrix to look, namely

$$\begin{bmatrix} 1.00E0 & 2.00E0 \\ 3.00E0 & 4.00E0 \end{bmatrix}$$

In either case the calculator understands that the object in Level 1 is a matrix and how to deal with it when performing calculations.

Now that you've entered a 2 x 2 matrix you can enter any size matrix using the same basic procedure. You use the cursor keys to move to the location of each element in the matrix and enter the values into the matrix with the ENTER key. Once you are finished ENTER will enter the matrix into the calculator and exit the Matrix Writer so that you can proceed with your matrix calculations.

The HP-48 actually provides a couple ways to solve equations using matrices. One is to enter the matrices into the stack just as with any other calculation. The second way is to use the Linear Equation Solver. We'll take a look at both ways.

## 6.4 Solving Matrix Equations Using the Stack

**Example 6-1:** Solve the linear system shown below.

$$\begin{bmatrix} 8 & 4 & 1 \\ 2 & -5 & 6 \\ 3 & 3 & -2 \end{bmatrix} , \begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ -5 \end{bmatrix}$$

**Solution:** Use the Matrix Writer to enter the 1 x 3 and 3 x 3 matrices into the stack and perform the division.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MATRIX | ↰ MTRW | Enter the Matrix Writer. |
| 7 ENTER | 7 ENTER | Enter the value of $B_1$. |
| ▼ | ▼ ◀ | Cursor over to Row 2, Column1. |
| 3 ENTER | 3 ENTER | Enter the value of $B_2$. |
| ▼ | | Cursor over to Row 3, Column 1. Note that the HP-48gII and HP-9g+ know where the next value goes. |
| -5 ENTER | -5 ENTER | Enter the value of $B_3$. |
| ENTER | ENTER | Enter **B** into the Stack. |
| ↱ MATRIX | ↰ MTRW | Enter the Matrix Writer. |
| 8 ENTER | 8 ENTER | Enter the value of $A_{11}$. |
| ▶ | | Cursor over to Row 1, Column 2 |
| 4 ENTER | 4 ENTER | Enter the value of $A_{22}$. |
| ▶ | | Cursor over to Row 1, Column 3 |

| | | |
|---|---|---|
| 1 ENTER | 1 ENTER | Enter the value of $A_{13}$. |
| ▼ ◀ ◀ | ▼ ◀ ◀ ◀ | Cursor over to Row 2, Column 1 |
| 2 ENTER | 2 ENTER | Enter the value of $A_{21}$. |
| ▶ | | Cursor over to Row 2, Column 2 |
| -5 ENTER | -5 ENTER | Enter the value of $A_{22}$. |
| ▶ | | Cursor over to Row 2, Column 3. |
| 6 ENTER | 6 ENTER | Enter the value of $A_{23}$. |
| ▼ ◀ ◀ | | Cursor over to Row 3, Column 1 |
| 3 ENTER | 3 ENTER | Enter the value of $A_{31}$. |
| ▶ | | Cursor over to Row 3, Column 2. |
| 3 ENTER | 3 ENTER | Enter the value of $A_{32}$. |
| ▶ | | Cursor over to Row 3, Column 3. |

| | | |
|---|---|---|
| -2 ENTER | -2 ENTER | Enter the value of $A_{33}$. |
| ENTER | ENTER | Enter **A** into the stack. |

At this point **A** is in Level 1 of the stack and **B** is in Level 2. Now fly in the face of conventional mathematics wisdom and press the "÷" key to try dividing **B** by **A**. The result is

$$\mathbf{x} = \begin{bmatrix} -3.73E0 \\ 7.27E0 \\ 7.80E0 \end{bmatrix}$$

But is this really the same result you would get if you followed the "correct" procedure of multiplying the inverse of **A** by **B**?

Let's find out. Re-enter the matrices into the stack using the above procedure. This time however, execute the following keystrokes:

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1/X | 1/X | 4:<br>3:<br>2: **B**<br>1: $\mathbf{A}^{-1}$ | Find the inverse of A. |
| ► | ► | 4:<br>3:<br>2: $\mathbf{A}^{-1}$<br>1: **B** | Swap the values in Levels 1 and 2 |
| x | x | 4:<br>3:<br>2:<br>1: **x** | Find the value of $\mathbf{A}^{-1} \cdot \mathbf{B}$ |

Once again, the result is

$$\mathbf{x} = \begin{bmatrix} -3.73E0 \\ 7.27E0 \\ 7.80E0 \end{bmatrix}$$

## 6.5 Using the Linear System Solver

The HP-48 has a special feature called the Solver which, as its name suggests, is useful for solving for the unknown variable in some equation. The Solver is actually a group of tools you can use to solve linear equations, polynomials, differential equations, and other common calculation problems. In this case we will use the Linear System Solver to find the solution to our system of linear equations.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ SOLVE | ↱ NUM.SLV | Access the Solver menus. |

| | | |
|---|---|---|
| ▲ ▲ | 4 | Select the Linear System Solver (identified by the label “Solve lin sys…”) |
| ENTER | ENTER | Enter the Linear System Solver. The Linear System Solver defaults to **A** as the first value to enter. The HP-48GX display will show “ENTER COEFFICIENTS MATRIX A”. The HP-48gII and HP-49g+ will show “Enter coefficients matrix A”. |
| ↱ MATRIX | ↰ MTRW | Enter the Matrix Writer to enter the coefficients of Matrix **A**. |
| ... | ... | Enter the coefficients of **A** using the steps shown in Section 6.4. When you are finished and press ENTER you will be returned to the Linear System Solver. The box for **A** will show the command line form of **A**. |
| ▼ ▼ | ▼ ▼ | Cursor down to enter the value of **B**. The HP-48GX display will show “ENTER CONSTANTS OR PRESS SOLVE”. The HP-48gII and HP-49g+ displays will show “Enter constants or press SOLVE”. |
| ↱ MATRIX | ↰ MTRW | Enter the Matrix Writer to enter the coefficients of Matrix **B**. |
| ... | ... | Enter the coefficients of **B** using the steps shown in Section 6.4. When you are finished and press ENTER you will be returned to the Linear System Solver. The box for **B** will show the command line form of **B**. |
| ▼ | ▼ | Cursor down to the box for **x**. The HP-48GX display will show “ENTER SOLUTIONS OR PRESS SOLVE”. The HP-48gII and HP-49g+ displays will show “Enter solutions or press SOLVE”. |
| SOLVE | SOLVE | Press SOLVE. After a short pause the calculator will display the solution for **x** in the box for **x**. |
| CANCEL | CANCEL | Exit the Linear System Solver. Level 1 will contain the message “Solutons:” along with the value of **x**. |

It appears that the Solver is a useful way to perform matrix calculations, but you may be wondering why exactly it is called a “solver” and not a “calculator”. The answer is that the Solver will solve for any unknown in the equation, provided all the other values are specified. For the equation

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{B}$$

you can solve for any value provided you specify the other two values. If you had entered **A** and **x**, for example, you could have had the Solver determine the value of **B**. Similarly, if you specified **B** and **x** the Solver could solve for the value of **A**. Unlike most programs which are written to solve for just one variable in an equation, the Solver can solve for any of them. This is quite useful in many applications. In Finance, for example, you may want to solve for any number of things, such as the present value of a loan, the total number of loan payments, the future value of a mortgage, the value of payments in an annuity, or the IRR (intrinsic rate of return, or “interest rate”) of a particular investment. Usually several programs would be needed to perform those calculations. In the HP-48 the Financial Solver will let you find any of those values.

# 7. Working with Complex Numbers

Up to now all the calculations in this tutorial have dealt with real numbers. A real number is a number that, logically enough, does not have an imaginary component. Complex numbers are numbers that can be expressed in the form

$$a + b\mathrm{i}$$

where i is the square root of -1. *a* is the real component and *b*i is the imaginary component. In electronics complex numbers often use j rather than i to represent the square root of -1, as i is often used to represent current. In addition, complex numbers are often written as

$$a + \mathrm{j}b$$

to clearly identify the value of the imaginary term.

## 7.1 Forms of Complex Numbers

Complex numbers can be represented in one of two forms.

The first form, which has already been shown, if referred to as rectangular form. It is called this because the values *a* and *b* can be thought of as the coordinates in a rectangular coordinate system in which the x-axis represents the real component and the y-axis represents the imaginary component. 3 - j4, for example, represents a complex number that is 3 units along the positive x-axis and 4 units along the negative y-axis. A complex number with *a* = 0 is purely imaginary. Similarly, a number with *b* = 0 is purely real.

The second form is referred to as polar form. Tthis form represents complex numbers as

$$r \angle \theta$$

This form represents a complex value, not by distances along the x- and y-axes, but by a distance *r* from the origin and an angle of rotation *θ* (by convention counterclockwise is positive) from the real axis. A complex number in which *θ* = ±90º is purely imaginary while a complex number with *θ* = 0º or *θ* = 180º is purely real.

## 7.2 Converting Complex Number Forms

From trigonometry it is easy to determine how *a* and *b* correspond to *r* and *θ*. The conversion equations are:

$$r = \sqrt{a^2 + b^2}$$
$$\theta = \tan^{-1}(b / a)$$
$$a = r \cos \theta$$
$$b = r \sin \theta$$

In the past these calculations had to be performed manually, whether by calculator or some other means. More modern scientific calculators have built-in polar-to-rectangular and rectangular-to-polar conversion functions (usually represented by P→R and R→P, respectively) to simplify these conversions. These conversions were useful to know for two reasons. The first is that it is easier to perform complex addition and subtraction using rectangular form, while it is simpler to perform complex multiplication and addition using polar form. Another reason is that calculators that supported complex numbers typically required the numbers to be in rectangular form. Polar expressions had to be converted to rectangular form during calculations, and then back again once the calculation was complete.

If you search you're your calculator you will find that the HP-48 does not have these conversion functions. There is a very simple explanation for this. The HP-48 does not need them because (everyone together) the HP-48 works with objects.

## 7.3 The Complex Number Object

You have already been introduced to number objects, program objects, and matrix objects. You will now learn about the complex number object. The complex number object works with what the complex number *is*, not what it looks like. Complex numbers are represented by ordered pairs in one of two forms:

(a, b) for rectangular form
(r, ∠θ) for polar form

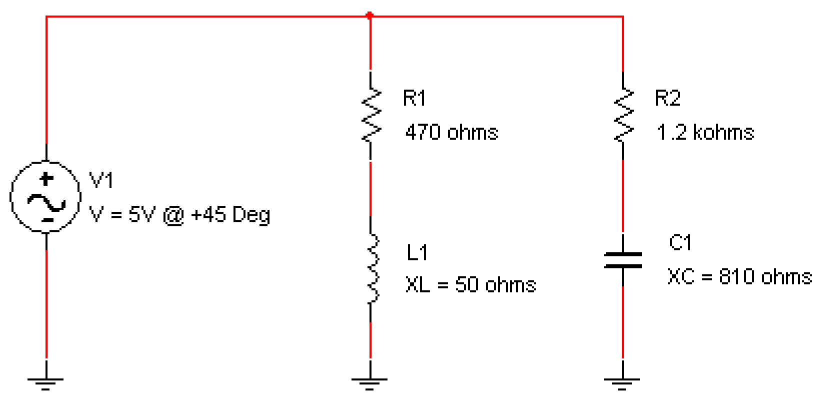
The HP-48 displays complex numbers based upon the coordinate system mode that is set. To place the calculator in rectangular mode, enter the following keystrokes:

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MODES | MODE | Access the Calculator Modes screen. |
| ▼ ▼ | ▲ ▲ | Cursor to the "Coord System" box. The HP-48GX display will show "CHOOSE COORDINATE SYSTEM". The HP-48gII and HP-49g+ will show "Choose coordinate system". |
| α R | ALPHA R | Select Rectangular mode. The "Coord System" box will show "Rectangular". |
| OK | OK | Set the calculator for rectangular mode. The HP-48gII and HP-49g+ will show "XYZ" at the top of the display. The HP-48GX will show nothing. |

To place the calculator in polar mode, enter the following keystrokes:

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MODES | MODE | Access the Calculator Modes screen. |
| ▼ ▼ | ▲ ▲ | Cursor to the "Coord System" box. The HP-48GX display will show "CHOOSE COORDINATE SYSTEM". The HP-48gII and HP-49g+ will show "Choose coordinate system". |
| α P | ALPHA P | Select polar mode. The "Coord System" box will show "Polar". |
| OK | OK | Set the calculator for polar mode. The calculator will show R∠Z at the top of the display |

**Example 7-1:** Set the calculator for rectangular mode and calculate the total impedance of the circuit shown.



**Solution:** Calculate the total resistance in rectangular mode using complex number objects.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 470<br>ENTER | 470<br>ENTER | 4:<br>3:<br>2:<br>1: 470.E0 | Enter the value of R1. You do not need to use the complex number notation, as a number object by itself is considered to be purely real. |
| ↰<br>( ) | ↰<br>( ) | 4:<br>3:<br>2:<br>1: 470.E0 | Specify a complex number object. |
| 0<br>SPC<br>50 | 0<br>SPC<br>50 | 4:<br>3:<br>2:<br>1: 470.E0 | Specify the reactance of the inductor, 0 + j50. You can use either “SPC” or “,” to separate the real and imaginary values within the parenthesis. |
| + | + | 4:<br>3:<br>2:<br>1: (470.E0, 50.0E0) | Calculate the impedance of the first branch. |
| 1/X | 1/X | 4:<br>3:<br>2:<br>1: (2.10E-3, -224.E-6) | Calculate the admittance of the first branch. |
| 1200<br>ENTER | 1200<br>ENTER | 4:<br>3:<br>2: (2.10E-3, -224.E-6)<br>1: 1.20E3 | Enter the value of R2. |
| ↰<br>( ) | ↰<br>( ) | 4:<br>3:<br>2: (2.10E-3, -224E-6)<br>1: 1.20E3 | Specify a complex number object. |

| | | | |
|---|---|---|---|
| 0<br>SPC | 0<br>SPC | 4:<br>3:<br>2: (2.10E-3, -224E-6)<br>1: 1.20E3 | Specify the reactance of the capacitor, 0 – j810. |
| -810<br>+ | -810<br>+ | 4:<br>3:<br>2: (2.10E-3, -224E-6)<br>1: (1.20E3, -810.E0) | Calculate the impedance of the second branch. |
| 1/X | 1/X | 4:<br>3:<br>2: (2.10E-3, -224E-6)<br>1: (572.E-6, 386.E-6) | Calculate the admittance of the second branch. |
| + | + | 4:<br>3:<br>2:<br>1: (268.E-3, 163.E-6) | Calculate the total admittance of the circuit. |
| 1/X | 1/X | 4:<br>3:<br>2:<br>1: (372.E0, -22.6E0) | Calculate the total impedance of the circuit. |

Note that other than having to specify a real and imaginary component for some of the components, the procedure is identical as that for working with purely real numbers.

**Example 7-2:** Calculate the total current for the circuit of Example 7-1. Show the answer in both rectangular and polar form.

**Solution:** You have already calculated the total impedance of the circuit, so simply it into the source voltage. Note that you do NOT have to convert the source voltage into rectangular form.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| ↰<br>( ) | ↰<br>( ) | 4:<br>3:<br>2:<br>1: (372.E0, -22.6E0) | Specify a complex number object. |
| 5<br>SPC<br>↱<br>∠<br>45 | 5<br>SPC<br>ALPHA<br>↱<br>6<br>45 | 4:<br>3:<br>2:<br>1: (372.E0, -22.6E0) | Enter the supply voltage in polar form. To specify the "∠" symbol in the HP-48gII and HP-49g+, use the character map shortcut. |
| ENTER | ENTER | 4:<br>3:<br>2: (372.E0, -22.6E0)<br>1: (3.54E0, 3.54E0) | Enter the supply voltage into the stack. Note that the calculator changes to rectangular mode, which is the current display mode. |
| ▶ | ▶ | 4:<br>3:<br>2: (3.54E0, 3.54E0)<br>1: (372.E0, -22.6E0) | Swap Levels 1 and 2 of the stack. |

| | | | |
|---|---|---|---|
| ÷ | ÷ | 4:<br>3:<br>2:<br>1: (8.89E-3, 10.0E-3) | Calculate the total current. The calculator shows the answer in rectangular mode. |
| ↱<br>MODES | MODE<br> | 4:<br>3:<br>2:<br>1: (8.89E-3, 10.0E-3) | Enter the Calcualator Modes screen. |
| ▼<br>▼ | ▲<br>▲ | 4:<br>3:<br>2:<br>1: (8.89E-3, 10.0E-3) | Select the coordinate system box. |
| α<br>P | ALPHA<br>P | 4:<br>3:<br>2:<br>1: (8.89E-3, 10.0E-3) | Select polar mode. |
| OK | OK | 4:<br>3:<br>2:<br>1: (13.4E-3, ∠48.5E0) | Set the calculator for polar mode. Any complex number object in the stack will now be shown in polar form. |

As you can see, you can enter complex numbers in either rectangular or polar form regardless of the display mode of the calculator, just as you could enter numbers as integers even though the calculator was set for engineering mode. When you work with impedances and complex values you can set the display to the mode required for the final answer and avoid converting between rectangular and polar modes for the intermediate results. Typically you will show voltages and currents in polar form, as magnitude and phase are easy to check with measuring equipment. Rectangular form is usually used when calculating impedances, as it allows you to more easily assess the resistive and reactive components, and for power when you must determine the values of true and apparent power.

# 8. Managing Memory

Previous sections showed you that you could store objects (such as programs) into variables so that you could recall them later as you needed them. While this is quite useful, it is possible that you could end up with dozens or hundreds of variables stored on your HP-48. This presents two very real problems:

1. You have so many variables that you can't keep track of them all.
2. You run out of suitable variable names to use.

Although variable names can be up to 127 characters long only the first few characters are displayed in the labels at the bottom of the calculator display. In a previous section I suggested that you prefix variable names with a letter indicating the type of object ("p" fr programs, "m" for matrices, etc.) to help distinguish similar variables, but at some point you will probably have some variables that have the same names. Memory management allows you to organize your variables to help avoid potential confusion.

## 8.1 Memory Organization

Just as your computer allows you to set up folders or directories on disks to store related files, your HP-48 allows you to set up directories to organize your calculator memory. Up to now you have been working in the HOME, or root, folder. If you look at the top of your calculator display you will see { HOME } just above the memory stack. This is the working directory. The working directory is the directory into which variables are placed when you create them. When you attempt to access a variable, the HP-48 looks first in the working directory. If it cannot find it in the working directory it will search the directories above the working directory in order up to the HOME directory until it finds the variable or determines that the variable does not exist and issues an error message.

You have already learned how to store objects in variables. Now you will learn how to create directories, maneuver through the directory structure, and manipulate variables within it.

## 8.2 Creating Directories

**Example 8-1:** Create a directory called PEC8 under the HOME directory.

**Solution:** To create a new directory, enter the keypresses shown.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ | ↰ | Access the Memory Management screen for the HP-48GX and the File Manager screen in the HP-48gII and HP-49g+. |
| MEMORY | FILES | |
| | OK | Select OK to access the Memory Management screen. |
| | NXT | Access the second screen of Memory Management functions. |
| NEW | NEW | Select the NEW tab at the bottom of the screen to acess the New Variable screen. The display will show "ENTER NEW OBJECT" for the HP-48GX and "Enter new object" for the HP-48gII and HP-49g+. |

| | | |
|---|---|---|
| ▼ | ▼ | Cursor down to the NAME: box. The display will show “ENTER VARIABLE NAME” for the HP-48GX and “Enter variable name” for the HP-48gII and HP-49g+. |
| α | ALPHA | Lock the keyboard in alphabetic mode. |
| α | ALPHA | |
| PEC8 | PEC8 | Enter the name PEC8 in the NAME: box. |
| α | ALPHA | Turn off the keyboard alphabetic mode. |
| ▼ | ▼ | Cursor down to the DIRECTORY selection box. The display will show “CREATE A NEW DIRECTORY?” for the HP-48GX and “Create a new directory?” for the HP-48gII and HP-49g+. |
| ✓CHK | ✓CHK | Select the ✓CHK button to specify creation of a new directory rather than a variable. |
| OK | OK | Select OK to confirm creation of the directory. This will take you back to the Memory Management screen. At the top of the list of objects in the screen is your new PEC8 directory. |
| CANCEL | CANCEL | Select cancel to exit the Memory Management screen |

While performing this exercise you probably noticed something interesting, namely, that the HOME directory wasn’t empty. Some of the objects were things that you created earlier and probably recognized, such as the pPARINSER program. Other objects, however, are created by the calculator itself when you use certain features such as the Solver. This underscores the need to keep your variables safely tucked away somewhere else. There is no guarantee that you and the HP-48GX aren’t trying to put something with the same name in the HOME directory (or for the HP-48gII or HP-49g+ in the CASDIR directory).

## 8.3 Copying Objects

One useful feature of the HP-48 is the ability to copy objects. This is useful for when you want to make backups of important objects or when you want to modify an object such as a program or large matrix without destroying the original.

Note that with the HP-48GX COPY will allow you to specify the name or the copy, or a new location (path), but not both at the same time. The reasoning behind this is simple. You cannot have two objects with the same name in the same place, so if you want to create it in the same directory you must specify a new name. Conversely, if you’re making a copy of a variable with the same name, you need to put it in another place (directory). With the HP-48gII and HP-49g+ you have no choice: you *must* copy the file to another directory. If you wish to create a copy with a new name in the same directory, you must copy the object to another directory, rename it, and then move it back.

**Example 8-2:** Create a copy of the variable pPARINSER in the PEC8 directory.

**Solution:** To create the copy of pPARINSER in the PEC8 directory, enter the keypresses shown.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MEMORY | ↰ FILES | Access the Memory Management screen for the HP-48GX and the File Manager screen in the HP-48gII and HP-49g+. |
| | ► | Press the ► key to display the contents of the HOME directory in the HP-48gII and HP-49g+. |
| ▼ | ▼ | Cursor down to the pPARINSER variable. Depending upon the contents of your HOME directory, you may need to press the ▼ key more than once. The cursor will wrap from the bottom of the list back to the top if you cursor past the last item accidentally. |
| COPY | COPY | Select the COPY tab at the bottom of the screen. The HP-48GX will show the Copy Variables screen. The NAME: box will display the pPARINSER name and default to the COPY TO: box with the message “ENTER VAR NAME OR VARIABLE PATH”. The HP-48gII and HP-49g+ will display the Chooser screen with the message “PICK DESTINATION” at the top. |
| CHOOSE | | Start the Chooser in the HP-48GX. The chooser will show the directory structure of the calculator and highlight the working directory. |
| ▼ OK | ▼ OK | Cursor down to the PEC8 directory. The MOVE TO: box will show { HOME PEC8 } as the new path for the variable in the HP-48GX. The PEC8 directory will be highlighted in the HP-48gII and HP-49g+ File Manager. Pressing OK will complete the copy. |
| CHOOSE | | You will return to the HOME directory. As you can see, the variable pPARINSER is still there (it would not have been had you used MOVE rather than COPY. Start the Chooser in the HP-48GX. You will still be in the File Manager in the HP-48gII and HP-49g+. |
| ▼ OK | ▼ ► | Cursor down to the PEC8 directory and display the contents. As you can see, the variable pPARINSER has been copied and is now in the PEC8 directory as well as the HOME directory. |
| CANCEL | CANCEL | Exit the Memory Management screen in the HP-48-GX and the File Manager screen in the HP-48gII and HP-49g+. |

If you compare the above procedures for the HP-48GX with those for the HP-48gII and HP-49g+ you can see that the memory management capabilities of the newer HP-48gII and HP-49g+ File Manager are something of an improvement over the procedures you must use for the HP-48GX.

## 8.4 Renaming Objects

Something that may have occurred to you during this exercise is how you can rename variables in the HP-48GX. Rather than specifying a path in which to move the variable, you simply enter the new variable name. The HP-48 will create a variable with the new name, move the contents of the old variable into it, and then delete the old variable for you. Although you can use this method with the HP-48gII and HP-49g+, these calculators offer a more straightforward method.

**Exercise 8-3:** Rename the variable pPARINSER in the PEC8 directory to pPINS.

**Solution:** To rename pPARINSER in the PEC8 directory to pPINS, enter the keypresses shown.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ⮣ MEMORY | ⮢ FILES | Access the Memory Management screen for the HP-48GX and the File Manager screen in the HP-48gII and HP-49g+. |
| CHOOSE ▼ | ▼ | Start the Chooser in the HP-48GX. Cursor down to the PEC8 directory. |
| OK | ► | Display the contents of the PEC8 directory. |
| ▼ | ▼ | Cursor down to the pPARINSER variable. Depending upon the contents of your HOME directory, you may need to press the ▼ key more than once. The cursor will wrap from the bottom of the list back to the top if you cursor past the last item accidentally. |
| MOVE | NXT RENAM | Select the MOVE tab at the bottom of the HP-48GX screen and the RENAM tab at the bottom of the HP-48gII and HP-49g+ screen. The HP-48gII and HP-49g+ will show the old name pPARINSER at the bottom of the screen. |
| α α | ALPHA ALPHA | Lock the keyboard in alphabetic mode |
| pPINS | pPINS | Use the keyboard to enter the new variable name pPINS. Remember to use the ⮢ key to enter a lower case letter. For the HP-48gII and HP-49g+ use the cursor keys (▲◄▼►) and ⇦ (backspace) keys to modify the name. |
| α | ALPHA | Turn off the keyboard alphabetic mode. |
| OK OK | ENTER | Confirm the name change and return to the Memory Manager in the HP-48GX and File Manager in the HP-48gII and HP-49g+. As you can see, the variable has been renamed to pPINS. |
| CANCEL | CANCEL | Exit the Memory Management screen in the HP-48-GX and the File Manager screen in the HP-48gII and HP-49g+. |

## 8.5 Moving Objects

The procedures for moving objects in the HP-48 is similar to copying them except that moving variables deletes them from the directory in which they were originally located. Note that the procedure is nearly identical to copying except for selecting MOVE instead of COPY in the Memory Management screen.

**Example 8-4:** Move the variable pPARINSER from the HOME directory to the PEC8 directory.

**Solution:** To move pPARINSER from the HOME directory to the PEC8 directory, enter the keypresses shown.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ MEMORY | ↰ FILES | Access the Memory Management screen for the HP-48GX and the File Manager screen in the HP-48gII and HP-49g+. |
| | ► | Press the ► key to display the contents of the HOME directory in the HP-48gII and HP-49g+. |
| ▼ | ▼ | Cursor down to the pPARINSER variable. Depending upon the contents of your HOME directory, you may need to press the ▼ key more than once. The cursor will wrap from the bottom of the list back to the top if you cursor past the last item accidentally. |
| MOVE | MOVE | Select the MOVE tab at the bottom of the screen. The HP-48GX will show the Move Variables screen. The NAME: box will display the pPARINSER name and default to the MOVE TO: box with the message "ENTER VAR NAME OR VARIABLE PATH". The HP-48gII and HP-49g+ will display the Chooser screen with the message "PICK DESTINATION" at the top. |
| CHOOSE | | Start the Chooser in the HP-48GX. The chooser will show the directory structure of the calculator and highlight the working directory. |
| ▼ OK | ▼ OK | Cursor down to the PEC8 directory. The MOVE TO: box will show { HOME PEC8 } as the new path for the variable in the HP-48GX. The PEC8 directory will be highlighted in the HP-48gII and HP-49g+ File Manager. Pressing OK will complete the move. |
| CHOOSE | | You will return to the HOME directory. As you can see, the variable pPARINSER is no longer there (it would have been had you used COPY rather than MOVE). Start the Chooser in the HP-48GX. You will still be in the File Manager in the HP-48gII and HP-49g+. |
| ▼ OK | ▼ ► | Cursor down to the PEC8 directory and display the contents. As you can see, the variable pPARINSER has been moved and is now in the PEC8 directory. |
| CANCEL | CANCEL | Exit the Memory Management screen in the HP-48-GX and the File Manager screen in the HP-48gII and HP-49g+. |

## 8.6 Deleting Variables

At this point the PEC8 directory will contain two variables, called pPARINSER and pPINs. Eventually you will decide that you want to remove some variables from your calculator, either because (as in this case) they are duplicates of other objects, you no longer need them, you can't remember what they are for, or you need more memory for some massive project on your calculator. Whatever the reason, the HP-48 allows you to do this with its memory management PURGE function.

**Example 8-5:** Delete the pPARINSER object in the PEC8 directory.

**Solution:** To delete the pPARINSER object in the PEC8 directory, enter the keypresses shown.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↱ | ↰ | Access the Memory Management screen for the HP-48GX and the File Manager screen in the HP-48gII and HP-49g+. |
| MEMORY | FILES | |
| CHOOSE | ► | Start the Chooser in the HP-48GX and cursor down to the PEC8 directory. Press the ► key to display the contents of the HOME directory in the HP-48gII and HP-49g+ and cursor down to the PEC8 directory. You may need to cursor down more than once. |
| ▼ | ▼ | |
| OK | ► | Enter the PEC8 directory. |
| ▼ | ▼ | Cursor down to the pPARINSER variable. Depending upon the contents of your PEC8 directory, you may need to press the ▼ key more than once. The cursor will wrap from the bottom of the list back to the top if you accidentally cursor past the last item. |
| NXT | NXT | Advance to the next screen and select the PURG tab at the bottom of the HP-48GX screen or the PURGE tab at the bottom of the HP-48gII or HP-49g+ screen. |
| PURG | PURGE | |
| | YES | The HP-48GX will immediately delete the selected object. The HP-48gII and HP-49g+ will ask "'pPARINSER' Are you sure?". Select the YES tab at the bottom of the screen to proceed with the deletion. pPARINSER will be removed from PEC8. |
| CANCEL | CANCEL | Exit the Memory Management screen in the HP-48-GX and the File Manager screen in the HP-48gII and HP-49g+. |

As indicated in the above example, the HP-48GX will delete the object immediately after you select PURG without giving you any way to undo the deletion. The HP-48gII and HP-49g+, on the other hand, are more considerate and will give you a chance to reconsider and cancel the operation. Because deleting objects is a irreversible (one-way) process always make sure to back up (copy) objects that are important to you and be very careful when using this operation. The HP-48 also allows you to serially upload information from your calculator to your computer for just that reason, but that topic is beyond the scope of this limited tutorial and will not be covered. Consult your HP-48 documentation for details on communications between your HP-48 and PC.

## 8.7 A Final Note

Although the above sections used variables for the examples showing how to copy, rename, move, and delete objects you can use these same memory operations on entire directories. This makes it easy to work with groups of variables in your computer at the same time and emphasizes the idea of organizing your data in directories. At the same time it underscores how dangerous deleting objects can be. One wrong keypress and your entire collection of prized (and possibly fairly lengthy) programs could be permanently erased.

# 9. More About Programming

By now you should be familiar with a number of your calculator's features and how to use them. In this section you will learn more about programming, which is a topic that this tutorial has already covered. Specifically you will learn about using local variables and structures, two features that make programming more efficient.

## 9.1 What Is a Local Variable?

You know that a variable is a storage place for an object. Up to now the variables you have used have been global variables. This means that you could access these variables at any time provided only that they were in the working path, that is, the working directory or some directory directly above it in the directory structure. If your working path is {HOME PEC8 MYPROGS} then you can access any variable in HOME, PEC8, or MYPROGS. Variables in the root directory HOME are accessible at any time, as HOME is the starting directory for all subdirectories and therefore always in the working path. Another feature of global variables is that they will exist until you specifically delete them (or until your batteries go dead and information in non-volatile memory is lost).

Local variables differ from global variables in two ways. First, they can only be accessed from inside the structure in which they are created. Second, local variables cease to exist once the program ends. If you are familiar with conventional programming this will sound very similar to the local variables declared inside program subroutines.

## 9.2 What Is a Structure?

In HP-48 programming a structure is an object that consists of two distinct parts. The first part is a local variable declaration that creates and initializes the local variables. The second part is a program section that contains instructions that operate on the local variables. Once again, if you are familiar with conventional programming this will sound very similar to many program subroutines. Typically a subroutine will contain code that declares the variables used within the subroutine, followed by the instruction code that uses them.

## 9.3 Why Use Local Variables?

In the previous section on programming you saw how you could use stack operation to manipulate and process data. This is quite useful, especially for manual calculations, but you may have wondered if there was some easier way to deal with data in programs. As you may have guessed, there are.

One way, which probably occurred to you when you first learned about variables, is to first store the data in variables and access the variables inside the program. This will work, but you have to be sure that the variables are in the working path when you run the program. Also, you have to remember to delete the variables afterwards or you run the risk of having directories filled with dozens or hundreds of variables that you don't really need.

The second way is to use structures. Because structures use local variables the program will always be able to access the data and the local variables will be automatically purged once the program has finished.

## 9.4 Your First Structure

To demonstrate structures, let's look at the venerable problem of adding two numbers.

**Example 9-1**: Create a structure that will add two numbers in the stack.

**Solution**: Make PEC8 your working directory and enter the following structure. Not all the keypresses for entering the variable name are shown, as you should be familiar by now with how to enter text into the calculator.

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↰ | ↱ | Specify a program object type. A structure object is a type of program object. |
| << >> | << >> | |
| → | → | Initiate the local variable declaration. |
| α | ALPHA | Lock the keyboard in alphabetic mode. |
| α | ALPHA | |
| X | X | Create the first local variable, called X. You can actually name this anything you want. The data on Level 1 will go into the first local variable. |
| SPC | SPC | |
| Y | Y | Create the second local variable, called Y. The data on Level 2 will go into the second local variable. The space is not necessary but makes the code more legible. |
| SPC | SPC | |
| ↰ | ↱ | Specify the code section of the structure. |
| << >> | << >> | |
| X | X | Recall X to the stack. The space is necessary to separate X from the next variable. |
| SPC | SPC | |
| Y | Y | Recall Y to the stack. Note that this will place X on Level 2 and Y on Level 1 of the stack. |
| SPC | SPC | |
| + | + | Add the numbers on Levels 1 and 2. The result will be automatically placed in Level 1 of the stack. |
| ENTER | ENTER | Place the structure on the stack. Note that you do not need to move the cursor outside the double set of double angle brackets. Note that this will take the keyboard out of alphabetic mode. |
| α | ALPHA | Lock the keyboard in alphabetic mode. |
| α | ALPHA | |
| pADD | pADD | Create a variable name for the structure. Remember to use the “↰” shift key to enter the lower-case ‘p’. |

| | | |
|---|---|---|
| α | ALPHA | Turn off the keyboard alphabetic mode. |
| STO | STO▶ | Store the structure in the variable pADD. |

You have now created your first structure, called pADD. When you access the structure it will add the objects in Levels 1 and 2 of the stack. Let's test it out and see if it works.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 1 ENTER | 1 ENTER | 4:<br>3:<br>2:<br>1: 1.00E0 | Enter the first number on the stack. |
| 2 ENTER | 2 ENTER | 4:<br>3:<br>2: 1.00E0<br>1: 2.00E0 | Enter the second number on the stack. |
| VAR | VAR | 4:<br>3:<br>2: 1.00E0<br>1: 2.00E0 | Access pADD in the working directory. Depending upon the actual contents you may need to use NXT to see all lthe entries. |
| pADD | pADD | 4:<br>3:<br>2:<br>1: 3.00E0 | Execute pADD. |

As you can see, pADD does add the first two numbers in the memory stack. If you did happen to use the NXT key to check all the variables in the working directory, you should have noticed that there was no X or Y variable shown. This is because local variables in a structure only exist when that structure is executed. Furthermore, they only exist for that structure.

## 9.5 Program Notation

When you write your own programs you will probably want to have some way of representing the code so that you can design the program on paper before you enter it into the calculator. There is no "official" way of doing this, but one way of keeping your code uncluttered and easy to understand is to use a structured programming style. In this style functional blocks are kept together and nested levels are indented to make them easier to identify. This is actually very similar to how the HP-48 represents the code. For the above program, structured programming style could represent it as this:

```
"pADD"
<<
    → X Y                 ; Define local variables X and Y
    <<
        X Y +             ; Add local variables X and Y
    >>
>>
```

This above representation of pADD is trivial because the program itself is trivial (you could do the same thing by pressing the "+" key) and structured programming style really doesn't gain us much. For the following somewhat more complex program, however, structured programming style helps to make things easier to understand.

```
"pY2D"
<<
    → R1  R2  R3                      ; Declare local variables R1, R2, R3
    <<
        R1  R2  *
        R1  R3  *
        R2  R3  *  +  +               ; Calculate (R1*R2 + R1*R3 + R2*R3)
        <<                            : Create a substructure with local variable YNUM
            → YNUM                    ; Store (R1*R2 + R1*R3 + R2*R3) in YNUM
            <<
                YNUM  R1  /           ; Calculate (R1*R2 + R1*R3 + R2*R3) / R1
                YNUM  R3  /           : Calculate (R1*R2 + R1*R3 + R2*R3) / R3
                YNUM  R2  /           ; Calculate (R1*R2 + R1*R3 + R2*R4) / R2
            >>
         >>                           ; End of substructure
         EVAL                         ; See below for comments
    >>
>>                                    ; End of structure pY2D
```

The following section discusses how this structure works.

## 9.6 Analysis of pY2D

As you may have guessed from the equations in the comments, the above structure performs a Wye-Delta conversion. For anyone who ever had to calculate three or four such conversions a program that will perform this conversion automatically is definitely useful. Before actually entering the structure, however, you should first make sure you understand what is happening. The above structure incorporates some features that are both interesting and instructive.

One thing that stands out is the presence of a structure (or substructure) within the main structure. In the Wye-Delta conversion another thing that stands out is that the numerator that each delta resistance uses in the equation is identical. For this reason it makes sense to calculate the numerator once and have it available for the subsequent calculations. By using a substructure you can store the calculated numerator in a local variable (called YNUM) and have subsequent calculations accessed it. It is true that the numerator could have been calculated each time by using the value of R1, R2, and R3, but this makes the program shorter and simpler to enter.

Another thing to note is that the substructure can still directly access the local variables R1, R2, and R3. This is because the substructure is part of the code within the structure that creates the local variables and can therefore access them just like any other code within the main structure code. Note, however, that because YNUM is created within the substructure, any code outside the substructure can NOT access YNUM. Therefore, any code that accesses YNUM must be within the substructure code section.

A final item is the use of EVAL. You may wonder why this is necessary or even what it does. The answer is that the HP-48 code, when it references a program or structure object, places it onto the stack. Although you may not have thought much of it at the time, when you pressed ENTER after entering a program the HP-48 placed the program on the stack rather than execute it. Similarly, when the main structure code executes it will simply place the substructure on the stack rather than execute it. To have the structure execute the substructure code you must use the EVAL function. This function will evaluate the object on Level 1 of the stack. For a variable EVAL will return the contents of the variable. In the case of a program or structure EVAL will execute the code.

The next section shows how to enter the pY2D structure into the HP-48. Before reading through it, try to enter the code on your own using the structured program listing for above.

## 9.7 Entering the pY2D Structure

Enter the pY2D structure using the keystrokes shown:

| HP-48 | HP-49 | Comments |
|---|---|---|
| ↰ | ↱ | Specify a program object type to create the main structure. |
| << >> | << >> | |
| → | → | Initiate the local variable declaration. |
| SPC | SPC | |
| α | ALPHA | Lock the keyboard in alphabetic mode. |
| α | ALPHA | |
| R1 | R1 | Create the first local variable, called R1. The contents of Level 1 will be placed into this variable. |
| SPC | SPC | |
| R2 | R2 | Create the second local variable, called R2. The contents of Level 2 will be placed into this variable. |
| SPC | SPC | |
| R3 | R3 | Create the third local variable, called R3. The contents of Level 3 will be placed into this variable. |
| SPC | SPC | |
| ↰ | ↱ | Specify the code section of the structure. |
| << >> | << >> | |
| R1 | R1 | Recall R1 to the stack. |
| SPC | SPC | |
| R2 | R2 | Recall R2 to the stack. |
| SPC | SPC | |
| x | x | Calculate the product of R1 and R2. This is left on the stack. |
| SPC | SPC | |
| R1 | R1 | Recall R1 to the stack. This will automatically push the product of |

| | | |
|---|---|---|
| SPC | SPC | R1*R2 up. |
| R3 SPC | R3 SPC | Recall R3 to the stack. |
| x SPC | x SPC | Calculate the product of R1 and R3. This is left on Level 1 of the stack, while the product of R1 and R2 is left on Level 2. |
| R2 SPC | R2 SPC | Recall R2 to the stack. This will automatically push the products of R1*R2 and R1*R3 up. |
| R3 SPC | R3 SPC | Recall R3 to the stack. |
| x SPC | x SPC | Calculate the product of R1 and R3. This is left on Level 1 of the stack, while the product of R1 and R3 is left on Level 2 and the product of R1 and R2 is left on Level 3. |
| + SPC | + SPC | Add the contents of Levels 1 and 2. This will leave R1*R3 + R2*R3 in Level 1 and R1*R2 in Level 2. |
| + SPC | + SPC | Add the contents of Levels 1 and 2. This will leave R1*R2 + R1*R3 + R2*R3 in Level 1. |
| ↰ << >> | ↱ << >> | Specify a program object type to create the substructure. |
| → SPC | → SPC | Initiate the local variable declaration. |
| YNUM SPC | YNUM SPC | Create the local variable YNUM. The contents of Level 1 (namely R1*R2 + R1*R3 + R2*R3) will be placed into this variable. |
| ↰ << >> | ↱ << >> | Specify the code section of the substructure. |
| YNUM SPC | YNUM SPC | Recall YNUM to the stack. |
| R1 | R1 | Recall R3 to the stack. |

| | | |
|---|---|---|
| SPC | SPC | |
| | ALPHA | Turn off the keyboard alphabetic mode for the HP-48gII and HP-49g+. |
| ÷ SPC | ÷ SPC | Calculate the value of (R1*R2 + R1*R3 + R2*R3) / R1.  This will be left on Level 1 of the stack. |
| | ALPHA ALPHA | Lock the HP-48gII and HP-49g+ keyboard in alphabetic mode. |
| YNUM SPC | YNUM SPC | Recall YNUM to the stack. |
| R3 SPC | R3 SPC | Recall R3 to the stack. |
| ÷ SPC | ÷ SPC | Calculate the value of (R1*R2 + R1*R3 + R2*R3) / R3.  This will be left on Level 1 of the stack while the previous value in Level 1 is pushed up into Levels 2. |
| YNUM SPC | YNUM SPC | Recall YNUM to the stack. |
| R2 SPC | R2 SPC | Recall R2 to the stack. |
| α | ALPHA | Turn off the keyboard alphabetic mode. |
| ÷ | ÷ | Calculate the value of (R1*R2 + R1*R3 + R2*R3) / R2.  This will be left on Level 1 of the stack while the values in Levels 1 and 2 are pushed into Levels 2 and 3. |
| ► ► | ► ► | Cursor past the two inner sets of angle brackets. |
| EVAL | EVAL | Recall Y to the stack.  Note that this will place X on Level 2 and Y on Level 1 of the stack. |
| ENTER | ENTER | Place the structure on the stack. |

| | | |
|---|---|---|
| α<br>α | ALPHA<br>ALPHA | Lock the keyboard in alphabetic mode. |
| pY2D | pY2D | Create a variable name for the structure. Remember to use the "↰" shift key to enter the lower-case 'p'. |
| α | ALPHA | Turn off the keyboard alphabetic mode. |
| STO | STO | Store the structure in the variable pY2D. |

Now take the structure for a test spin.

**Example 9-2:** What are the resistances Ra, Rb, and Rc for a delta network that is equivalent to a wye network with R1 = 1.0 kΩ, R2 = 2.2 kΩ, and R3 = 5.6 kΩ?

**Solution:** Use the pY2D program to determine the values of Ra, Rb, and Rc.

| HP-48 | HP-49 | Display | Comments |
|---|---|---|---|
| 5600<br>ENTER | 5600<br>ENTER | 4:<br>3:<br>2:<br>1: 5.60E3 | Enter the value of R3 on the stack. |
| 2200<br>ENTER | 2200<br>ENTER | 4:<br>3:<br>2: 5.60E3<br>1: 2.20E3 | Enter the value of R2 on the stack. |
| 1000<br>ENTER | 1000<br>ENTER | 4:<br>3: 5.60E3<br>2: 2.20E3<br>1: 1.00E3 | Enter the value of R1 on the stack. |
| VAR | VAR | 4:<br>3: 5.60E3<br>2: 2.20E3<br>1: 1.00E3 | Access pY2D in the working directory. Depending upon the actual contents you may need to use NXT to see all lthe entries. |
| pY2D | pY2D | 4:<br>3: 3.59E3<br>2: 20.1E3<br>1: 9.15E3 | Execute pY2D. The answers are Ra = 9.15E3, Rb = 20.1E3, and Rc = 3.59E3. |

In this program you noticed that the values of R1, R2, and R3 were entered in reverse order, as the local variables expected R1 to be on Level 1 and R3 to be on Level 3. In practice it doesn't really matter in what order you enter the values as the delta and wye networks are symmetrical, but keeping the values in a specific order will help you to remember which stack levels contain which values. In this example the values are easy to remember as the code was written to place R1 and Ra, the "lowest" subscripted values, in the lowest stack level and R3 and Rc, the “highest” subscripted values, in the highest stack level.

Now that you’ve seen how to create a structure and use local variables, try programming your calculator to perform a delta-wye conversion using the following structured programming style to guide you.

```
“pD2Y”
<<
    → Ra  Rb  Rc                     ; Declare local variables R1, R2, R3
    <<
        Ra  Rb  Rc + +               ; Calculate (Ra + Rb + Rc)
        <<                           : Create a substructure with local variable DDEN
            → DDEN                   ; Store (R1 + R2 + R3) in DDEN
            <<
                Ra  Rb  *  DDEN  /   ; Calculate (Ra * Rb )/(R1 + R2 + R3)
                Rb  Rc  *  DDEN  /   ; Calculate (Rb * Rc )/(R1 + R2 + R3)
                Ra  Rc  *  DDEN  /   ; Calculate (Ra * Rc )/(R1 + R2 + R3)
            >>
        >>                           ; End of substructure
        EVAL                         ; See below for comments
    >>
>>                                   ; End of structure pD2Y
```

As with the previous programming example, the values are entered in reverse order so that Ra will be on Level 1, Rb on Level 2, and Rc on Level 3 and the calculated values of R1 on Level 1, R2, on Level 2, and R3, on Level 3.  One you have both pY2D and its inverse pD2Y in your calculator, you can use pY2D to generate the delta values from a wye network and then get the original delta values back just by using pD2Y on the calculated wye values.

## 10. In Conclusion

This tutorial has only scratched the surface of the capabilities of the HP-48. There are many more features, including equation editing, plotting, and symbolic calculations, that were beyond the scope of this brief tutorial. Now that you have sampled what you can do, feel free to experiment and see what you can do with your calculator. The more you know and more comfortable you feel with using it, the better it will serve you.

Happy calculating!