PROGRAMMING THE TI-59 & THE HP-41 CALCULATORS

BY PAUL GARRISON

No. 1442
$18.95

PROGRAMMING THE TI-59 & THE HP-41 CALCULATORS

BY PAUL GARRISON

TAB TAB BOOKS Inc.
BLUE RIDGE SUMMIT, PA. 17214

FIRST EDITION

FIRST PRINTING

Printed in the United States of America

Library of Congress Cataloging in Publication Data

Garrison, Paul.
Programming the TI-59 and the HP-41 calculators.

Includes index.
1. TI Programmable 59 (Calculating-machine) 2. HP-41 (Calculating-machine) and the H.P.-41 calculators.
QA76.8.T14G37 001.64'2 82-5718
ISBN 0-8306-2442-2 AACR2
ISBN 0-8306-1442-7 (pbk.)

Contents

Introduction

Most potential purchasers of Texas Instruments and Hewlett-Packard programmable calculators do not have an opportunity to work with both computers before deciding on one. I had the opportunity to work with the TI-59 and HP-41C for some time and have attempted to describe the way each works in as basic and simple a way as possible. This book is not intended to take the place of the instruction manuals prepared by the manufacturers, however. Those manuals contain very detailed descriptions of the multitude of functions available with each computer and are invaluable to the user.

I've often found manufacturer's instruction books extremely difficult to use, largely because they assume a degree of user familiarity with computers, a degree often lacking in the user who has just purchased his first programmable calculator. This book is designed to help you select one or the other computer, and to bridge the gap between the user who knows absolutely nothing about computer programming, and the more experienced user who can benefit from the kind of information which can be gleaned from the manufacturer's instruction manuals.

Writing this book has been a lot of fun. I enjoyed devising programs for each calculator, though I must admit to getting confused every so often by constantly having to switch back and forth between AOS (algebraic operating system) and RPN (reverse Polish notation). I can't honestly say I find that one performs better than the other. Both have peculiarities the user must get used to, but beyond that, the choice is largely one of personal preference.

To facilitate understanding of computer references, throughout this book you should be aware that computer **keys** are shown in boldface type, (**A**), and all *displays* are shown in italic type (*DISPLAY*).

I want to take this opportunity to express my appreciation to Texas Instruments and Hewlett-Packard for providing me with the printers for their computers. Without them the preparation of the program sections of this book would have been a nightmare.

Chapter 1

The Programmable Calculator

Texas Instruments and Hewlett-Packard refer to their machines, the TI-59 and HP-41, as *programmable calculators*; Radio Shack calls its similar product a *pocket computer*. Naming the machine is really a matter of semantics. According to *Webster's, to calculate* means *to ascertain by computation; to compute; to reckon up. To compute* means *to count; to reckon; to cast together several sums or particulars; to ascertain the amount.* There is no real difference between the meanings of the two words, though in day-to-day usage we tend to think that calculating contains a degree of estimation or judgment and computing ensures a greater degree of mathematic precision.

We generally describe the wide variety of electronic adding machines available to consumers as calculators, while we tend to think of computers as something larger, capable of storing huge amounts of information and data for on-demand use. But, except for the fact that their data-storage capability is limited, our programmable calculators are, in fact, full-fledged pocket computers, and I refer to them as computers throughout this book.

The difference between these computers and ordinary electronic calculators is that the calculator can perform a mathematical function only if and when every step involved in that function is keyed into it by the operator. Once the calculation is completed, all steps are forgotten. If the same calculation needs to be repeated, each step must again be keyed into the calculator.

This is not so with our computers. When told to do so, they will remember each step of a calculation (and sometimes there can be hundreds of steps) and repeat the entire series at the press of a single key. Thus, a computer is like a phonograph record or tape cassette: it records impulses and, when appropriately activated, uses those impulses as commands to perform certain given functions.

Despite their small size, the TI-59 and HP-41 computers have so wide a range of capabilities that each can take the beginner several months to learn to utilize all of their capabilities. Don't make the mistake of thinking these machines are endowed with some kind of supernatural intelligence, however. They are machines, and dumb machines at that. All they can really do is distinguish between two opposites or values such as yes and no, zero and one, or up and down. They get their aura of intelligence from the fact that they can perform these selection processes with such incredible speed that, they can come up with results to a problem that might take you weeks or even months to figure out. And, they can do it in seconds.

Not everyone can make good use of a computer, but many more people than are aware of it could use one to save time and improve their individual productivity. What tends to scare many people away from investigating the use of computers is the preconceived, though incorrect, idea that programming is some kind of mystical art that must be performed by experts, an art which the average mortal can never expect to master. That is utter nonsense.

A program is any mathematical formula, no matter how simple, which must be performed more than once. A thorough knowledge of mathematics, though sometimes useful, is not necessary. The average person, with the ability to add, subtract, multiply, and divide with pencil and paper, will find that he or she can create some fairly complicated programs with a minimum of difficulty.

Personal computers, whether portable or the desk-top variety, are part of our lives today and will undoubtedly be more prominent in our lives tomorrow. And it goes without saying that the better we understand their capabilities and limitations, the better we'll be able to use them to a good advantage.

Chapter 2
Texas Instruments TI-59

Texas Instruments TI-59 (Fig. 2-1) is a complete system which includes the card reader. Its list price is $250, but it is sold at some discount stores for under $200. The only major accessory available for the TI-59 is the thermal printer (Fig. 2-2). This is a fairly large unit which must be used with standard house current and is therefore not easily portable. Its list price is $225 and is advertised by discount houses for between $150 and $170.

A powerful computer, TI-59 can handle a variety of computer programs. It does not have a nonvolatile memory: whenever it is turned off, all data and programs previously stored in the computer are lost. The only way to retain programs is to write them on magnetic cards before turning off the computer. The TI-59 deals solely with numbers and uses no letters in its display.

The TI-59 operates on the principle of the algebraic operating system (AOS). Problems are entered into the computer in exactly the same way in which they are dealt with using pencil and paper. For instance, to add two and two you press the keys **2 + 2 =**. Unlike the RPN (reverse Polish notation) system, parentheses and parentheses within parentheses can be used with AOS to solve complicated calculations.

Texas Instruments markets a fairly long list of special-purpose programs for the TI-59. They are available in the form of a module which retains them permanently. (See the Appendix for a list of software available from the manufacturers.)

The TI-59 keyboard consists of 45 keys. All but three are designed to perform more than one function. The keys (Fig. 2-3) are arranged in nine rows of five keys each. You can change the register configuration of the computer to fit the type and length of program you will enter (by writing directly to the keyboard or reading the program from a magnetic library

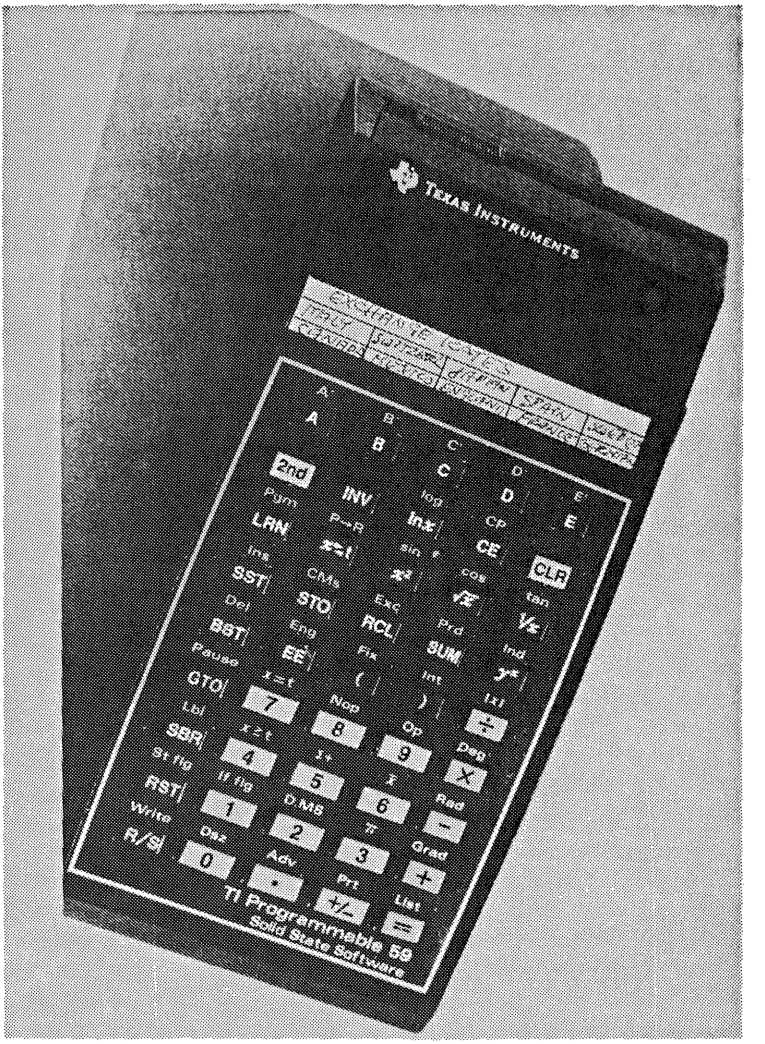


Fig. 2-1. The Texas Instruments TI-59 programmable calculator.

card for execution). When the computer is first turned on, its memory is divided into room for 480 program steps and 60 data storage registers. This configuration can be changed to accommodate as many as 960 program steps, in which case no registers are available for data storage or retrieval, or as few as 160 program steps, in which case 100 positions are available for data storage.

There are two slots on the TI-59 computer. One is used to feed magnetic cards (Fig. 2-4) into the card reader, to write a new program or

execute a previously written program. The other slot displays the magnetic program card or a facsimile of it, helping the operator remember which steps must be taken to run the program.

The display is formed by light-emitting diodes (LEDs). These use a fair amount of electricity, limiting the computer operation to only two to three hours of continuous operation between battery-charge cycles. Whenever possible, it is a good idea to keep the computer plugged into the converter and, in turn, into the wall socket. The card reader motor especially tends to drain the batteries quickly, and repeatedly using the card reader on battery power along should be avoided, if possible.

The keys of the TI-59 are set in left-to-right horizontal rows of five keys each. Each key has a primary function (function 1) and a secondary function (function 2). (Primary functions are found on the face of each key on the TI-59; secondary functions are printed above each key.) The descriptions which follow begin with the top row of keys and end with the bottom row. Following the row, key, and function identification, you will find the identifying mark for each key function listed.

Row 1

Keys 1-5, function 1: **A, B, C, D, E.** These are so-called user-defined keys. They can be used as labels to identify entire programs, or subroutines, or to store information as part of a program or to activate the operation of a program or program section.

Keys 1-5, function 2: **A′, B′, C′, D′, E′.** These, too, are user-defined keys and operate in the same manner as the primary function of these keys.

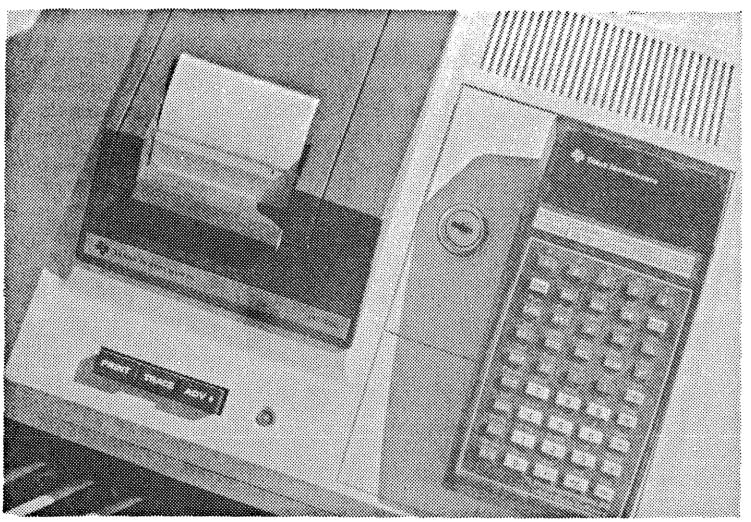


Fig. 2-2. The TI-59 computer system includes an optional printer which provides hard copies of programs and computer runs.

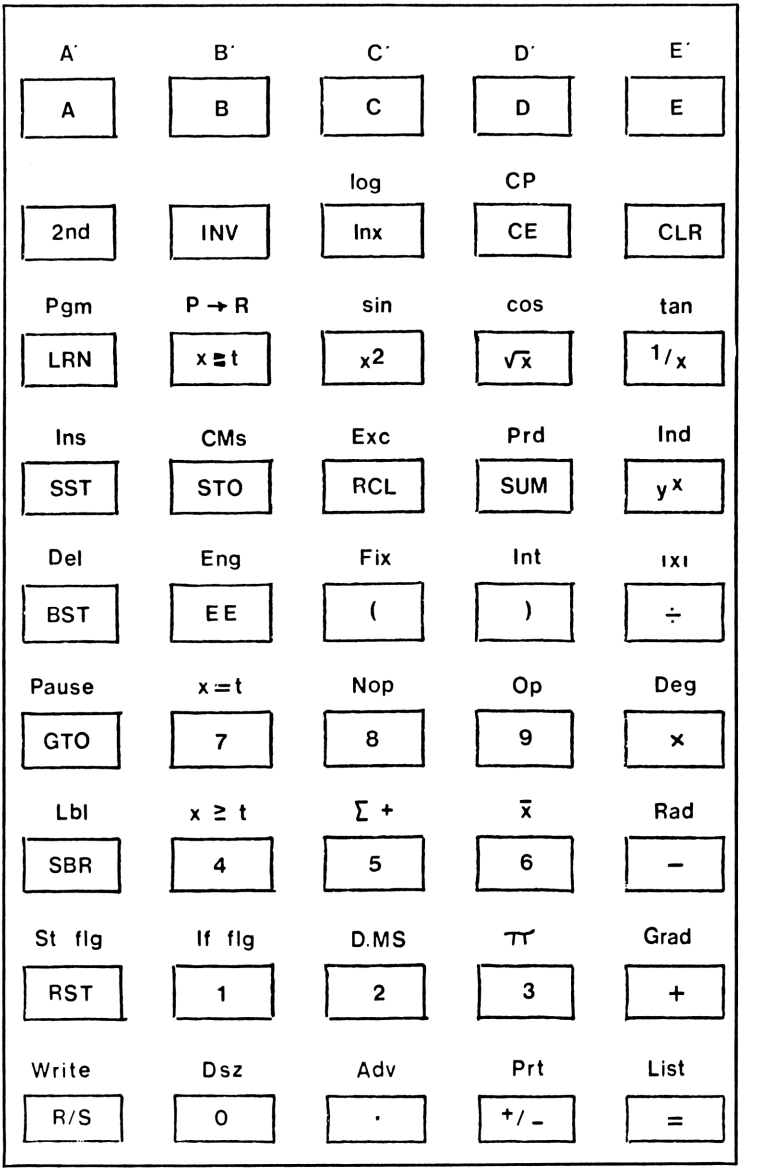


Fig. 2-3. The TI-59 keyboard.

Row 2

Key 1, function 1: **2nd**. This is the shift key. Pressing it before pressing any other key will activate that key's secondary function. This key has no secondary function.

Key 2, function 1: **INV**. This key also changes the function of the key next pressed, by reversing its meaning. Example: pressing **INV sin** will produce the arc sine of the previously displayed value. This key, like the **2nd** key, has no secondary function. The results of pressing the **INV** key in conjunction with other keys are presented with the description of each individual key.

Key 3, function 1: **lnx**. This key calculates the natural logarithm of the displayed value. Example: **10 lnx** produces the answer *2.302585093*.

Keys 2 and 3, function 1: **INV lnx**. This key performs e^x which is the reverse of the logarithm, the natural antilogarithm. Example: **2.302585093 INV lnx** produces *10*.

Key 3, function 2: **log**. This key calculates the common logarithm of the displayed value. Example: **10 2nd log** results in *1*.

Keys 2 and 3, function 2: **INV log**. Computes the common antilogarithm. Example: **1 INV 2nd log** results in *10*.

Key 4, function 1: **CE**. Clear entry: this key clears the last entry without disturbing any other information. It does not disturb operations in progress. The key can also be used to stop a flashing display from flashing.

Key 4, function 2: **CP**. This key clears all program memory locations, clears subroutines, resets flags, and clears the T register. It does not affect data stored in the memory registers. When used as a step in a program, this key clears only the T register.

Key 5, function 1: **CLR**. This key clears the display and all calculations in progress. It will also stop the flashing of a flashing display. Using this key does not affect the data stored in memory registers, the program memories,

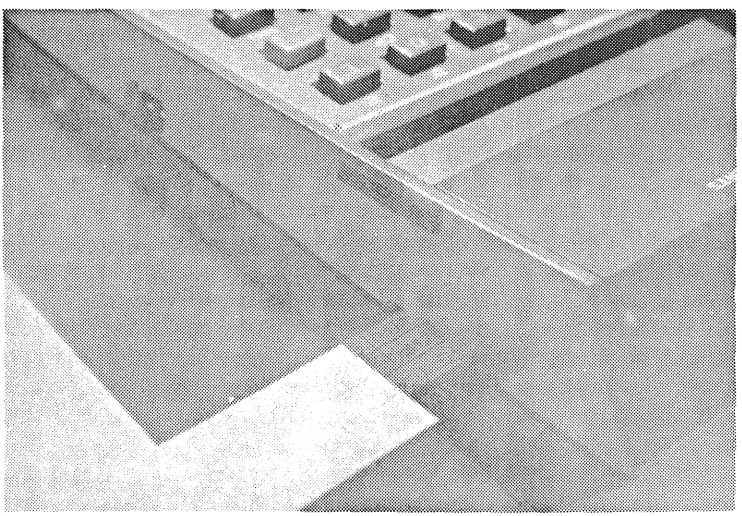


Fig. 2-4. Magnetic cards are used to record programs. The card reader is an integral part of the computer.

the computer partitioning, the contents of the T register, the fixed-decimal display format and angular mode or engineering notation. This key has no secondary function.

Row 3

Key 1, function 1: **LRN**. The learn key commands the computer to remember every keystroke for subsequent execution as part of running a program. When **LRN** is pressed, the display changes from *0* to *000 00*, and when program steps are entered, the left three digits represent the number of the program step, and the right two digits identify the key stroke. Example: *007 95* means that the eighth program step (steps are numbered starting with 000), ninth row, fifth key is = (Fig. 2-5).

Key 1, function 2: **Pgm**. Program: this key is used to call up pre-recorded programs which are part of the program library available from Texas Instruments. It must be used with two numbers which identify the program number in the program card (module) currently installed in the computer. Example: With the Aviation Module in the computer, pressing **2nd Pgm 11** activates the Great Circle Navigation program.

Key 2, function 1: **x ≷ t**. Pressing this key exchanges the contents of the X register with those of the T register, putting the latter into the display.

Key 2, function 2: **P→R**. This performs polar to rectangular conversion.

Key 2, function 2: **INV P R**. Performs rectangular to polar conversion.

Key 3, function 1: $\mathbf{x^2}$. Produces the square of the displayed value. Example: **12** $\mathbf{x^2}$ results in **144.**

Key 3, function 2: **sin**. Computes the sine of the displayed value. Example: **10 2nd sin** results in *.1736481777*.

Key 3, function 2: **INV sin.** Computes the arc sine of the displayed number. Example: **.176481777 INV 2nd sin** results in *10*.

Key 4, function 1: $\sqrt{\mathbf{x}}$. Computes the square root of the displayed number. Example: **144** $\sqrt{\mathbf{x}}$ results in *12*.

Key 4, function 2: **cos**. Computes the cosine of the value in display. Example: **10 2nd cos** results in *0.984807753*.

Key 4, function 2: **INV cos.** Computes the arc cosine of the displayed value. Example: **0.984807753 INV 2nd cos** results in *10*.

Key 5, function 1: **1/x**. Computes the reciprocal value of the displayed value. Example: **10 1/x** results in *0.1*. The reciprocal value is the value which will bring the same result as its reciprocal when the multiplication and division functions are reversed. Example: **100 ÷ 10** = *10;* **100 × 0.1** = *10*.

Key 5, function 2: **tan**. Computes the tangent of the value in the display. Example: **10 2nd tan** results in *.1763269807*.

Key 5, function 2: **INV tan.** Computes the arc tangent of the value in the display. Example: **.1763269807 INV 2nd tan** results in *10*.

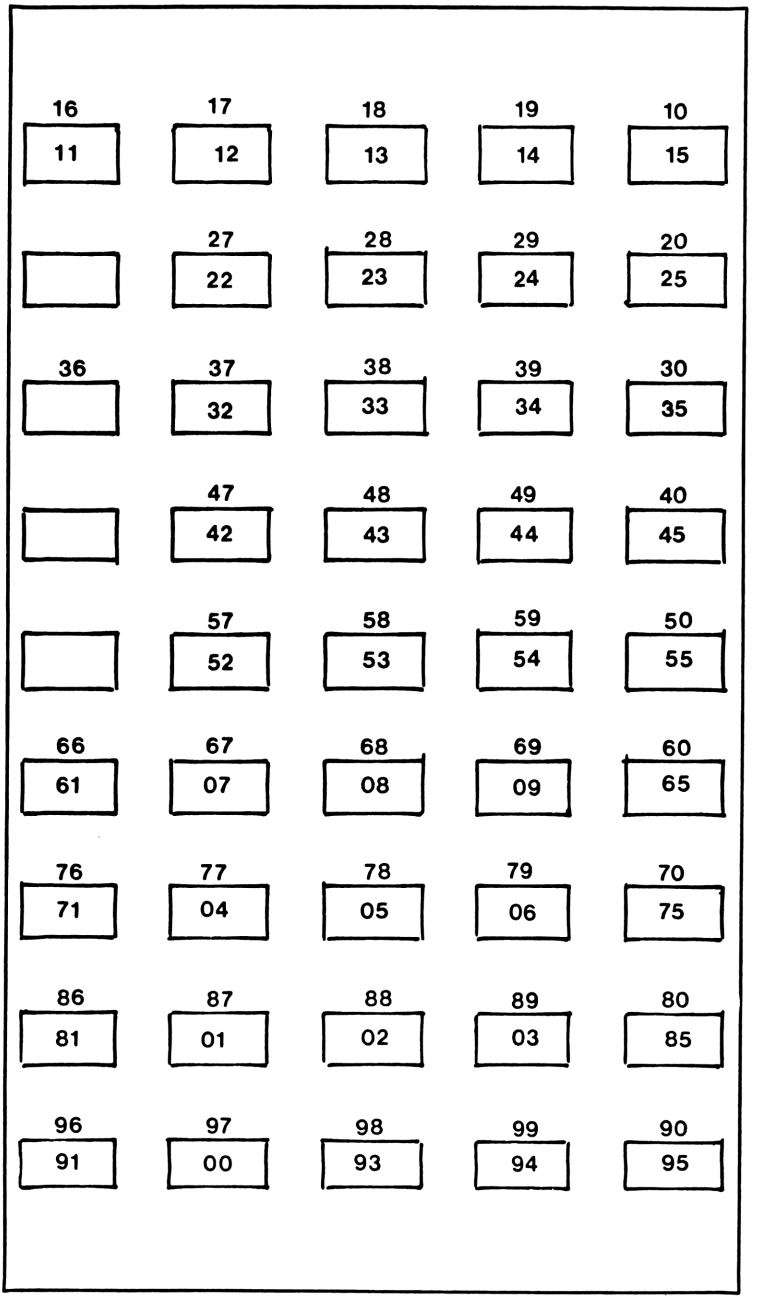


Fig. 2-5. The TI-59 display deals only in digits, and programmable keys are assigned a number which appears in the display.

Row 4

Key 1, function 1: **SST.** Single step; used in the program mode to advance the program one step at a time.

Key 1, function 2: **Ins.** Insert; pressing this key in program mode will provide one empty step **ahead** of the step shown in the display, permitting you to insert one new step. Such an insertion automatically changes the numbers of each subsequent program step by one.

Key 2, function 1: **STO.** Store; when used in conjunction with two digits, this function stores the displayed value in the memory position identified by those two digits.

Key 2, function 2: **CMs.** This key clears all data stored in data memories.

Key 3, function 1: **RCL.** Recall; recalls data from data memories when used in conjunction with two digits identifying the memory position in question.

Key 3, function 2: **Exc.** Exchange; using this key in conjunction with two digits exchanges the data stored in the memory register identified by those two digits with the data in the display. The result is that the previously stored data is moved to the display, and the previously displayed data is now stored in that memory register.

Key 4, function 1: **SUM.** Used with two digits to identify the appropriate memory register, this key will add the amount in the display to the amount stored in that particular memory register. The display remains unchanged. To view the result of the addition process, the data has to be recalled from memory, using either the **RCL** or **2nd Exc** keys.

Key 4, function 1: **INV SUM.** This key, used in conjunction with two digits, subtracts the value in the display from the value stored in the register identified by those digits. To view the result, use either the **RCL** or the **2nd Exc** key.

Key 4, function 2: **Prd.** This key, when used in conjunction with two digits identifying the appropriate memory register, multiplies the value in display with the value stored in that memory register, and stores the product in that register. To view the result it is necessary to use either the **RCL** or the **2nd Exc** key.

Key 4, function 2: **INV Prd.** This key divides the contents of the memory register with the value in the display and then stores the result in the memory register. It must be used with two digits identifying the register. To view the result, use either the **RCL** or the **2nd Exc** key.

Key 5, function 1: $\mathbf{y^x}$. Raises the displayed value to a given power. Example: **10** $\mathbf{y^x}$ **5** = *100,000* (where 10 equals y and 5 equals x).

Key 5, function 1: **INV** $\mathbf{y^x}$. Produces the x root of y. Example: **10 INV** $\mathbf{y^x}$ **5** = results in *1.584893192.*

Key 5, function 2: **Ind.** Used in conjunction with two digits, performs indirect operations. The indirect function is explained in detail in the TI-59 instruction manual.

Row 5

Key 1, function 1: **BST**. Back step; used to make one step at a time backwards through a program with the computer in program mode.

Key 1, function 2: **Del**. Delete, this key will delete the program step in the display from the program and will change the number of all subsequent program steps by one.

Key 2, function 1: **EE**. Enter exponent; using this key informs the computer that the next number is an exponent of 10. All subsequent data will be displayed in scientific notation format until **CLR** is pressed, or until **INV EE** is used.

Key 2, function 1: **INV EE**. Negates the result of the previously used **EE**.

Key 2, function 2: **Eng**. Places the subsequent displays into engineering notation format.

Key 2, function 2: **INV Eng**. Negates the result of the **Eng** command.

Key 3, function 1: (. Open parenthesis. The computer is designed to be able to deal with math functions in up to nine sets of parentheses set within each other.

Key 3, function 2: **Fix**. Used in conjunction with one digit, this key controls the number of digits to the right of the decimal point which appear in the display. It does not remove fractions from the computer memory, which always uses the total figure (including all fractions) for calculations, even though these fractions do not appear in the display. Example: The number in the computer is *1.234567890*. By pressing **2nd Fix 4** the display changes to *1.2346*, and pressing **2nd Fix 0** produces a display of *1*.

Key 3, function 2: **INV Fix**. Using the **INV 2nd Fix** function removes the effect of a previous **Fix** entry and again displays the total figure, including all fractions.

Key 4, function 1:). Close parenthesis.

Key 4, function 2: **Int**. This key deletes the fraction from the display and from all calculations, retaining only the integer portion of the number. Example: **123.4567 2nd Int** results in *123*.

Key 4, function 2: **INV Int**. Removes the integer portion of the number and retains only the fractions. Example: **123.4567 INV 2nd Int** results in *.4567*.

Key 5, function 1: ÷. This is the command to divide once = is pressed.

Key 5, function 2: **1 × 1**. Causes always the absolute value of a number to remain in the display and to be used in calculations. Example: *−123.45* becomes *123.45* but a + number is not affected by this key.

Row 6

Key 1, function 1: **GTO**. Go to; when used as a step within a program, it commands the computer to change course and go to a designated subroutine. If pressed when the computer is not in program mode it will, if followed by the appropriate digits, position the program pointer to a desired line. Pressing **LRN** will then bring the line into display.

Key 1, function 2: **Pause**. When used as a step within a program, this key inserts a pause lasting part of a second, in order to facilitate reading the display. Several pauses can be strung together to increase the length of time the data remains in the display. When the key is held down during program execution, it will cause the result of each step to be displayed briefly. In order to monitor a library program in this manner, that program must first be placed into the computer, recorded on a magnetic card, and then reentered into the computer from the card.

Key 2, function 1: **7**. The digit 7.

Key 2, function 2: **x=t**. This key is one of the decision-making keys. If the contents of the X register is equal to that of the T register, the computer will execute the next step. If not, it will skip the next step and execute the one following.

Key 2, function 2: **INV x=t**. Here the effect is reversed. If the contents of the two registers are *not* equal, the computer will execute the next step. If they *are* equal it will skip the next step and go to the one following.

Key 3, function 1: **8**. The digit 8.

Key 3, function 2: **Nop**. No operation; this key can be used to cause an unwanted program instruction to be ignored by the computer, or to provide spacing between parts of a program to simplify subsequent additions.

Key 4, function 1: **9**. The digit 9.

Key 4, function 2: **Op**. This key can be used to perform a variety of functions, all depending on the two digits which are pressed in conjunction with this key:

2nd Op 00. Used in conjunction with the optional thermal printer, it initializes the print register.

2nd Op 01. Used with the printer, it places 10 digits into the display as alpha-numeric codes.

2nd Op 02, 03 and **04**. These are the same as **01**, for different portions of the print column.

2nd Op 05. Use with the printer: prints the contents of the print register.

2nd Op 06. Prints the last four characters of **OP 04** with the current display value. Use with the printer.

2nd Op 07. Use with printer; plots a * curve in column 0-19 guided by the display.

2nd Op 08. Lists all labels currently in the program memory; use with the printer.

2nd Op 09. When used in conjunction with an identification, it will place a library program into the computer program memory. This is important when the operator wants to edit or change a prerecorded library program. Example: **2nd Pgm 15 2nd Op 09** will bring program number 15 of whatever library module is currently installed into display. For example, if the Master Library module is in place, the called-up program would be the Random Number Generator program.

2nd Op 10. Applies signum function to the value in the display register.

2nd Op 11. Calculates variances; two or more data blocks must have been previously entered.

2nd Op 12. Calculates slope and intercept.

2nd Op 13. Calculates correlation coefficient.

2nd Op 14. Calculates y′ for an x in the display.

2nd Op 15. Calculates x′ for a y in the display.

2nd Op 16. Displays the current partition of the memory storage area. Unless the partition has been changed by the operator since the computer was turned on, the display will show *479.59*.

2nd Op 17. When preceded by one digit, this key combination repartitions the memory storage area. The leading digit represents 10 data storage registers. Example: **2 2nd Op 17** results in *799.19* and **8 2nd Op 17** results in *319.79*.

2nd Op 18. If there is no error condition in the program, it sets flag 7.

2nd Op 19. If there is an error condition in the program, it sets flag 7.

2nd Op 20-29. Adds 1 to the value stored in any of the first 10 registers (00 through 09), as identified by the second digit. Nothing changes in the display and to see the effect either **RCL** or **2nd Exc** must be pressed.

2nd Op 30-39. Similar to 20-29 except the stored value is decremented by **1.**

Key 5, function 1: **x**. The command to multiply, when = is pressed.

Key 5, function 2: **Deg**. Places the computer into the degree mode.

Row 7

Key 1, function 1: **SBR**. Subroutine; used in conjunction with some type of identification this key labels a subroutine.

Key 1, function 1: **INV SBR.** Identifies the end of a subroutine and tells the computer to return to the place in the program following the branch to the subroutine.

Key 1, function 2: **Lbl**. Label; used in conjunction with other keys to identify a program as a whole or a portion of a program.

Key 2, function 1: **4**. The digit 4.

Key 2, function 2: **x ≥ t**. A decision-making key. If the value in the X register is equal to or greater than that in the T register, the computer will execute the next step. If not, it will skip that step and go to the following step.

Key 2, function 2: **INV x ≥ t**. If the value in the X register is smaller than that in the T register, the computer executes the next step. Otherwise it skips to the following step.

Key 3, function 1: **5**. The digit 5.

Key 3, function 2: **Σ +**. Assimilates variable pairs into data registers 01 through 06.

Key 3, function 2: **INV Σ +**. Removes such data pairs.

Key 4, function 1: **6**. The digit 6.

Key 4, function 2: $\bar{x}$. Calculates and displays the mean of the dependent y-array data. Pressing **x ≷ t** displays the mean of the independent x-array data.

Key 4, function 2: **INV** $\bar{x}$. Calculates and displays the standard deviation of the dependent y-array data. Pressing **x ≷ t** displays the standard deviation of the independent x-array data.

Key 5, function 1: **−**. The command to subtract once = is pressed.

Key 5, function 2: **Rad**. Places the computer in the radian mode.

Row 8

Key 1, function 1: **RST**. Reset; causes the computer to reset the program pointer to the 000 position. It clears the subroutine return register and resets program flags.

Key 1, function 2: **St flg**. Set flag; the computer can accommodate 10 flags numbered 0 through 9, and the key must be used with one digit to identify the flag number being set.

Key 1, function 2: **INV St flg.** Also used with one digit, it removes the flag identified by that digit.

Key 2, function 1: **1**. The digit 1.

Key 2, function 2: **If flg.** This key, too, must be used in conjunction with one number to identify the flag in question. It is a decision-making key. If the called-for flag is set, the computer will execute the next step. If not, it will skip that step and go on to the next.

Key 2, function 2: **INV If flg.** Same as the last key except that the computer will execute the next step if no flag is set. If the flag is set it will go to the following step.

Key 3, function 1: **2.** The digit 2.

Key 3, function 2: **D.MS.** Degrees, minutes, seconds, changes values expressed in degrees, minutes and seconds or hours, minutes and seconds into the comparable values expressed in the decimal format.

Key 3, function 2: **INV D.MS.** Changes values expressed in the decimal format into the comparable values expressed as degrees, minutes and seconds, or hours, minutes and seconds. Example: **12.56 INV 2nd D.MS** results in *12.3336* representing 12 hours or degrees, 33 minutes and 36 seconds.

Key 4, function 1: **3.** The digit 3.

Key 4, function 2: **π**. Pi; pressing this key puts *3.141592654* into the display.

Key 5, function 1: **+**. The command to add once = is pressed.

Key 5, function 2: **Grad.** Places the computer into the grad mode.

Row 9

Key 1, function 1: **R/S.** Run/stop; the key is used when writing a program and during program execution to stop the program execution and start the program again once it has stopped.

Key 1, function 2: **Write.** This key is used to cause the computer to write the contents of its program and data memories onto magnetic cards. It should be used in conjunction with digits 1, 2, 3 or 4, identifying the consecutive order of the passes through the card reader.

Key 1, function 2: **INV Write.** This is used as a command in a program to cause the computer to read a given card at the right moment. The card is inserted before executing the program, but not read until the command is reached during execution.

Key 2, function 1: **0.** The digit 0.

Key 2, function 2: **Dsz.** Decrement and skip on zero; this key decreases the contents of a data register (0 through 9) by one, and when it reaches zero skips one line in the program. At any other time it executes the next line.

Key 2, function 2: **INV Dsz.** Skip one line each time the amount in the applicable register is not equal to zero, and execute the next line when it is equal to zero.

Key 3, function 1: . Decimal point.

Key 3, function 2: **Adv.** Used only with the printer, it advances the paper by one space, achieving doublespaced printing.

Key 4, function 1: **+/−**. Changes the sign of the value shown in the display. Example: **4.3 +/−** results in *−4.3*. Pressing **+/−** again returns the display to *4.3*.

Key 4, function 2: **Prt.** When used as a step in the program, it causes the printer to start printing. Can also be executed from the keyboard.

Key 5, function 1: **=**. The equals sign activates previously entered calculation commands and displays the results.

Key 5, function 2: **List.** Lists every step in the program starting at the position the program pointer is currently located. To start from the beginning, press **RST 2nd List**.

Key 5, function 2: **INV List.** Lists the contents of all data registers, starting with the register currently in the display. To start with any other register, first press the desired starting register. Example: To start the listing with register 00 press **00 INV 2nd List**. Can be used only in conjunction with the printer.

Since the display capability of the TI-59 is limited to numerals, each key and each added function of each key has been given a number, and it is that number which appears in the program display when a program is being written or program steps are checked.

Figure 2-5 shows the TI-59 keyboard with numbers which stand for its various key designations.

When you write programs for the TI-59, every key stroke, with very few exceptions, is a step in the program. Thus, the figure 1000, when part of a program instruction, becomes four separate program steps: one—zero—zero—zero. The exceptions are the **2nd** key, which does not constitute a separate program step since it is invariably part of a two-key instruction. Also, the two digits which follow keys requiring a two-digit identification (such as **STO** and **RCL**) represent a single step. Thus, **RCL 00** is a two-step command.

When **GTO** is inserted in the program, followed by a three-digit identification of the line to which the computer is supposed to go, those three digits represent only two steps. **GTO 133**, the command to go to line 133, involves three program steps: **GTO - 01 - 33**. In these cases it is necessary to press the identifying key (**STO, RCL,** etc.) followed by the digits. If a mistake is made, it can only be corrected by again pressing **STO** (or **RCL** or whatever) and then the correct digits. Barring that, the two digits would end up as two separate program steps.

Chapter 3

Hewlett-Packard HP-41

Hewlett-Packard was the first company to produce and market a programmable calculator: the HP-65 sold for around $800 but is no longer in production, having been superceded by a number of different models. The HP-41C is the latest and most powerful of the group.

The HP-41C (Fig. 3-1) consists of a basic portable computer priced around $250. It can usually be obtained from discount houses for a little under $200. The HP-41C has enough storage and memory space for the average user, though anyone planning to get involved in fairly lengthy and complicated programs may find its 63 registers insufficient. Hewlett-Packard offers optional plug-in memories that increase the maximum number of available registers by 64 each (Fig. 3-2). A total of four of these plug-in modules can be added to the basic unit, increasing the number of available registers to 319. Another option is the Quad Memory Module, which takes the place of four individual modules and increases the number of registers to 319. With quad module a card reader or printer or both can also be plugged into the computer. This is not possible when more than three of the individual plug-in memory modules are being used because they take up all available ports.

A good alternative is the HP-41CV. It has the full range of 319 registers installed in the basic unit and can, therefore, be used in conjunction with the other accessories. From here on the text refers to the HP-41 C unless the HP-41CV is specified. Where applicable, the number of registers required for a given operation is noted.

The two most valuable features of the HP-41 are its capability to communicate with the user in alphanumeric language, and its nonvolatile memory.

The alphanumeric capability allows the operator to give instructions to the computer using a code resembling abbreviated ordinary English (Fig.

Fig. 3-1. The HP-41C programmable calculator.

3-3), and the computer can talk back to the operator in the same way. The operator can also place words and sentences into the display. This makes the execution of complicated programs easy, even for a person who is unfamiliar with the routines necessary to run the program. This capability is virtually indispensible.

Nonvolatile memory means that any information that has been entered into the computer will remain there even when it is shut off. (As a matter of fact, it remains even while old batteries are removed and new ones in-

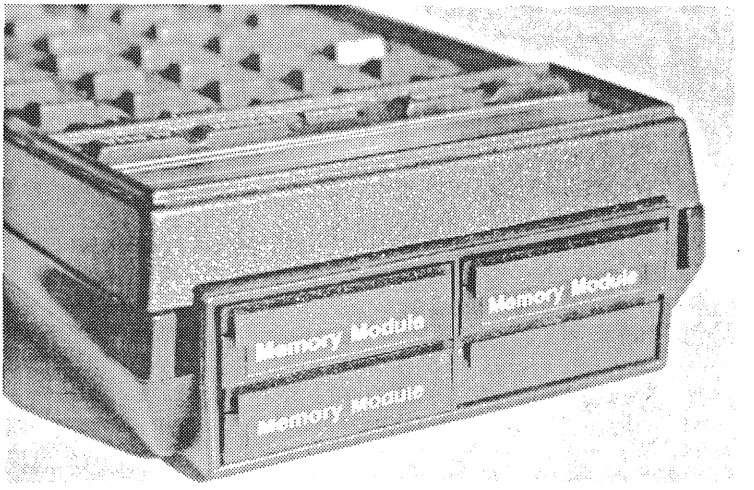


Fig. 3-2. Plug-in memories are available to increase the number of registers.

stalled.) The only way to get rid of something stored in the computer is to purposely erase what happens to be in the display, to replace the contents of a register with new data or information, or to clear out the entire computer, a procedure which can be done only by using two keys simultaneously, a safeguard to avoid accidental erasure. The clearing procedure results in this message in the display window: *MEMORY LOST.*

Depending on the length of programs (and on the number of available registers), the computer can store several programs permanently in its memory. They can be called up at will by pressing a single key.

Aside from the plug-in memories, the primary accessories for the HP-41 are a card reader, a thermal printer, and a rechargeable battery pack.

The card reader (Fig. 3-4) lets you record programs and other data on magnetic cards for future use. The card reader may be of little use to a person who uses the prepared programs offered by Hewlett-Packard (in their series of special Application Pacs) exclusively. A user who writes his or her own programs, but does not have a card reader, must be careful that the number and size of the programs do not exceed the register capability of the computer, and that the programs are designed to avoid using data storage registers which are also used by the prerecorded Application Pacs. (This will be explained further in Chapter 6.)

Besides its function of reading and writing to magnetic cards, the card reader contains a considerable number of preprogrammed instructions.

Frequent use of the card reader runs down the batteries, which otherwise can be expected to last a year or more. If you expect to use the card reader often, you should consider spending $30 for the rechargeable battery pack. Replacement batteries (four N-size batteries) cost $4 each.

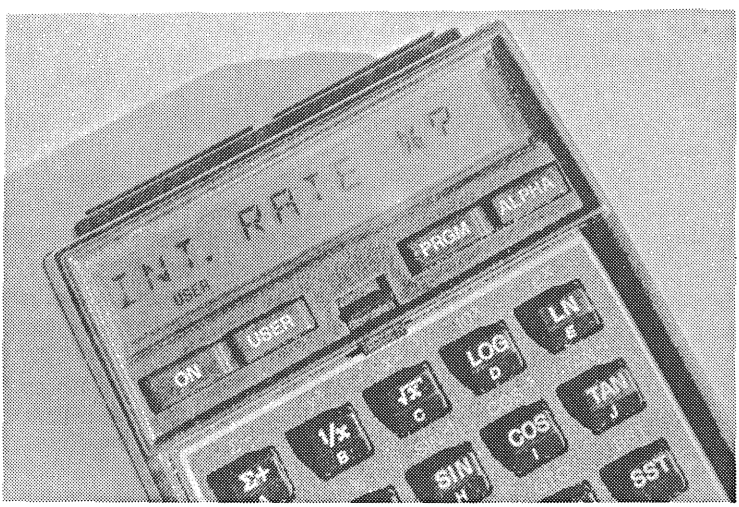


Fig. 3-3. The alphanumeric display communicates in something close to plain English.

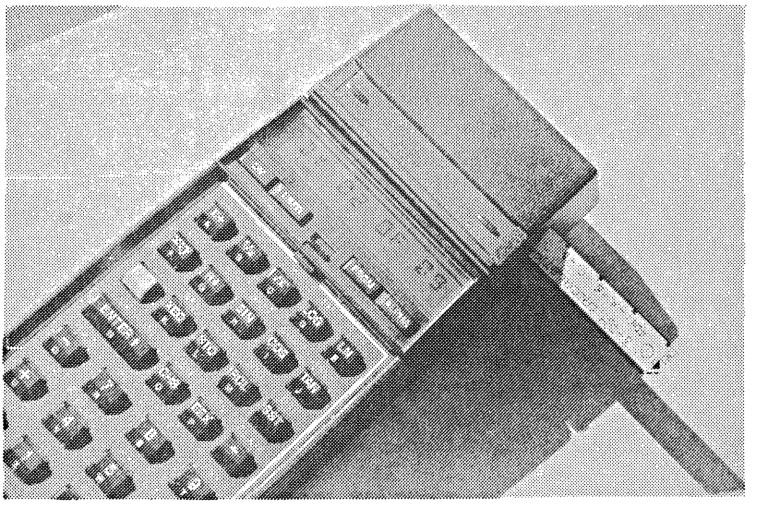


Fig. 3-4. The card reader for the HP-41C is an expensive but useful option.

The printer (Fig. 3-5) is a small, lightweight unit that comes in a carrying case that can easily travel with you. It, too, uses a fair amount of electricity. Though it will operate for a while on its batteries, the rechargeable battery pack does require recharging fairly frequently. This $350 printer will record entire programs step by step, or it simply prints the specified results.

The HP-41 (Fig. 3-6) consists of 35 keys arranged in three rows of five keys and five rows of four keys, plus four special-purpose keys at the top. Most of the keys have a description printed on top and another on the slanted section of the front of the key, shown in the bottom portion of each key in Fig. 3-6. In addition, all but one of the regular keys have a third function which is printed on the face of the computer above each key, and which can be executed by first pressing the shift key (shown in black in the illustration).

The keyboard drawing on the back of the computer shows additional key functions which can be called up when the computer is in the alpha mode. Figure 3-7 shows what each key represents in the alpha mode. The primary meaning of the key is printed on its face, and the secondary function, available by first pressing the shift key, is shown above each key. Don't forget these additional key capabilities because they aren't on the face of the computer.

What follows is a brief description of the function performed by each key in the normal, normal-shift, alpha and alpha-shift modes. The keys in each row are described from left (key 1) to right (key 4 or key 5).

Top Row

Key 1: **ON.** This key turns the computer on and off.

Key 2: **USER.** This key places the computer in user mode and the word **USER** will appear in the left-hand bottom corner of the display. User mode means that any key on the keyboard assigned a meaning other than its original meaning by the operator will execute that meaning instead of its

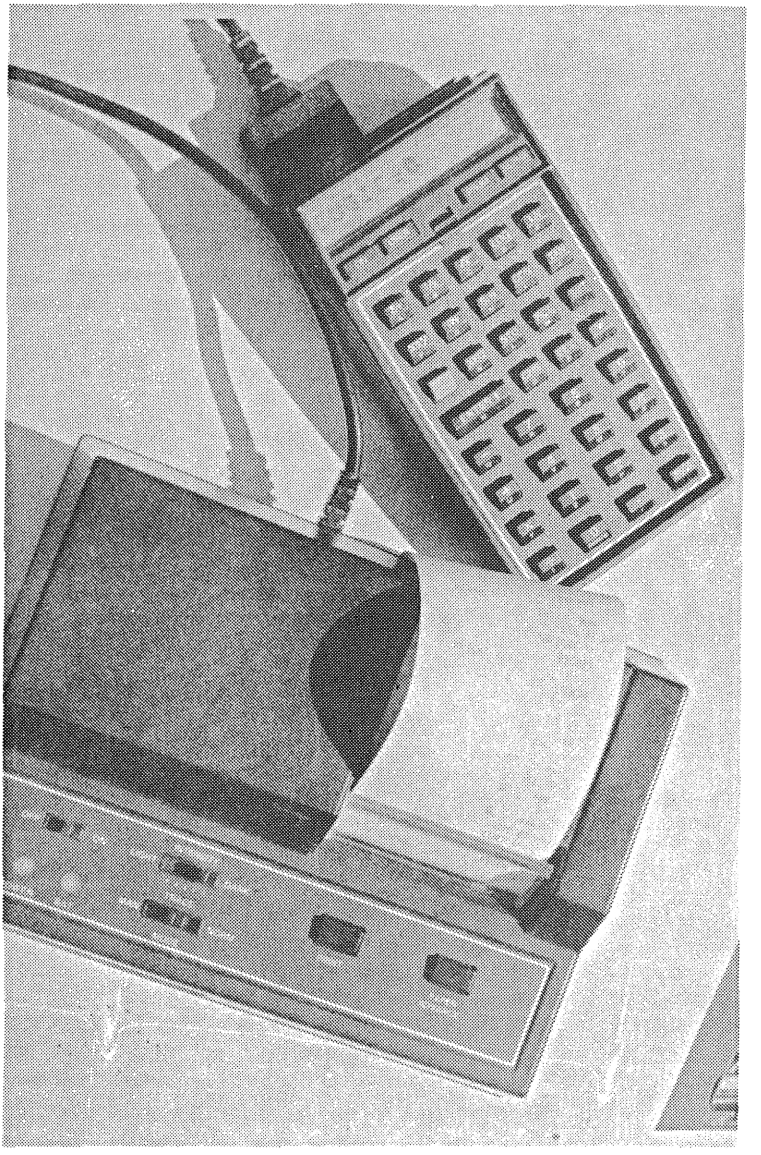


Fig. 3-5. The optional printer for the HP-41C is a battery-operated portable system.

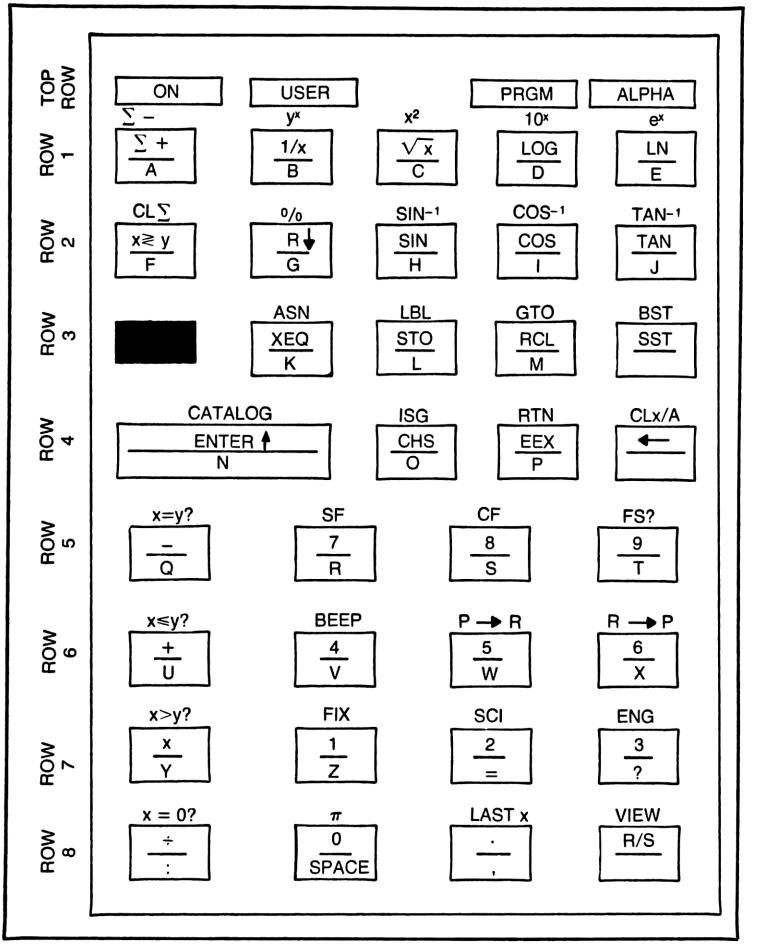


Fig. 3-6. The basic keyboard of the HP-41C.

original function. This saves the operator many key strokes. You cannot perform standard calculations with the computer in user mode. Pressing **USER** when the computer is in user mode will return it to the normal mode.

Key 3: **PRGM.** This key tells the computer that the information next input is part of a program. The computer will not execute the functions keyed in, but will remember each function and bit of information as it is entered, and execute the set when called upon. The word **PRGM** will appear two thirds of the way toward the right at the bottom of the display window, and the display will consist of a two- or three-digit number to the left, and additional information in the form of digits and letters to the right. Thus, if the 33rd step in a program calls for the computer to retrieve the information stored in memory register number 07, the display will

show: *33 RCL 07* with the annunciator PRGM displayed below. Pressing the **PRGM** key when the computer is in program mode will take it out of program mode.

Key 4: **ALPHA.** This key changes just about everything. The alpha keyboard (Fig 3-7) goes into effect, replacing the original functions. Pressing **ALPHA** when the computer is in alpha mode and the annunciator alpha is visible in the right-hand bottom corner of the display, will return the keyboard to its normal function.

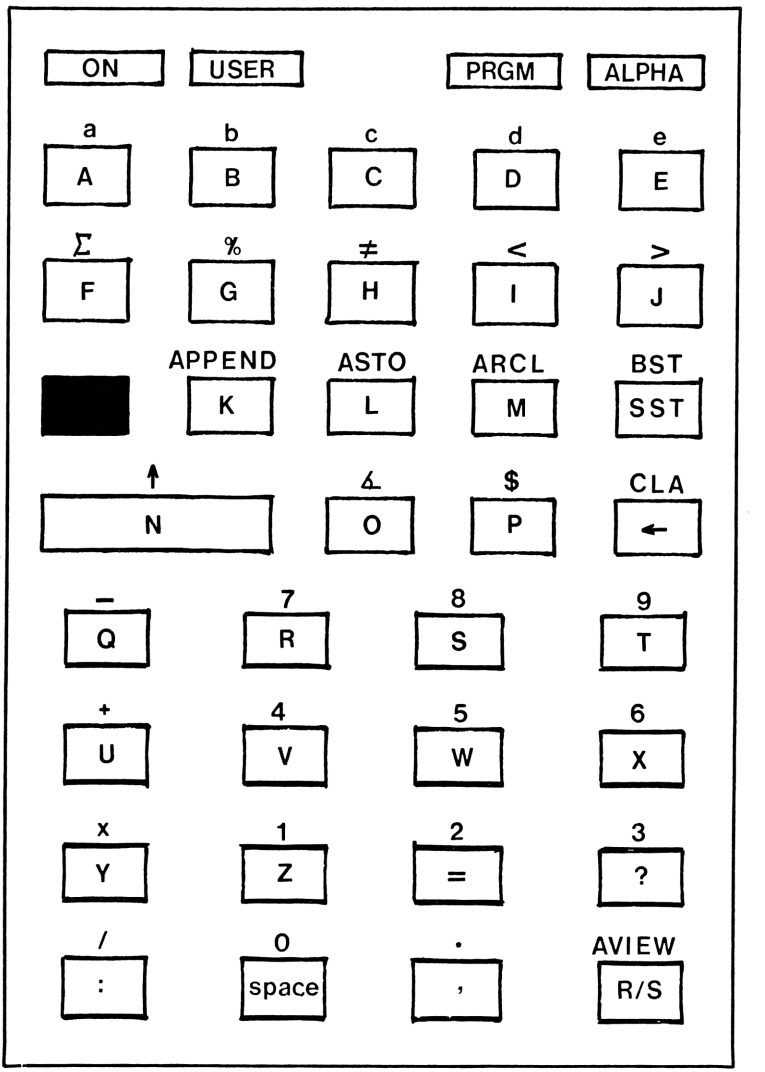


Fig. 3-7. The Alpha keyboard of the HP-41C.

Row 1

Key 1, normal mode: **Σ +**. This key performs some fairly complicated functions by accumulating statistical data into a special register.

Key 1, shift mode: **Σ −**. Performs corrective subtractions from statistics accumulations.

Key 1, alpha mode: **A.** The uppercase letter A.

Key 1, alpha shift mode: **a.** The lowercase letter a.

Key 2, normal mode: **1/x.** Changes the displayed value into its reciprocal value. For example, where 1 mile equals 1.6093 kilometers, 1 kilometer equals 0.6214 miles, 1.6093 kilometers and 0.6214 miles are reciprocals of one another. If you multiply any number by one figure or divide by its reciprocal you arrive at the same result: 25 × 1.6093 = 40.2325 and 25 ÷ 0.6214 = 40.2317 (the difference in fractions results from rounding-off).

Key 2, shift mode: $\mathbf{y^x}$. Raises a number to a given power. For instance, if you want to know what 13^{23} stands for, enter **13** and move it to the y-register by pressing **ENTER** . Enter **23** and press $\mathbf{y^x}$. The display shows the result: *4.1754 25* which represents an endless series of digits equalling 4.1754×10^{25}.

Key 2, alpha mode: **B.** The uppercase letter B.

Key 2, alpha shift mode: **b.** The lowercase letter b.

Key 3, normal mode: $\sqrt{\mathbf{x}}$. The square root of the displayed value. Example: **144** $\sqrt{\mathbf{x}}$ produces *12.*

Key 3, shift mode: $\mathbf{x^2}$. The square of the displayed value. Example: **12** $\mathbf{x^2}$ produces *144.*

Key 3, alpha mode: **C.** The uppercase letter C.

Key 3, alpha shift mode: **c.** The lowercase letter c.

Key 4, normal mode: **LOG.** Displays the common logarithm of the displayed number. Example: **12 LOG** *1.0792.*

Key 4, shift mode: $\mathbf{10^x}$. Produces the common antilogarithm of the displayed number. Example: **1.0792** $\mathbf{10^x}$ results in *12.*

Key 4, alpha mode: **D.** The uppercase letter D.

Key 4, alpha shift mode: **d.** The lowercase letter d.

Key 5, normal mode: **LN.** Produces the natural logarithm of the displayed number. Example: **12 LN** results in *2.4849.*

Key 5, shift mode: $\mathbf{e^x}$. Produces the natural antilogarithm of the displayed number. Example: **2.4849** $\mathbf{e^x}$ results in *12.*

Key 5, alpha mode: **E.** The uppercase letter E.

Key 5, alpha shift mode: **e.** The lowercase letter e.

Row 2

Key 1, normal mode: **x ≷ y.** Pressing this key exchanges the contents of the X register with those of the Y register.

Key 1, shift mode: **CLΣ.** This key clears previously accumulated data from the statistic register.

Key 1, alpha mode: **F.** The uppercase letter F.

Key 1, alpha shift mode: **Σ**. Statistics symbol, to be used in conjunction with other alpha characters as a function command.

Key 2, normal mode: **R↓**. Roll down rolls the registers down one at a time to permit the operator to view the contents of each, or to prepare the contents of any non-x register to be used in a subsequent calculation.

Key 2, shift mode: **%**. Computes percentages. Example: **140 ENTER 25 %** produces: *35*, which means that 25 percent of 140 equals 35.

Key 2, alpha mode: **G.** The uppercase letter G.

Key 2, alpha shift mode: **%**. While this looks the same as key 2, shift mode, this puts the % symbol into the display. It does not execute its function.

Key 3, normal mode: **SIN.** Displays the sine of the previously displayed number. Example: **100 SIN** results in *0.9848*.

Key 3, shift mode: **SIN^{-1}**. Produces the arc sine of the displayed number. Example: **0.9848 SIN^{-1}** results in *80*.

Key 3, alpha mode: **H.** The uppercase letter H.

Key 3, alpha shift mode: **=**. Means not equal to, and can be used in programming instructions by entering **X ≠ Y.** This directs the computer to perform the next function if the contents of the X register are not equal to those in the Y register or to skip that function and go directly to the next.

Key 4, normal mode: **COS.** The cosine of the displayed number. Example: **100 COS** results in *−0.1736*.

Key 4, shift mode: **COS^{-1}**. The arc cosine of the displayed number. Example: **−0.1736 COS^{-1}** results in *100*.

Key 4, alpha mode: **I.** The uppercase letter I.

Key 4, alpha shift mode: **<**. Smaller than; used as a decision-making tool by the computer. If X < Y, then do Z.

Key 5, normal mode: **TAN.** The tangent of the displayed number. Example: **100 TAN** results in *−5.6713*.

Key 5, shift mode: **TAN^{-1}**. The arc tangent of the displayed number. Example: **−5.6713 TAN^{-1}** results in *−80*.

Key 5, alpha mode: **J.** The uppercase letter J.

Key 5, alpha shift mode: **>**. Larger than; see explanation under Key 4, alpha shift mode.

Row 3

Key 1: the **SHIFT** key. This key is the only key with only one function. It is colored orange. Pressing it changes the function of any key from the primary to the secondary.

Key 2, normal mode: **XEQ.** Pressing this key commands the computer to execute the command which follows. It is nearly always used in conjunction with an alpha command, such as **XEQ ALPHA P S E ALPHA**, which commands the computer to insert a pause in the program to permit the operator to read what is on the display.

Key 2, shift mode: **ASN.** This key is used in conjunction with others, usually alpha characters, to assign a special function to another key. Example: **shift ASN ALPHA A L T ALPHA LN** will assign the function **XEQ ALPHA A L T ALPHA** to the **LN** key. Until the key assignment is erased or replaced, pressing **LN** with the computer in user mode will call up the program given the identification label **ALT.**

Key 2, alpha mode: **K.** The uppercase letter K.

Key 2, alpha shift mode: **APPEND.** This function is used in order to add alpha characters to other, previously entered alpha characters. Example: DIST is on the display. With the computer in alpha mode, press **shift APPEND**, and a short line will be added to the display: *DIST* -, which indicates that the computer is waiting for added input. Enter **A N C E** and the complete word *DISTANCE* is on display.

Key 3, normal mode: **STO.** This key commands the computer to store the displayed digits (not letters) in a memory register. The key must always be used in conjunction with two digits to identify the register number. Example: **100 STO 01** stores 100 in the 01 register.

Key 3, shift mode: **LBL.** Stands for label, and is used with two digits or one or more alpha characters to label a program or a section of a program. Labels help the computer locate specific locations within the stored information.

Key 3, alpha mode: **L.** The upper case letter L.

Key 3, alpha shift mode: **ASTO.** This is the same as **STO** except it is used with the computer in alpha mode and stores alpha characters in memory registers. Example: With the computer in alpha mode, press **shift ASTO 00** and the displayed alpha characters will be stored in register 00. Only six alpha characters can be stored in one memory position.

Key 4, normal mode: **RCL.** This key recalls data from memory positions. It must be used in conjunction with two digits to identify the memory being recalled. Example: **RCL 01** will produce *100* (previously stored in that memory position). With the computer **not** in alpha mode, press **RCL 00** and the display will produce *DISTAN*, part of the word distance previously stored in register 00.

Key 4, shift mode: **GTO.** Go to; this key must be used in conjunction with two digits or one or more alpha characters. It directs the computer to search for a given label or program line. It can also be used to edit a program, to place a given program line into display. In that case it must be followed by pressing the decimal point once, and then entering three digits. Example: **GTO . 003** will place the third line of the program into the display as soon as the **PRGM** key is pressed. Pressing the **GTO** key followed by two decimal points performs a compressing action to push all program components as close together as possible in order to leave a maximum amount of space for additional programs or program steps. Example: **GTO . .** will result in the display word *PACKING*. This can take several seconds, after which the data originally on the display will reappear.

Key 4, alpha mode: **M.** The uppercase letter M.

Key 4, alpha shift mode: **ARCL.** This is the same as **RCL**, but is used in alpha mode to recall characters stored in alpha mode. It is always used in conjunction with two digits. Example: **ARCL 00** displays *DISTAN* - .

Key 5, normal mode: **SST.** This key, used in the **PRGM** mode, moves the program one line forward, so the operater may single-step (SST) through a program. When not in program mode, it performs one program step at a time, first displaying the function commanded by the step and then the result of the execution of this function. To function in this manner the key must be held down for a fraction of a second. Letting go of it too quickly produces no result at all, and holding it down too long produces the display *NULL*, to indicate that the keystroke was cancelled and nothing has taken place.

Key 5, shift mode: **BST.** This function is the same as SST except in reverse. It moves the program backwards one step at a time. This key cannot be used to negate a previously performed function.

Key 5, alpha mode: **SST.** Same as when not in alpha mode.

Key 5, alpha shift mode: **BST.** Same as when not in alpha mode.

Row 4

Key 1, normal mode: **ENTER ↑**. This key performs several identical tasks. In ordinary calculations it is used after the first value has been put in display. Example: **2 ENTER↑ 2 +** results in *4* in the display. When used as a program step, it moves the contents of the X register into the Y register, usually to facilitate subsequent decision-making on the part of the computer.

Key 1, shift mode: **CATALOG.** Pressing this key followed by **1, 2,** or **3** displays a list of what is contained in the computer. **CATALOG 1** results in a sequential display of all program and program-section labels which have been stored by the user. **CATALOG 2** presents all commands that are part of the permanent memory of either the card reader or the printer or both (if they are installed), and which can be called up for execution by the operator. **CATALOG 3** presents all the standard functions available with the HP-41. The display goes so fast it is virtually impossible to see each as it flashes on. The speed can be reduced by holding down the **SST** key, making it possible to read each function.

Key 1, alpha mode: **N.** The uppercase letter N.

Key 1, alpha shift mode: **↑**. Moves the contents of the X register up into the Y register.

Key 2, normal mode: **CHS.** Changes the sign of the displayed number. Example: **100 CHS** results in *−100.*

Key 2, shift mode: **ISG.** This stands for Increment and Skip if Greater. This key is used in controlled looping operations.

Key 2, alpha mode: **O.** The uppercase letter O.

Key 2, alpha shift mode: **∠**. Places a symbol for an angle into the display.

Key 3, normal mode: **EEX.** Used to enter numbers with powers of 10. Example: **50 EEX 12** will result in *58 12* which, in fact, represents 58×10^{12}.

Key 3, shift mode: **RTN.** A command placed on the end of a subroutine, telling the computer to go back to the step in the program following the one which commanded it to detour to the subroutine.

Key 3, alpha mode: **P.** The uppercase letter P.

Key 3, alpha shift mode: **$.** Places a dollar sign into the display.

Key 4, normal mode: ←. Pressing this key erases the last character that was placed in display. Example *113.78* is on the display; press ← and the display changes to *113.7* - .

Key 4, shift mode: **CLX/A.** Clears the display. In normal mode the display changes to zeros. In alpha mode the display blanks out completely, except for the applicable annunciators.

Key 4, alpha mode: ←. This function is the same as in regular mode.

Key 4, alpha shift mode: **CLA.** Clears the display.

Row 5

Key 1, normal mode: **-.** Command to subtract. Example: **5 ENTER↑ 6 -** results in *−1.*

Key 1, shift mode: **x=y?.** Asks the computer whether the value in the X register is equal to that in the Y register. If the answer is yes, the computer will execute the next program line. If the answer is no, the computer will skip that line and go to the following line. Example: 10 is in the Y register and you have put 0 into the X register. They are not equal, so the computer will skip the next program line. If you had put 10 into the X register, the next program line would have been executed. This and the other decision-making keys are important in the formulation of advanced programs.

Key 1, alpha mode: **Q.** The uppercase letter Q.

Key 1, alpha shift mode: **−.** Places the minus symbol into the display.

Key 2, normal mode: **7.** The digit 7.

Key 2, shift mode: **SF.** Stands for Set Flag. Flags permit the computer to continue on either way at a fork in the program. Flags are numbered and any flag-related command calls for two digits, such as 01 for flag number 1. There are 10 general-purpose flags and 10 special-purpose flags as well as flags with predetermined meanings. When flags 00, 01, 02, 03, or 04 are set, the fact is shown by a tiny 0, 1, 2, 3, or 4 at the bottom of the display window.

Key 2, alpha mode: **R.** The uppercase letter R.

Key 2, alpha shift mode: **7.** Places the digit 7 into the display.

Key 3, normal mode: **8.** The digit 8.

Key 3, shift mode: **CF.** Stands for Clear Flag, and must be used with two digits to identify the flag which it is supposed to remove.

Key 3, alpha mode: **S.** The uppercase letter S.

Key 3, alpha shift mode: **8.** Places the digit 8 into the display.

Key 4, normal mode: **9.** The digit 9.

Key 4, shift mode: **FS?**. Stands for Flag Set? This is one key which permits the computer to make decisions. In this case, if a given flag is set, the next program step is executed. If it is not, that step is skipped. The key must be used in with two digits to identify which flag it refers to.

Key 4, alpha mode: **T.** The uppercase letter T.

Key 4, alpha shift mode: **9.** Places the digit 9 into display.

Row 6

Key 1, normal mode: **+**. The command to perform an addition.

Key 1, shift mode: **x≤y?**. Asks whether the contents of register X are equal to or smaller than those in register Y. If yes, the next program step is executed. If no, that step is skipped.

Key 1, alpha mode: **U.** The uppercase letter U.

Key 1, alpha shift mode: **+**. Places the plus symbol into the display.

Key 2, normal mode: **4.** The digit 4.

Key 2, shift mode: **BEEP.** Activates a beeping sound which can be programmed, to draw attention to a certain display or displayed result.

Key 2, alpha mode: **V.** The uppercase letter V.

Key 2, alpha shift mode: **4.** Places the digit 4 into the display.

Key 3, normal mode: **5.** The digit 5.

Key 3, shift mode: **P→R.** Polar to rectangular conversion, used in dealing with angles and degrees.

Key 3, alpha mode: **W.** The uppercase letter W.

Key 3, alpha shift mode: **5.** Places the digit 5 into the display.

Key 4, normal mode: **6.** The digit 6.

Key 4, shift mode: **R→P.** Rectangual to polar conversion.

Key 4, alpha mode: **X.** The uppercase letter X.

Key 4, alpha shift mode: **6.** Places the digit 6 into the display.

Row 7

Key 1, normal mode: **x.** The command to multiply.

Key 1, shift mode: **x>y?**. Asks whether the contents of the X register are greater than those of the Y register. If yes, the computer executes the next program step. If no, it skips that step.

Key 1, alpha mode: **Y.** The uppercase letter Y.

Key 1, alpha shift mode: **×**. Places the multiplication symbol into the display.

Key 2, normal mode: **1.** The digit 1.

Key 2, shift mode: **FIX.** This key, when used in conjunction with a single digit, limits the number of decimal positions which are in display. Example: after pressing **FIX 4** the number *12.34567890*, would appear as *12.3457* and after pressing **FIX 0** it would be *12.*

Key 2, alpha mode: **Z.** The uppercase letter Z.

Key 2, alpha shift mode: **1.** Places the digit 1 into the display.

Key 3, normal mode: **2.** The digit 2.

Key 3, shift mode: **SCI.** Changes the display to scientific notation. For instance, the figure *321795643* would change to *3.218 08* if **SCI 3** is pressed while it was in display. The key is always used with one digit which controls the number of displayed digits after the decimal point.

Key 1, alpha mode: **:**. Places a colon into the display.

Key 1, alpha shift mode: **÷**. Places the symbol for divide into the display, but the display will look like this: /.

Key 2, normal mode: **0.** The digit 0.

Key 2, shift mode: π. Pi; places *3.141592654* into the display.

Key 2, alpha mode: **SPACE.** Provides the capability to put a space between two words in display.

Key 2, alpha shift mode: **0.** Places the digit 0 into the display.

Key 3, normal mode: **.**. The decimal point.

Key 3, shift mode: **LAST X.** Places the contents of the last X register into the display.

Key 3, alpha mode: **,**. Places a comma into the display.

Key 3, alpha shift mode: **.**. Places a period into display. Note: periods, commas and colons, when used in an alpha display, do not reduce the number of spaces available for letters, as they are placed between the letter spaces.

Key 4, normal mode: **R/S.** Run/stop. This key is used to stop the execution of a program, and then to start it again.

Key 4, shift mode: **VIEW.** Tells the computer that the operator wants to view the data or alpha characters stored in a given register. The key is always used in conjunction with two digits to identify the correct register.

Key 3, alpha mode: **=**. An equal sign is placed into the display. The principle of reverse Polish notation means that in calculating activity no equal sign is used. The purpose of this equal sign in the alpha mode is to be able to use it in such displays as *NM=176.5*, which would be a displayed result of a calculation.

Key 3, alpha shift mode: **2.** Places the digit 2 into the display.

Key 4, normal mode: **3.** The digit 3.

Key 4, shift mode: **ENG.** Like SCI, this key changes a displayed value to engineering notation. The primary difference is that in scientific notation there is one digit ahead of the decimal point while in engineering notation there are two digits ahead of the decimal point.

Key 4, alpha mode: **?**. Places a question mark into the display.

Key 4, alpha shift mode: **3.** Places the digit 3 into the display.

Row 8

Key 1, normal mode: **÷**. The command to divide.

Key 1, shift mode: **x=0?**. Asks if the contents of the X register equals zero. If so, the next program step is executed. If not, it skips that step.

Key 4, alpha mode: **R/S.** Same as regular mode.

Key 4, alpha shift mode: **AVIEW.** Used to view alpha data in the

display; usually followed by **XEQ ALPHA P S E ALPHA**, which inserts a pause and holds the display for about one second before continuing to the next program step.

XEQ ALPHA Function

In addition to the functions which can be called up by pressing a key or a combination of keys, the HP-41 contains a variety of preprogrammed functions which can be called up by pressing **XEQ ALPHA** followed by the letter identifier for the function in question. In each instance the required keystroke sequence is **XEQ ALPHA** function identifier **ALPHA** (though any function used with great frequency can be assigned to a key—see the meaning of the ASN key above—thus reducing the number of required key strokes). The following is a list of these functions in alphabetical order:

ABS. Absolute value always presents a plus value. Thus, **–54 ABS** will result in 54, but **54 ABS** remains 54. (Unlike the **CHS** function which would change 54 to –54.)
ADV. Advances the paper if the printer is connected to the computer.
ASHF. Shifts the characters in an alpha string six characters to the left.
CLD. Clears the display.
CLP. Clear program; clears a program from the computer memory when followed by that program's identification label.
CLRG. Clears all registers of stored data. This command must be used with discretion, because it clears not only the registers being used by a specific program but **all** registers.
CLST. Clears each of the automatic memory stacks to zero.
COPY. Used to copy into computer memory a program contained in one of the Application Pacs offered by Hewlett A. Packard. This is useful for changing or modifying the prerecorded programs.
D - R. Converts degree mode to radian mode.
DEC. Changes an octal value in display to a decimal value.
DEG. Places the computer in the degree mode.
DEL. When used in conjunction with three digits, it causes specific program lines to be deleted.
DSE. Decrement and skip if equal; this is the reverse of the keyboard function **ISG.**
END. Indicates end of program.
FACT. Factorial; used to handle combinations and permutations.
FC?. Flag Clear?, requiring two digits to identify the flag in question.
FC?C. Flag Clear?, if not, Clear it. Also needs a two-digit identification.
FRC. Eliminates the integer and retains the fractional portion of a displayed value. Example: **2.456 FRC** results in *.456.*
FS?C. Flag Set?, test and then Clear. This also needs a two-digit identification.
GRAD. Places the computer into the grad mode.
HMS. Changes time in decimal format to time in hours, minutes and

seconds. Example: **11.789 HMS** results in *11.4720*, representing 11 hours, 47 minutes and 20 seconds (also applies to degrees).
HMS+. Permits addition with time or degree values without first converting them to decimal format.
HMS–. Same as above, in terms of subtraction.
HR. Converts hours, minutes, and seconds into their decimal equivalent. Example: **11.4720 HR** results in *11.7890.*
INT. Removes the fractions and retains the integer portion of a displayed value. Example: **12.3456 INT** results in *12.*
MEAN. Calculates the arithmetic average of the values accumulated in the statistical registers.
OCT. Converts decimal values to octal values.
PACK. A command for the computer to pack programs into the smallest possible space; the same as pressing the **GTO . .** keys.
%CH. Calculates the percent of change between two values, usually based on elapsed time.
PROMPT. Causes a display, usually of alpha characters, to remain in display until the operator has entered the requested data. When data has been entered, **R/S** must be pressed in order to continue the program.
PSE. Inserts a pause of approximately one second duration into the program to permit the operator to read what is in the display.
R - D. Converts from radians to degrees.
RAD. Places the computer in radians mode.
RND. Rounds a number, resulting in the number appearing in the display. Example: 1.234567890 is retained in the computer even though by using **FIX 2** the displayed number is only 1.23. Using the **RND** function causes 1.23 to be the total number retained in and used by the computer.
SDEV. Standard deviation; calculates the standard deviation of data stored or accumulated in the statistical registers.
SIZE. This is used to determine the register configuration and allocation for a program. It requires a three-digit input representing the number of registers used for storage and retrieval of data. Thus, if the computer has been partitioned by the command **SIZE 040**, and the program calls for **STO 41**, the display will read "nonexistent."
TONE. When used in conjunction with two digits, produces a variety of musical tones.
X=0?. If the value in the X register is not equal to zero, execute the next line; otherwise, skip that line.
X<0?. If the value in the X register is smaller than zero, execute the next line, otherwise skip it.
X<=0?. If the value in the X register is smaller than or equal to zero, execute the next line, otherwise skip it.
X>0?. If the value in the X register is greater than zero, execute the next line, otherwise skip it.
X<>. Used with a two-digit identification, it exchanges the contents of the X register with those of any other register.

Chapter 4
Anatomy of a Computer Program

In its simplest form, a computer program is any function or series of functions that a computer will repeat on command. Thus Fig. 4-1 could be described as a program. This is how it works: Enter any number into the computer and then, by pressing one execution key instead of the three keys represented by the program, the computer will add 2 to that number and display the result. The only difference between this program and one involving many steps is the amount of time it would take manually to perform the required key strokes each time the program is run with a different variable or set of variables.

A computer program uses fixed values, stored in its registers, when making computations based on inputted variables. Thus, the addition function and the stored value 2 in our simple program in Fig. 4-1 are fixed values. The number we key into the computer to add to the stored value 2 is the variable.

The primary reason to use computers is that they can compare an unlimited number of variables with a series of fixed values and display the result tirelessly and quickly. Let's say that you want to know how long it is going to take your car to cover a given distance at different speeds. The distance to be covered is 100 miles and the speeds involved range from five miles per hour, in five-mph increments, up to 100 mph.

The program is shown in Fig. 4-2. It represents "100 divided by variable equals." If you punch in 5, 10, 15, and so on, within seconds you will obtain the complete series of results given in terms of hours and decimal parts of hours. If you want the results in terms of hours and minutes or hours, minutes, and seconds, the computer has a built-in program for that conversion. If the command to convert is included in the program steps, it will provide the appropriate presentation (Table 4-1).

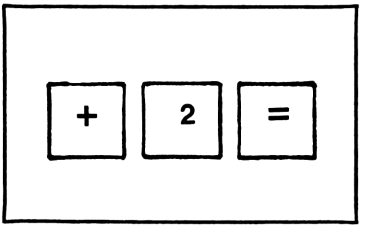


Fig. 4-1. The minimum ingredients of a program: A fixed value and the command to relate it to a variable value.

This may appear to be a useless exercise; it is used here as a means of making explanations as simple and basic as possible. A practical application for the Time Value program would determine the best compromise between fuel consumption and time—the speed at which the combination of fuel burned and time spent produces the most desirable result. If, based on experiments or manufacturer's claims, the actual amount of fuel consumed per hour for each speed is known, by inputting those rates of consumption and adding steps we can arrive at the most economical speed (which may not be 55 mph). The additional step is to multiply the hours (in decimal format, not hours:minutes:seconds) with the rate of fuel consumption.

Once we have a basic program, adding a few steps can change the significance of the amount or type of information it provides, and soon we have a program that will give us answers to some intricate questions.

That brings me to a prerequisite of successful program writing: Before you start to create a program, it is extremely important that you have a clear idea in your mind what each step along the way should accomplish. You'd be surprised how easy it is to overlook a simple but necessary step and to place commands in an incorrect order.

There is a well-used expression in computer circles: "garbage in, garbage out" or GIGO. The computer will accept what you tell it and will use your input to perform its functions. Thus, if you input inaccurate, information that inaccuracy is likely to be blown quite out of proportion by the time the computer gets through with its task.

Work your program out on paper first, making sure each step is where it belongs and that you know with certainty what the execution of that step will produce. Flowcharts and other means of preparing a program on paper are discussed in the next section. A complete understanding and analysis of each program step and section is the foundation on which successful program creation can be built.

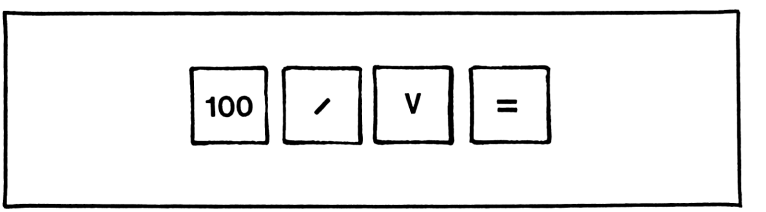


Fig. 4-2. The fixed value, 100, is to be divided by V, the variable value.

Table 4-1. Time Value Program Table.

Speed (mph)	Time (decimals)	Time (H:MM:SS)
5	20.0	20:00:00
10	10.0	10:00:00
15	6.666667	6:40:00
20	5.0	5:00:00
25	4.0	4:00:00
30	3.333333	3:20:00
35	2.857142	2:51:26
40	2.5	2:30:00
45	2.222222	2:13:20
50	2.0	2:00:00
55	1.818182	1:49:05
60	1.666667	1:40:00
65	1.538462	1:32:18
70	1.428571	1:25:43
75	1.333333	1:20:00
80	1.25	1:15:00
85	1.176471	1:10:35
90	1.111111	1:06:40
95	1.052632	1:03:09
100	1.0	1:00:00

PROGRAM PREPARATION: BEYOND FLOWCHARTS

Before writing a program of any degree of complexity, you should first take a pencil and piece of paper and break the program into its components. Basically, every program can be thought of as consisting of three main segments: one to enter data into the computer; one to perform the computations; and one to display the result or results. The first and third segments are usually quite simple and straightforward; the middle portion gets complicated.

Most books on programming urge you to prepare flowcharts so you can visualize the program. Personally, I think that's a lot of nonsense. It often takes more time and effort to figure out an intelligent way to depict what the program does than to forge ahead by the trial-and-error method. I prefer to divide the program into blocks of related steps and then worry later about how they fit together.

For example, let's take Bricks, one of the less complicated programs, described in Chapter 1 of this book. It consists of a segment that accepts the variable data, and a segment that asks a question. Depending on the answer to that question, the program can go in either of two directions. It then returns to a segment which displays the first results. Following the display, there is a segment to accept data, another that computes the data, and a final segment that displays the final result. Figure 4-3 is a flowchart representation of the program.

My way of accomplishing the same thing would look something like this:

1. Area to be covered, side 1 and side 2, STO 00 and 01.
2. 00 × 01 = square footage; × 144 = square inches, STO 00.
3. Mortar, yes or no.
   - ☐ If yes, each brick = 22.69 sq. in.
   - ☐ If no, each brick = 17.44 sq. in.
4. 00 × 22.69 = STO 01
5. 00 × 17.44 = STO 01

   } Display number of bricks needed.
6. Cost per brick, STO 02
7. 01 × 02 (FIX 2) = STO 03. Display total cost.

This way of outlining the program presents it in plain English, and you don't have to figure out what all those boxes mean. If you want to go one

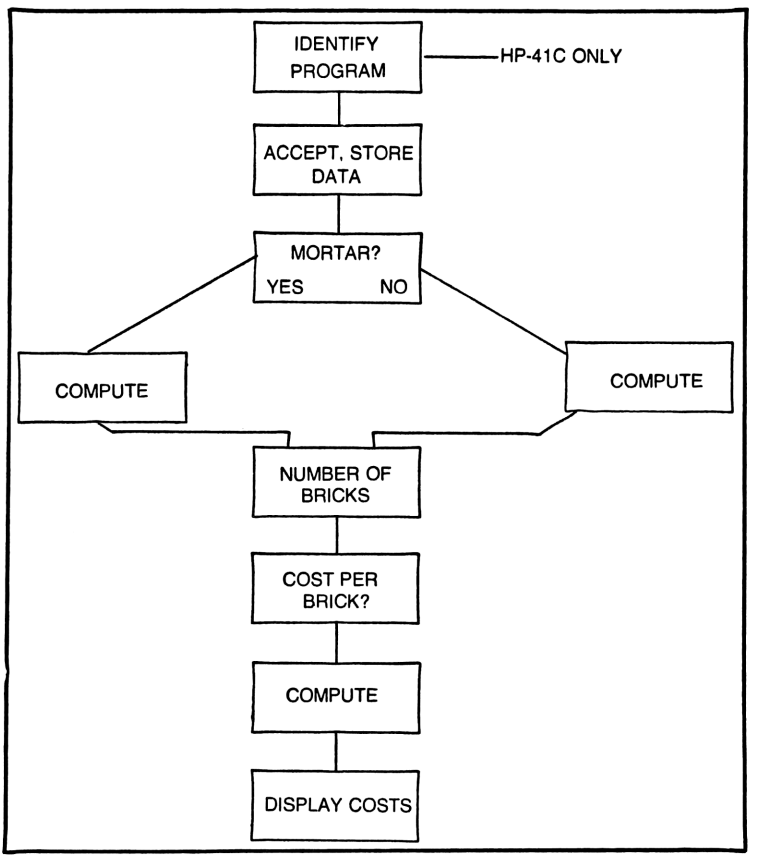


Fig. 4-3. A simple example of a flowchart, this one for the program How Many Bricks in a Brick Wall?

step further, you could list the consecutive order in which the different steps are used:

$$A - B - C - \left\{ \begin{matrix} D \\ E \end{matrix} \right\} - F - G$$

Let's translate this outline into the actual program in two versions, one for each computer. The programs are shown in Figs. 4-4 and 4-5.

If you compare Fig. 4-3 with the computer printouts in Figs. 7-11 and 7-12, you will notice they are not the same. In nearly all instances, there are always different ways to design a program to arrive at the same result.

When you are working on a program, especially if it looks as if it is going to be lengthy, make notes of the memory positions used and the purpose each is used for, and also keep track of labels and of what is accomplished in each program section headed by a label.

Every programmer has found himself using high numbers for memory positions because he couldn't remember how many had already been used. Equally often, sections are labelled **LBL Z** or **LBL X** because no list was kept of the letters and numbers used for labels early in the program.

Keeping program records is not only helpful while you are designing and writing the program, it will also prove invaluable during the editing process. Editing is the time when we discover that the results produced by the program are obviously wrong, and we have to find out where we made a mistake: the computer never goofs.

PARTITIONING YOUR COMPUTER

I find the instruction books for both computers do a very poor job explaining partitioning and why it is important. To start with, think of the inside of the computer as one of those hotel meeting rooms that has a dividing wall which can be placed anywhere to create two rooms of any size. That is, more or less, the principle of partitioning.

To carry the room example one step further, imagine that one of the rooms is used to store things, while the other is used for meetings and other activities. Obviously, if you have much to be stored, but only a small group to listen to a lecture, you would use a large room for storage and a small one for activities. If, on the other hand, the audience for the speech is quite large, you'd need the large room for the activities and there wouldn't be as much space available for storage.

To relate this to the computer, pretend there are a certain number of square feet of space which can be divided to suit our needs. A certain number of feet will be needed to store data in memory positions. But whatever is left over can be used to hold program steps.

All programs consist of a number of program steps designed to manipulate data stored in memory positions. Since the type of computers we are discussing in this book are not particularly well suited for data storage, the

TI-59

```
LBL     1
A       7
STO     .
00      4
R/S     4
LBL     =
B       STO
STO     01
01      LBL
R/S     SUM
LBL     RCL
C       01
STO     R/S
02      x
R/S     RCL
LBL     02
D       =
RCL     STO
00      03
x       FIX
RCL     2
01      R/S
x       LBL
1       E
4       RCL
4       00
=       x
STO     RCL
00      01
LBL     x
EE      1
/       4
2       4
2       =
.       STO
6       00
9       GTO
=       INV
STO
01
GTO
SUM
LBL
INV
RCL
00
/
```

To use the program, enter the lengths of sides 1 and 2, pressing **A** after 1, and **B** after 2. Then enter the cost per brick and press **C**. Then, if mortar is to be used, press **D** for a display of the number of bricks, and **R/S** for the cost. If mortar is not to be used, press **E** for the number of bricks and then **R/S** for the cost.

Fig. 4-4. Bricks for the TI-59 from the author's outline.

HP-41C

```
LBL BRICKS
SIDE 1?
PROMPT
STO 00
SIDE 2?
PROMPT
STO 01
$ PER BRICK?
PROMPT
STO 02
RCL 00
RCL 01
x
144
x
STO 00
MORTAR? 0=NO
PROMPT
X=0
GTO 01
RCL 00
22.69
/
STO 01
GTO 03
LBL 01
RCL 00
17.44
/
STO 01
LBL 03
NO.BRICKS=
ARCL 01
AVIEW
STOP
RCL 02
x
STO 03
COST=$
ARCL 03
A VIEW
STOP
END
```

Aside from having to remember how to get the program started (by pressing **XEQ ALPHA BRICKS ALPHA**, or by assigning the word BRICKS to a user-defined key) the user doesn't need to remember what data needs to be entered at any point in the program, because the display will prompt him when it is needed. When the results are being displayed, the alpha portion of the display explains what the figures represent.

Fig. 4-5. Bricks for the HP-41C from the author's outline.

number of spaces required for program steps is usually far greater than that needed for data storage. The two computers treat partitioning differently, so let's take one at a time.

TI-59

When the TI-59 is turned on, it is automatically partitioned to accommodate 480 program steps (numbers 000 through 479) and has room for 60 memory-storage registers (00 through 59). This partitioning is satisfactory for most purposes, but there are times when we need more room for program steps. There may be other times when we need a lot of room to store data, but don't need too many program steps. It is possible to repartition for up to 100 available data-storage positions, leaving room for only 160 program steps. Conversely, we can make room for up to 960 program steps, leaving no room at all for data storage.

The TI-59 can only be partitioned in increments of 10 data-storage positions, equal to 80 program steps. Table 4-2 lists the possible choices:

Partitioning uses the **2nd Op** capability of the computer, and is described in detail on page V-22 of the personal programming guide published by Texas Instruments and provided to all purchasers of the TI-59.

With the TI, the standard partitioning will be sufficient except under exceptional conditions. If we realize while writing a program that we're going to need more room for program steps press **LRN** to take the computer out of the learn mode, perform the partitioning series of keystrokes (**3 2nd Op 1 7** or any other appropriate first digit) and the computer will repartition without interfering with the program part that has already been written. Simply press **LRN** to continue writing the program.

Be aware that if the TI has been repartitioned in a non-standard way while writing a program which is then recorded on a magnetic card, the computer later will not accept that magnetic card.

HP-41C

The HP-41C instruction book doesn't talk about partitioning. It is referred to as sizing, an equivalent function, though it is done quite differently. The TI has its number of registers fixed and cannot be increased, this is not the case with the HP. The HP programmable calculator comes in two versions, the HP-41C and the HP-41CV. The difference between the two (aside from price) is the number of available registers. The HP-41C has only 63 registers available, which can accommodate 445 bytes. The HP-41CV has 319 registers available, accommodating 2,237 bytes. Plug-in modules are available for the HP-41C, each of which has 64 registers; four plug-in modules will upgrade the HP-41C to the capability of the HP-41CV.

The HP-41C (or CV) can be partitioned in any way the user chooses. It is partitioned by pressing **XEQ ALPHA S I Z E ALPHA,** followed by three digits which represent the number of data-storage registers needed, leaving the balance for program steps. If there are no plug-in memories in

Table 4-2. Possible Partitioning Combinations.

Program Steps	Data-Storage Positions
960	0
880	10
800	20
720	30
640	40
560	50
480 (normal)	60 (normal)
400	70
320	80
240	90
160	100

your computer and if you follow the partition command with **001,** pressing **PRGM** will display *00 REG 62,* indicating 62 positions available for programming. If you follow those key strokes with **062** and then press **PRGM,** the display will read *00 REG 01:* space for only one step is available.

Every program uses a certain amount of memory, but it is foolish to reserve an unnecessarily large portion of the registers for that purpose, as it severely limits the number of programs (or program steps) the computer can accommodate. Every program has its size. If a program uses only positions R_{00}, R_{01}, and R_{02}, it is a size 003 program.

Before starting to write a program we could estimate the number of memory positions which may be needed. It isn't necessary to do this, however, because the computer will accept instructions involving registers for which it is not partitioned. It is possible to write a whole program without partitioning the computer in any way. If we then try to run the program, as soon as the program calls for a **STO** or **RCL** or any other instruction involving a memory position, the display will read *NONEXISTENT,* reminding us we forgot to start off with the appropriate size command.

You may change the size command while running a program if a mistake has been made in the original sizing. Simply punch in **XEQ ALPHA S I Z E ALPHA** plus 3 digits and after changing the size, the computer will return to the step in the program where the mistake was discovered.

It is always best to use the smallest size number possible with the HP in order to have plenty of room for programming.

USER-DEFINED LABELS

Computers which can accept user-defined labels give the user the ability to change the meaning of almost any key on the keyboard to some

function the user would like to be able to command with a single key stroke or program step. The user can give a special assignment to a key, an assignment which need have no relationship to the function printed on or above the key.

The TI-59 system of dealing with user-defined keys and labels is simple. The computer has five keys, or ten potential functions, designed to be used exclusively as user-defined keys or labels. They are the primary functions A, B, C, D and E and the secondary functions A′, B′, C′, D′, and E′. These functions can be used anywhere in the program as labels by inserting **Lbl A** at the appropriate place in the program. If the computer then encounters an A while running the program, it goes to the place in the program where **Lbl A** is found.

For most applications these ten user-defined key positions are sufficient. Occasionally a situation arises that requires more than ten labels. All keys on the TI-59 keyboard, with the exception of the secondary functions LRN, Ins, Del, SST, BST, Ind and R/S (and, of course, the digits 0 through 9) can be used as a label for a subroutine or a certain program position. Thus, if the computer encounters **GTO 2nd cos,** it will search for, find and then go to the place in the program at which **Lbl 2nd cos** is located.

The HP-41 keyboard includes a key marked **USER.** When that key is pressed, the word user appears in the left-hand bottom section of the display window. Pressing the **USER** key again removes the display and takes the computer out of user mode. In the regular (non-user) mode each key performs the function printed on or above it. In the user mode, any key which has received a special assignment will perform that function, ignoring the legend printed on or above it. Keys without special assignments operate as they would in the normal mode.

Many keystrokes are often needed to cause the HP-41 to perform a given function call. The computer will accept the entire series of keystrokes and combine them into one program step. Thus, the command used to prompt the operator to enter a variable calls for nine key strokes: **XEQ ALPHA P R O M P T ALPHA.** That isn't bad when it happens once or twice in a program, but when a lengthy command is used dozens of times, programming can get tedious.

A simple way around that problem is to press (in any mode, regular, user, program but **not** alpha): **shift ASN ALPHA P R O M P T ALPHA** followed by a key of your choice. The **EEX** key might be a good choice since its alpha meaning is P, and under normal conditions the **EEX** function is rarely used. The display will briefly show: *ASN PROMPT 43,* 43 identifying the assigned key as the third from the left in the fourth row from the top. From now on, pressing **EEX** with the computer in program mode will take the place of those nine key strokes and will produce the display *PROMPT.* This system can be used with any and all existing functions or labels regardless of whether they are part of the HP-41 inventory or have been previously entered by the operator.

It is not possible to assign a label or function to a key when it is not in the HP-41 inventory and when it has not as yet been used by the operator. Thus, even though you may plan to use, say, **LBL AA** later in the program, if you attempt to assign **LBL AA** to a key before it has become part of a program, the display will respond with *NONEXISTENT*.

All keys are available for assignment with the exception of the digit keys, the alpha characters (in the alpha mode) and the **shift, ON, USER, PRGM** and **ALPHA** keys. Whenever possible, use only keys with original functions that won't be used often in your programming while your special key assignment remains active. I once assigned a special meaning to the **STO** key. Afterwards, every time I wanted to store a value in some memory position, the display would read *XEQ 'ST* because I always forgot to take the computer out of user mode before pressing the **STO** key.

Key assignments remain active indefinitely unless they are changed by the operator or the entire memory is cleared and the *MEMORY LOST* condition is called up. Key assignments which have outlived their usefulness can be removed by pressing **shift ASN ALPHA ALPHA,** followed by the key to be cleared. This routine will return the original function to the key.

The ability to assign a customized function to many keys can greatly improve your programming efficiency.

INITIALIZING A PROGRAM

Initializing or starting a program sounds simple. Well, it is and it isn't. With the TI-59, program initialization is not a great problem. Since only one program resides in the computer at a time, there is no danger of calling up the wrong one. Initialization usually is performed by pressing **RST** followed by **R/S,** which will set the computer to the configuration appropriate for the program while concurrently erasing stray data which could foul things up if accidentally used by the program.

Initialization is a different story with the HP-41C, which can and often does contain a large number of programs in its permanent memory. The operator must know what to do to get the program he needs started.

There are two basic ways to start a program. Let's assume the program is labeled *TEST,* meaning the first step in the program is *01 LBL TEST*. To call up this program we can use the following series of key strokes: **XEQ ALPHA T E S T ALPHA,** and the program is now ready to be run. That's a lot of keystrokes to use every time the program is run. A simpler way would be to assign the label TEST to a single key, by pressing **shift ASN ALPHA T E S T ALPHA** and then **LN** (or any other key). With the computer in user mode, pressing **LN** will then ready the program.

One way or another, you must remember which user-defined key operates the program, which identification is used as a label, or preferably both (in case you accidentally change the user-defined key to something else).

If a time comes when you can't remember which key to use and you can't remember the identification for the label, press **shift CATALOG 1** and **SST,** and each label used in any program stored in the computer will appear for a brief moment. This will remind you which label stands for which program. If even this doesn't help you remember, you have no alternative other than to try each label displayed, through the catalog procedure, until you find the program you're looking for.

This is just one more reason the importance of record keeping cannot be overemphasized.

Chapter 5
Record Keeping and Editing

A written record should be kept of any program that has any value at all. Copying it to a magnetic card is not sufficient; cards can be lost, stepped on, and ruined too easily. Designing a complicated program, editing it to make sure it works the way it's supposed to, is a lot of work; you shouldn't put yourself in the position where you might have to do that work all over again.

If you have a printer, the solution is easy: just hook up the printer and let the computer produce its own printed record. If no printer is available, the only alternative is to go through the program step by step and write each step down. It may seem like a lot of work, especially if the program has many hundreds of steps, but it is a necessity.

Having a printed, or hard copy, version of your program is also a tremendous help in editing it. Besides the advantage of being able to look at the entire program all at once, the printers for both computers have a trace mode which traces and prints every step the computer takes when it is running a program. By examining a trace record, whatever mistakes might be hidden are likely to show up.

Editing programs means trying to find the program step or set of steps that appears to be the reason something else in the program isn't functioning the way it should, or simply attempting to improve the working program, or eliminate superfluous program steps which may have been left over from the time the program was still being developed.

Without a written record, editing can consume hours or days just to locate some minor mistake or omission.

One good way to simplify the task is to insert many R/S commands, wherever there is space in the program. This command causes the program to stop and display whatever was last in the X register; we can examine it and make sure that, up to that point, everything is the way it should be.

Then when we press R/S, the computer goes on to the next R/S command where it stops again. Always keep in mind that you may have written commands into the program which cause the computer to jump back and forth within the program. Thus, when it does come to such a stop, look at the actual program step (press **LRN** with the TI-59 or **PRGM** with the HP-41C) to make sure you know what portion of the program you're looking at.

Whenever the editing involves making major changes or devising additions to an existing program, it is always a good idea first to copy the original program on a magnetic card. Then, if something goes haywire in the process of making the change, (and it happens, believe me) at least you won't have to start all over from scratch. The magnetic cards can be used and erased over and over again, so using them at different intermediate stages is not wasteful.

Chapter 6
Peripheral Functions

There is more to programming than punching in steps and running the final product. The TI-59 and HP-41 both have peripheral functions that can make life easier while you write and edit your programs. Two of these are the magnetic cards and calculator functions.

YOUR COMPUTER AS A CALCULATOR

Sometimes, with programs loaded into our computers, we are suddenly confronted with the need to use a calculator to solve mathematical problems. There is no reason why you can't do this on your computer without disturbing the program. Certain precautions may have to be taken, however.

If the program in your computer consists of program steps and data stored in certain memory registers, take care not to use those registers to store any data involved in the calculation you will perform.

Similarly, if the program is in the process of being run and you halt it to perform a calculation, none of the memory registers used by the program when it is being run should be used. If you don't remember which registers are being used, either look at the record of the key strokes which make up the program, or simply pick some registers certain to be unemployed.

This is simple with the HP-41C, as the size designation indicates the memory registers used by the program (size 012 means registers 00 through 11 are used by the program), but you may have to increase the sizing (**XEQ ALPHA S I Z E ALPHA** plus a new three-digit number) to prepare the computer to accept data in other registers. If that is not done, the display will read *NONEXISTANT*.

Since the HP-41C can simultaneously contain dozens of programs it might be difficult to determine which registers are being used and which are

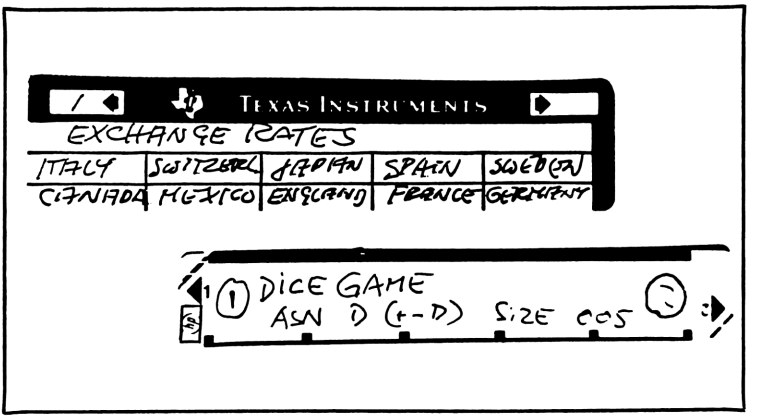


Fig. 6-1. Both computers use magnetic cards for program storage.

not. If no notes are available, one way to get the information is to turn the computer on, key in any number, and then press **STO** and a register number you believe to be free. If the display reads *NONEXISTENT*, the register is free and by executing a new size command the register can be used. Otherwise, you used a register which is part of a program. If it is one being used for permanent data storage, no harm is done. If it is a register used for data storage, the correct data will have to be restored before the program can be run.

There is one more thing to look out for with the HP-41C. Be sure to take the computer out of the user mode before using it as a calculator. If there are any user-defined keys (and there nearly always are), pressing such a key to use it for calculation can bring about some rather strange results, such as a display showing that Program X has inadvertantly been called up and initialized.

Since the TI-59 can hold only one program at a time, it is not too difficult to avoid stepping on any program "toes."

THE MAGNETIC CARDS

Both the TI-59 and the HP-41 use magnetic cards (Fig. 6-1) to permanently store programs and data. They are an important consideration because magnetic cards extend the total capacity of both computers to a point which has no practical limit.

Magnetic cards, like magnetic tape, have certain characteristics demanding special handling or they might lose the information stored on them or otherwise malfunction.

If they are placed close to a permanent magnet or to an appliance containing strong magnets, the information stored on the cards will usually be distorted or lost. On the other hand, I have used an electric typewriter within inches of the cards (the typewriter motor consists of electromagnets) and have had no trouble with the cards in spite of this.

Writing on the cards with most pens will result in ugly smears. Pencil seems to work quite well, but for better, more permanent identification, get a Koh-I-Nor Rapidograph 00 pen and the special ink that must be used with it. Once dry, that ink is indelible and it won't smear.

Texas Instruments and Hewlett-Packard provide pocket-sized card holders for their magnetic cards. Those designed for the HP-41C have slots for the cards which are excessively tight, and I have had to stretch them by inserting some implement which could be used for the purpose. After stretching the cards could be inserted into the slots without the danger of ruining them by bending or scratching.

Chapter 7
Programs

The best, and quite possibly the only way, to learn how to program is to study a variety of other programs. Try to analyze how they perform a given task.

Preparing the following programs, I tried to stick to subjects that will be familiar to you and to stay away from scientific and engineering subjects. Once you thoroughly understand the intricacies of program design, you will be able to write programs within the areas of your specific expertise.

Some of the programs are very simple, designed strictly as a means of demonstration. Others are complicated and lengthy, and in order to understand how they function it is often necessary to break them down into their individual components. In each case I have attempted to describe the programs and their functions so that even the inexperienced layman can follow them after just a bit of practice.

USER INSTRUCTIONS IN BRIEF

Read this section carefully. It will simplify the procedure of running certain programs so you won't have to reread entire sections of program description each time. I have eliminated in each program description those initial steps that are identical in all instances. I have included the partitioning data for the TI and the size data for the HP, as well as the number of magnetic cards and the number of passes through the card reader are required to store each program.

The initial steps which will not be repeated for each set of instructions, are:

TI-59. Feed magnetic program cards through the lower (card reader) slot. After each pass a steady number should be in the display. If it is not, repeat. (If the battery pack needs recharging, the computer may refuse to

read cards.) A flashing 0 or number indicates that the card has not been read satisfactorily. When the program has been accepted by the computer, slide the program card into the upper slot, so the inscriptions on the card can be used as a reminder of what data to enter, when and how. You are then ready to run the program.

HP-41C or CV. After turning the computer on, press **XEQ ALPHA S I Z E ALPHA** followed by a three-digit number representing the size of the program. Then, if the program is not stored in the computer's permanent memory, feed the program cards through the card reader. After the first card the display will prompt you with *RDY 2 OF 6*, or some appropriate message. Programs can be initialized in two ways. One way is to press **XEQ ALPHA A B C** (or whatever is used as the initial label which identifies the program) **ALPHA**, causing the program name to appear in the display. Alternately, a key can be assigned to do the same thing. If such a key assignment has been made (see Chapter 4) only that one key needs to be pressed to initialize the program.

SPEED/TIME/DISTANCE

In Chapter 4 I used a speed/time problem to determine the amount of time it would take to cover a distance of 100 miles at any speed. Now let's write a program for this problem and gradually refine it.

Initially we want to be able to enter a speed figure into the computer and have it return the answer, expressed in terms of hours, minutes and seconds. We must first determine what mathematic steps are involved in calculating time en route based on speed. There is only one step: the distance must be divided by the speed to determine the time. So, the guts of the program will consist of **distance ÷ speed = time.**

TI-59. Since the TI-59 communicates only with digits, we automatically eliminate all steps dealing with program identification and prompting the operator for input.

First, we place the computer into program mode by pressing **LRN**, producing the display: *000 00*. Then press **2nd Lbl A ÷ R/S**. At this point in the program the computer will have received the information about the distance we want to examine. Now press **2nd Lbl B =**, which brings us to the point the result is displayed in terms of hours and decimals of hours. To convert that, press **INV 2nd D.MS** followed by **R/S**, and that is the end of the program. Press **LRN** to take the computer out of program mode (Fig. 7-1).

To run the program, start by entering the distance, **382** and press **A** (Fig. 7-2). Then enter the speed, **65**, and press **B**. The display will show *5.523692308*. There are so many extra digits because we failed to program a fix-decimal limit. Press **LRN** once more to again access the program mode. The display will show *010 00*, meaning that step number 10 is the first step past the end of the program, and it is empty. Use the **BST** key to go backwards. After pressing it a few times we'll see step number six: *006 95*. Since 95 is the equals sign, this would be as good a place as any to insert

TI59	
000 LBL	006 =
001 A	007 FIX
002 /	008 4
003 R/S	009 INV
004 LBL	010 D.MS
005 B	011 R/S

Fig. 7-1. The program steps for the Speed/ Time/ Distance program for the TI-59.

the fix limit. To do this, press **SST** to go one step forward, then press **2nd Ins** which produces the display *007 00*, indicating that the computer has prepared an empty place in the program for a new instruction. Now press **2nd FIX**, followed again by **2nd Ins**, to make room for the added digit. Press **4** and then take the computer out of program mode by pressing **LRN**. That long figure from before may reappear in the display; if it's disconcerting, press **CLR** and the display reverts to *0*. Enter the distance once more, **382** and press **A** and enter the speed, **65**, then press **B**. This time the display reads *5.5237*.

HP-41C. First give the program a label; without it the computer wouldn't know which program to work (assuming there is more than one program in the computer). Press **ON** followed by **PRGM** and the display will read something like *00 REG 32* (or some other number), meaning we have 32 data-storage registers available. Since we don't expect to need more than one storage register we can continue writing the program. Since it is a Speed/Time program, we'll label it ST. Press **shift LBL ALPHA S T ALPHA**. The display will show *01 LBL 'ST*. From now on, until or unless that program is intentionally replaced or removed, asking the computer to execute ST will always start this program (Fig. 7-3).

When we call up the program it would be nice to be reassured we've called the right program. To place an identification into the display (Fig.

TI59	
Press:	Display:
382	382
A	382.0000
65	65
B	5.5237

Fig. 7-2. The keystrokes and displays when running the program in the TI-59.

HP-41 (short version)

```
01 LBL 'ST
02 'DISTANCE ?
03 PROMPT
04 'MILES/HOUR ?
05 PROMPT
06 /
07 HMS
08 STOP
09 GTO 'ST
10 END
```

Fig. 7-3. The short version of the Speed /Time /Distance program for the HP-41C.

7-4), press **ALPHA S P E E D shift / T I M E ALPHA** and the display will show *02' TIME/SPEED*. To command the computer to keep this legend in the display long enough to read (to insert a short pause), press **ALPHA shift A VIEW ALPHA** followed by **XEQ ALPHA P S E ALPHA**. This produces two program steps. One is displayed as *03 AVIEW*, which tells the computer that we want to look at the alpha message; the other is *04 PSE*, which tells the computer to pause for a little less than a second.

Now it's time to start the program. Since we want the results displayed in hours, minutes and seconds, we'll need four spaces to the right of the decimal point. Press **shift FIX 4** and the display will show *05 FIX 4*. The distance we're concerned with is 100 miles, so enter the number **100** here,

HP-41 (long version)

```
01 LBL 'ST
02 'SPEED/TIME
03 AVIEW
04 PSE
05 FIX 4
06 'DISTANCE ?
07 PROMPT
08 'MILES/HOUR ?
09 PROMPT
10 /
11 HMS
12 STO 00
13 'TIME
14 AVIEW
15 PSE
16 'H.MMSS
17 ARCL 00
18 AVIEW
19 STOP
20 GTO 'ST
21 END
```

Fig. 7-4. The longer version of the Speed /Time /Distance program for the HP-41C.

the display showing *06 100*. The computer must now tell the operator it wants to know the speed to use in the calculation. Press **ALPHA M I L E S shift / H O U R ALPHA** and the display shows *07 MILES/HOUR*, telling us what the computer wants to know. Follow this by pressing **XEQ ALPHA P R O M P T ALPHA**, which puts *08 PROMPT* into the display and reminds the computer that to wait until new data has been entered. Press ÷ (the display shows *09 /*) to tell the computer to divide the previously entered figure of 100 by the data just entered.

At this point the computer is ready to give us the result, but that result would be in hours and decimals of hours, which we don't want. To tell the computer to convert the result to hours, minutes and seconds press **XEQ ALPHA H M S ALPHA.** That puts *10 HMS* into the display. Store that figure in register 00 by pressing **STO 00** (display: *11 STO 00*). To remind the operator what he is about to see, press **ALPHA T I M E ALPHA** followed by **ALPHA shift AVIEW ALPHA** and **XEQ ALPHA P S E ALPHA** which places the word *TIME* into display for a little less than a second. To display the result, press **ALPHA H shift • M M S S =**, which shows in the display as *15 'H.MMSS=*. Follow that by pressing **shift ARCL 00**, which recalls the result previously stored in register 00 and places it into the display along with the H.MMSS= identification. Pressing **shift AVIEW ALPHA R/S** places *18 STOP* into the display. We're ready to run the program; press **PRGM** to take the computer out of program mode.

There are two ways to get a program started. One is to press **XEQ ALPHA S T ALPHA.** The other is to assign that command to a key by pressing **shift ASN ALPHA S T ALPHA** and the key, for instance, the top right key marked **LN**. Press **USER**, and check that the word user appears at the left bottom edge of the display. Simply press **LN** and that one keystroke will always activate the program, assuming the computer is in user mode (Figs. 7-5 and 7-6).

The first thing that shows in the display when we activate the program is *SPEED/TIME*. That remains in display slightly short of a second and

HP-41 (short version)	
Press:	Display:
LN	DISTANCE ?
382	382
R/S	MILES/HOUR ?
65	65
R/S	5.5237

Fig. 7-5. The keystrokes and the displays when running the short version of the program.

HP-41 (long version)	
Press:	Display:
LN	SPEED/TIME
	DISTANCE ?
382	382
R/S	MILES/HOUR ?
65	65
R/S	TIME
	H.MMSS=5.5237

Fig. 7-6. The keystrokes and the displays when running the longer version of the program.

then changes to *MILES/HOUR*. That display remains until we enter the speed figure we're interested in. Enter **55** and the display shows *55*. Press **R/S** to start the program again. The display shows *TIME* and then changes to *H.MMSS=1.4905*, representing one hour, 49 minutes and five seconds.

To examine other speeds, press **LN** and the program starts over. There is another, simpler way to start the program over which does not require the computer in user mode. To tell the computer that if we press **R/S** after the result has been displayed, run the program to the end, then press **PRGM** to put the computer into program mode. It will undoubtedly present you with a display something like *00 REG 29*. Press **shift BST** and the display will change to *STOP*, the last program step you previously entered. Now press **shift GTO ALPHA S T ALPHA** and the display will show *19 GTO 'ST*. Press **PRGM** to take the computer out of program mode and you may (it is not mandatory) press **shift GTO • •**; the display will read *PACKING* because that command placed an *END* on the end of the program, and asked the computer to compress the program into the shortest possible space (to leave as much room as possible for additional programs). Packing a program means to compress it.

We now have a nice, simple program which will readily give you the time required to cover 100 miles at any speed.

If having the distance figure remain at 100 miles is not practical, simple change in the program will expand its capabilities.

First press **PRGM**. The display will read *GTO 'ST* or some such thing; press **SST** a few times and the display will go back to the beginning of the program (*01 LBL 'ST*) and show each program step in succession. When step 06 comes up the figure *100* will be in display. Press the ← key to delete that step, and the display will return to *05 FIX 4*. To have the computer prompt you for the distance, press **ALPHA D I S T A N C E space ? ALPHA XEQ ALPHA P R O M P T ALPHA**. Then press **SST** and *MILES/HOUR* will appear in the display. Take this opportunity to make an

improvement; since *MILES/HOUR* is not a statement but a question, it should have a question mark at the end. Press←, which removes the display, and then enter it again, this time ending with **space ?**. We're now ready to run the new and improved version of our program.

Press **PRGM** to take the computer out of program mode, then press **LN** (or **XEQ ALPHA S T ALPHA**). After first displaying *TIME/SPEED*, the display reads *DISTANCE?*. Enter the distance, say, **382** followed by **R/S**. The display will ask for *MILES/HOUR?*; enter whatever average speed you want to use, say **65**, followed again by **R/S**. The display reads *TIME* and then changes to *H.MMSS 5.5237*, meaning that at 65 mph it will take five hours, 52 minutes and 37 seconds to cover 382 miles.

Since there appears to be a huge discrepancy in the number of program steps used by the two computers, let me point out that the HP-41 can also be programmed to perform the complete program with only ten program steps. The result is the same, but the program is not as elegant. The short version, along with the long version, is shown in Figs. 7-3 and 7-4.

It never fails. Once you have successfully written a simple program, there always comes a time when you suddenly realize it might be nice to expand that program by adding other steps to solve additional problems.

Expanded Program

So far our program is capable of doing only one thing: given distance and speed information, it tells us the time it takes to cover that distance at that speed. Even in that simple state, it is not restricted to use with automobiles or airplanes or other conventional means of transportation. For instance, when it asks for distance information, enter 1,000,000,000 (without the commas, of course) for one billion miles. When it asks for miles per hour, enter 186,411.4833, the speed of light in miles per second. Multiply that figure by the number of seconds in an hour, or 3,600, and the result is the speed of light per hour, 660,281, 339.9. Now, if we ask the computer to use that figure in its computation, it tells us it takes light 1.3052, or one hour, 30 minutes and 52 seconds to travel one billion miles. Since radio waves travel at the speed of light, that is the time it takes for some radio messages to travel to and from space vehicles as Voyager II.

It's logical that a program designed to solve time/speed/distance problems would do so in all possible combinations. Thus, given distance and speed it will tell time, given distance and time it should compute the speed, and given speed and time it should compute the distance. It should also be designed to permit the user to simply ask for any of those three types of computations. To accomplish all this, here is what is involved:

TI-59. We'll assign the distance value to key A, the speed value to key B and the time value to key C. While there are a number of ways to set this up, the simplest and easiest to understand is to assign the result in terms of distance to key A′, the result in terms of speed to key B′ and the result in terms of time to key C′.

Store the variables as they are entered in registers 01, 02, and 03. Then, when the appropriate key is pressed, the computer will recall those variables, make the appropriate computations and display the result. Figure 7-7 shows the steps involved. A decimal-fix limitation was added to the distance and speed sections to avoid unnecessary fractions.

Figure 7-8 shows how simple it is to run the program and to get answers to three different types of problems.

HP-41C. This time I decided to take the longer of the two versions which we prepared because the alpha identifications make running the program a simple manner, needing no special routine. That is, I thought I'd be able to simply add to that program, but in the end it didn't work. Instead, I worked out an entirely new set of program steps. Figures 7-9 and 7-10 show the steps and what happens when the program is run.

Program Steps

TI-59

Cards:1
Passes:1
Partition: 479 59

Short Version:
1. Enter distance
2. Press **A**

TI59

000 Lbl	026 02	—	014 D.MS	040 Lbl	
001 A	027 =		015 STO	041 C'	
002 STO	028 R/S		016 03	042 Fix	
003 01	029 Lbl		017 R/S	043 04	
004 R/S	030 B'		018 Lbl	044 RCL	
005 Lbl	031 Fix		019 A'	045 01	
006 B	032 00		020 Fix	046 /	
007 STO	033 RCL		021 00	047 RCL	
008 02	034 01		022 RCL	048 02	
009 R/S	035 /		023 03	049 =	
010 Lbl	036 RCL		024 x	050 INV	
011 C	037 03		025 RCL	051 D.MS	
012 Fix	038 =			052 R/S	
013 04	039 R/S				

Fig. 7-7. The extended version of the Speed /Time /Distance program for the TI-59.

T159 (INPUT NUMBERS ARE ARBITRARY)

PRESS:	DISPLAY:	
100	100	DISTANCE
A	100	
78	78	SPEED
B	78	
2nd C'	1.2821	TIME
100	100	DISTANCE
A	100	
1.35	1.35	TIME
C	1.5833	TIME (DECIMALS)
and B'	63	SPEED
85	85	SPEED
B	85	
3.47	3.47	TIME
C	3.7833	TIME (DECIMALS)
2nd A'	322	DISTANCE

Variables can be entered in any order, it makes no difference.

Fig. 7-8. The keystrokes and displays when running the extended version of the program in the TI-59.

3. Enter speed
4. Press **B**

Display: Time in hours, minutes and seconds

Extended Version:

1. Enter distance
2. Press **A**
3. Enter speed
4. Press **B**
5. Press **2nd C'**

Display: Time (H.MMSS)

or

1. Enter distance
2. Press **A**
3. Enter time (H.MMSS)
4. Press **C**
5. Press **2nd B'**

Display: Speed

or

1. Enter speed
2. Press **B**
3. Enter time (H.MMSS)
4. Press **C**
5. Press **2nd A'**

Display: Distance

HP-41C

Cards: 1
Passes: 1 (short); 2 (long)
Size: 004

Short Version:

Display: *DISTANCE?*
1. Enter distance; **R/S**
Display: *MILES/HR?*
2. Enter speed; **R/S**
Display: Time in hours, minutes and seconds.

Extended Version:

1. Press **A** (not in alpha mode)
Display: *DISTANCE?*
2. Enter distance; **R/S**
Display: *TIME?*
3. Enter time (H.MMSS); **R/S**
Display: *MPH=* and speed
or
1. Press **B** (not in alpha mode)
Display: *MPH?*
2. Enter speed; **R/S**
Display: *TIME?*
3. Enter time (H.MMSS); **R/S**
Display: *MILES=* and distance
or
1. Press **C** (not in alpha mode)
Display: *MPH?*
2. Enter speed; **R/S**
Display: *DISTANCE?*
3. Enter distance; **R/S**
Display: *H.MMSS=* and time

HP-41

```
01 LBL A
02 MPH ?
03 PROMPT
04 STO 01
05 RTN
06 LBL B
07 DISTANCE ?
08 PROMPT
09 STO 02
10 RTN
11 LBL C
12 TIME ?
13 PROMPT
14 HR
15 STO 03
16 RTN
17 LBL AA
18 XEQ B
19 XEQ C
20 RCL 02
21 RCL 03
22 /
23 STO 00
24 FIX 0
25 MPH=
26 ARCL 00
27 AVIEW
28 STOP
29 LBL BB
30 XEQ A
31 XEQ C
32 RCL 01
33 RCL 03
34 x
35 STO 00
36 FIX 0
37 MILES=
38 ARCL 00
39 AVIEW
40 STOP
41 LBL CC
42 XEQ A
43 XEQ B
44 RCL 02
45 RCL 01
46 /
47 HMS
48 STO 00
49 FIX 4
50 H.MMSS=
51 ARCL 00
52 AVIEW
53 STOP
54 END

Key assignments:
(for clarity the letter
identifications of the
keys are used, though the
computer must NOT be in
alpha mode)
To find speed.....A
To find distance.....B
To find time.....C
    (The computer must
     be in USER mode.)
```

Fig. 7-9. The program steps for the extnded Speed/Time/Distance program for the HP-41C.

HP-41 (INPUT NUMBERS ARE ARBITRARY)

PRESS:	DISPLAY:	
ON	10,000.	LEFT OVER
USER	10,000. (USER)	
A	DISTANCE?	
150	150	MILES
R/S	TIME?	
1.05	1.05	HOURS, MINUTES
R/S	MPH=138	RESULT (SPEED)
B	MPH ?	
88	88	MILES PER HOUR
R/S	TIME ?	
1.47	1.47	HOURS, MINUTES
R/S	MILES=157	RESULT (DISTANCE)
C	MPH ?	
88	88	MILES PER HOUR
R/S	DISTANCE ?	
322	322	MILES
R/S	H.MMSS= 3.3933	RESULT (HOURS, MIN, SEC)

Programs can be called up in any order.

Fig. 7-10. The keystrokes and displays when running the extended version of the program in the HP-41C.

HOW MANY BRICKS TO BUILD A WALL?

This program, referred to in Chapter 4, determines how many bricks it will take to cover a certain area. Since especially here in New Mexico people like to build brick floors in which no mortar is used and the bricks are butted together directly, the program will provide answers with mortar or without. The program is based on the standard size of new bricks. If bricks of a non-standard size are to be used, or if you want answers for some other rectangular shape, you will have to multiply the length of the two sides of that non-standard brick or rectangle and enter the result in the appropriate place in the program. In the HP program the square-inch area to be reckoned with if mortar **is** used is 24.39 square inch, found in step 22 in the program. If mortar is **not** used, the area is 19.08 square inches, found in step 33 in the program. The way to easily replace these data is to press (not in the program mode) **GTO . .** followed by **022** or **033**. Then, when you press **PRGM**, step 22 (or 33) will appear in the display. Simply press – and the entry is deleted and the preceding step (21 or 32) is in display. Simply key in the new number and from then on, unless that number is changed again, the computer will use the new data in its computations (Figs. 7-11 and 7-12).

TI-59. It takes one side of a magnetic card to place the program into the computer. The display must respond with a steady *1*. We then key the wall data into the computer: **65, A, 8 R/S.** The next step is to enter the cost per brick, **0.14** and press **B.** The computer now has what it needs. We press **C** and the display shows *3070*, the number of bricks needed with mortar. Press R/S and the display changes to *429.82*.

Next, press **D** and the display shows *3925*, the number of bricks needed without mortar. Pressing **R/S** changes the display to *549.43*, the cost of that larger number of bricks.

HP-41C. After identifying the program with *BRICKWALL*, the display asks *WALL SIDE 1?* Enter **65** for a 65-foot long wall. The next

```
000  76 LBL 027  13  C  054  14  D
001  11  A  028  58 FIX 055  58 FIX
002  42 STO 029  00  00 056  00  00
003  00  00 030  43 RCL 057  43 RCL
004  91 R/S 031  00  00 058  00  00
005  42 STO 032  55  ÷  059  55  ÷
006  01  01 033  02  2  060  01  1
007  65  × 034  04  4  061  09  9
008  43 RCL 035  93  .  062  93  .
009  01  00 036  03  3  063  00  0
010  95  = 037  09  9  064  08  8
011  65  × 038  95  =  065  95  =
012  01  1 039  42 STO 066  42 STO
013  04  4 040  01  01 067  01  01
014  04  4 041  91 R/S 068  91 R/S
015  95  = 042  58 FIX 069  58 FIX
016  42 STO 043  02  02 070  02  02
017  00  00 044  43 RCL 071  65  ×
018  43 RCL 045  02  02 072  43 RCL
019  01  01 046  65  × 073  02  02
020  91 R/S 047  43 RCL 074  95  =
021  76 LBL 048  01  01 075  42 STO
022  12  B 049  95  =  076  03  03
023  42 STO 050  42 STO 077  91 R/S
024  02  02 051  03  03 078  00  0
025  91 R/S 052  91 R/S
026  76 LBL 053  76 LBL
```

Fig. 7-11. The Brickwall program, written for the TI-59.

```
   01♦LBL "BR"
02 "BRICKWALL"
   03 AVIEW
   04 PSE
   05 FIX 0
06 "WALL SIDE 1?"
   07 PROMPT
   08 STO 00
09 "WALL SIDE 2?"
   10 PROMPT
   11 STO 01
   12 RCL 00
   13 *
   14 144
   15 *
   16 STO 00
17 "MORTAR? 0=NO"
   18 PROMPT
   19 X=0?
   20 GTO 01
   21 RCL 00
   22 24.39
   23 /
   24 STO 01

   25♦LBL 03
26 "NO.BRICKS="
   27 ARCL 01
   28 AVIEW
   29 STOP
   30 GTO 02

   31♦LBL 01
   32 RCL 00
   33 19.00
   34 /
   35 STO 01
   36 GTO 03

   37♦LBL 02
38 "$ PER BRICK?"
   39 PROMPT
   40 STO 02
   41 RCL 01
   42 *
   43 STO 03
   44 FIX 2
   45 "COST=$"
   46 ARCL 03
   47 AVIEW
   48 STOP
   49 END
```

Fig. 7-12. Using this program, the HP can figure the number of bricks or other rectangular shapes needed to cover a given area.

question, *WALL SIDE 2?* is answered with the height of the wall, say **8** feet. The display now asks *MORTAR? 0=NO*. For this exercise, assume that mortar will be used. Simply press **R/S**. The display comes back immediately as *NO.BRICKS = 3,070.* **R/S** changes that to *$ PER BRICK?* Let's say 14 cents, and enter **0.14**. The display tells us *COST = $429.82* (plus labor).

Just by way of comparison, without the use of mortar it would have taken 3,925 bricks, raising the cost to $549.43.

Program Steps

TI-59

Cards:1
Passes: 1
Partition: 479 59
1. Key in length of wall
2. Press **A**
3. Key in height of wall
4. Press **R/S**
5. Enter cost per brick
6. Press **B**
7. Press **C**

Display: Total number of bricks, assuming mortar is used.

8. Press **R/S**
Display: Total cost

9. Press **D**
Display: Total number of bricks when no mortar is used

10. Press **R/S**
Display: Total cost

Cards: 1
Passes: 2
Size: 004
Display: *WALL SIDE 1?*
1. Enter length of wall; **R/S**
Display: *WALL SIDE 2?*
2. Enter height of wall; **R/S**
Display: *MORTAR? 0=NO*
3. If mortar is to be used, press **R/S**. If not, press **0** and **R/S**
Display: *NO. OF BRICKS=* and number
4. Press **R/S**
Display: *$ PER BRICK?*
5. Enter cost per brick; **R/S**
Display: *COST= $* and total cost

TEST

Here is a simple program that demonstrates the basic differences between the ways the two computers function. The purpose of the program is to perform a series of calculations with any arbitrary number. You will discover that regardless of what the number is, the result of the calculation is always the same. The formula is:

number × 2 + 8 − 2 = / 2 − number = 3

Number is the variable which must be put into the computer; 3 is the result no matter the input number.

TI-59. You start by entering a label and follow it by determining a memory position for the variable number. Then punch in the calculation in the same order shown above and store the result in another memory position. In this case the program would work just as well without that final **STO 01.**

The keystrokes and the effects produced by them are:

Action	Display	Remarks
Turn TI-59 on	0	Places computer into the **LRN**
Press **LRN**	000 00	(learn) mode, which is the mode in which it remembers each keystroke and records a program.
Press **2nd Lbl**	001 00	Prepares the computer to accept a label.
Press **A**	002 00	Label A has been entered.
Press **STO**	003 00	Prepares the computer to accept information as to where to store input data.
Press **0** and **0**	004 00	Tells the computer to store input data in memory register 00. From now on, when a number is keyed in and **A** is pressed after that, the number will be placed into register 00

Action	Display	Remarks
		(always replacing anything that might previously have been stored in R_{00}).
Press × (times)	005 00	Prepares the computer to multiply the stored value by the next number.
Press **2**	006 00	The input value will be multiplied by 2.
Press =	007 00	Produces the result.
Press +	008 00	Prepares the computer to add the next number to the result.
Press **8**	009 00	Adds 8 to the above.
Press =	010 00	Produces the sum.
Press –	011 00	Prepares the computer to deduct the next number from the previous sum.
Press **2**	012 00	Deducts 2.
Press =	013 00	Computes the result.
Press ÷	014 00	Prepares to divide previous result by the next number.
Press **2**	015 00	Divides by 2.
Press =	016 00	Computes the result of the division.
Press –	017 00	Prepares to deduct the next value.
Press **RCL**	018 00	Since the next value is the one placed into memory (R_{00}) we now ask the computer to recall it.
Press **0** and **0**	019 00	Tells the computer that the desired value is stored in R_{00}.
Press =	020 00	Computes the result.
Press **STO**	021 00	Prepares to store the result in a memory position (this is optional).
Press **0** and **1**	022 00	Tells the computer to store the result in R_{01}.
Press **R/S**	023 00	R/S stands for Run/Stop and tells the computer in this instance that the program is finished and the final result should be displayed in the display window.
Press **LRN**	0	This takes the computer out of the **LRN** (learn) mode and the program can now be run.

Assuming the first number we want to use is, say, 27.85, the keystrokes are these:

Press **2**	2	
Press **7**	27	
Press •	27.	
Press **8**	27.8	
Press **5**	27.85	
Press **A**	C then 3	The flickering C indicates that the computer is computing. The stable 3 is the result of the computation.

We're now ready to repeat the process with different numbers input as often as we want.

HP-41C. Since the HP-41C displays letters as well as numbers and operates with reverse Polish notation (RPN), the order in which steps are entered is different. Start by giving the program a name (TEST). Then enter a display to tell the operator what the computer wants to know next. Follow this by entering the actual steps of the calculation and finish by having the computer announce that the answer to the problem is always the same. Follow with a command to go back to the beginning, to permit us to run the program again with a different input.

Keystroke	Display	Remarks
ON	*33.8*	This display could be anything. It is simply whatever was in the display when the computer was last turned off, because the HP-41C has a non-volatile memory.
PRGM	*00 REG 46*	Puts the computer into the program mode and displays the number of registers which are available for data storage.
shift (the yellow key) and **LBL**	*01 LBL __*	Indicates that this is the first step and that the computer wants to know the name of the program. The two spaces after *LBL* prompt the operator for two digits.
ALPHA	*01 LBL _*	Since we want to call the program Test, change the computer to the alpha mode. You'll notice that in this mode there is only one dash after *LBL*, meaning that only one letter is required.

Keystroke	Display	Remarks
T E S T	*01 LBL TEST* _	By pressing the keys marked in blue with the letters, we have placed the program name into the display. The dash prompt at the end of the line means that unless we tell the computer this is the end of that word, the next keystroke will be displayed as a continuation of the word test.
ALPHA	*01 LBL TEST*	By taking the computer momentarily out of the alpha mode we announce that this is the end of this line and the dash prompt disappears from the display.
ALPHA	*01 LBL TEST*	Since we want to see the name of the program when the computer is turned on, we now go back into alpha mode.
T E S T	*02 TEST* _	This time the name of the program appears without the *LBL*.
shift VIEW	*03 AVIEW*	By executing these keystrokes, the computer is told that we want to see alpha characters.
ALPHA	*03 AVIEW*	Takes the computer out of alpha mode.
XEQ	*04 XEQ* _ _	Prepares the computer to execute a command.
ALPHA	*04 XEQ* _	The command we want to give is in the form of letters, so we return the computer to the alpha mode.
P S E	*04 XEQ PSE* _	The command is *PSE*, meaning pause. This is a program step permanently stored in the HP-41C, and it tells the computer to display the message for a period of slightly less than a second.
ALPHA	*04 PSE*	Places the *PSE* command into step 4.
ALPHA	*04 PSE*	Returns the computer to the alpha mode.
A N Y space N U M B E R	**ANY NUMBER** _	This tells the operator what to

Keystroke	Display	Remarks
		do next (not important with a simple program like this, but it can prove to be very helpful in complicated programs). It looks as if we have skipped step 5. We haven't. **ANY NUMBER** was step 5; because of the number of letters there was no room to display the program step number.
XEQ	*06XEQ __*	
ALPHA	*06 XEQ –*	
P R O M P T	*06 XEQ PROMPT*	
ALPHA	*06 PROMPT*	This series of keystrokes tells the computer some input must be made before continuing with the program.
STO 0 0	*07 STO 00*	Tells the computer to store the input data in memory register number 00 (R_{00}).
2	*08 2 –*	
×	*09 x*	At this point the computer will multiply the data in R_{00} by 2. (Remember, with reverse Polish notation there are no equal signs or parentheses.)
8	*10 8 __*	The next number the computer will use in some way is 8.
+	*11 +*	Tells the computer what to do with 8.
2	*12 2 __*	Next number to use.
–	*13 –*	Deduct 2.
2	*14 2*	
÷	*15 /*	Divide by 2.
STO 0 1	*16 STO 01*	Stores the intermediate result in R_{01}.
RCL 0 0	*17 RCL 00*	Recalls the data stored in R_{00}.
–	*18 –*	Deducts the data from R_{00} from the amount stored in R_{01}.
STO 0 1	*19 STO 01*	Replaces the previously stored data with the new result in R_{01}.
shift FIX 0	*20 FIX 0*	Tells the computer not to display any decimals.
ALPHA	*20 FIX 0*	

Keystroke	Display	Remarks
A L W A Y S =	*21 ALWAYS=*	
shift RCL 01	*22 ARCL 01*	Recalls data stored in R_{01} to the alpha register to be displayed, along with the word ALWAYS.
shift VIEW	*23 AVIEW*	
R/S	*24 STOP*	
ALPHA	*24 STOP*	
shift GTO	*25 GTO __*	These keystrokes tell the computer to go to the line in the program which reads test. Takes the computer out of program mode. We're now ready to run the program again.
ALPHA	*25 GTO _*	
T E S T	*25 GTO TEST*	
PRGM	*0.0000*	

There are two ways to get the program started. You can enter this series of keystrokes each time: **XEQ ALPHA T E S T ALPHA**, obviously a bit cumbersome, or, you may assign the label test to a single key:

shift ASN	*ASN*	ASN stands for assign.
ALPHA	*ASN _*	
T E S T	*ASN TEST _*	
ALPHA	*ASN TEST _*	
LN	**15** then changes to *0.0000*	
USER		*0.0000* with a tiny user in the left bottom corner of the display. In this mode the computer will execute the special assignments which the user has given to certain keys.
LN	*TEST*	This display changes to prompt the operator to enter any number.
	ANY NUMBER	
2 7 . 8 5	*27.85 _*	
R/S	*ALWAYS= 3.*	**R/S** tells the computer to compute and then to display the result together with the string of letters previously entered. We're ready to start over again.
R/S	*TEST, ANY NUMBER*	

While the optional printers for the two computers are by no means necessary, they do permit you to study the programs entered to detect mistakes and examine the steps the computer takes to arrive at the final

```
                                    014  02  2
000  76 LBL   007  85  +           015  95  =
001  11  A    008  08  8           016  75  -
002  42 STO   009  95  =           017  43 RCL
003  00  00   010  75  -           018  00  00
004  65  ×    011  02  2           019  95  =
005  02  2    012  95  =           020  42 STO
006  95  =    013  55  ÷           021  01  01
                                    022  91 R/S
```

Fig. 7-13. Printout of the Test program for the TI-59.

```
01♦LBL "TEST"      10 8          19 STO 01
  02 "TEST"        11 +          20 FIX 0
  03 AVIEW         12 2          21 "ALWAYS="
  04 PSE           13 -          22 ARCL 01
05 "ANY NUMBER"    14 2          23 AVIEW
  06 PROMPT        15 /          24 STOP
  07 STO 00        16 STO 01  25 GTO "TEST"
  08 2             17 RCL 00     26 .END.
  09 *             18 -
```

Fig. 7-14. Printout of the Test program for the HP-41C.

result. Figures 7-13, 7-14, 7-15, 7-16, 7-17, and 7-18 show the two programs and a complete trace of the running of the programs.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 002
1. Enter any number	1. Enter any number
2. Press **A**	2. Press **R/S**
Display: Result	Display: Result.

```
27.85  STO      63.7   -
         0        2.   =        27.85
27.85            61.7           27.85   =
27.85   ×        61.7   ÷          3.
   2.   =          2.   =          3.   STO
 55.7               30.85                  1
 55.7   +          30.85   -          3.
   8.   =          30.85   RCL              R/S
 63.7 _                      0
```

Fig. 7-15. This printout traces the operation of the TI-59 as it runs the Test program.

```
XEQ "TEST"

01♦LBL "TEST"
"TEST"
AVIEW
TEST
PSE
"ANY NUMBER"
PROMPT
ANY NUMBER
27.85    RUN
STO 00
2
*
56.      ***

8
+
64.      ***
2
-
62.      ***
2
/
31.      ***
STO 01
RCL 00
28.      ***
-
3.       ***
STO 01

FIX 0
"ALWAYS="
ARCL 01
AVIEW
ALWAYS=3.
STOP
RUN
GTO "TEST"

01♦LBL "TEST"
"TEST"
AVIEW
TEST
PSE
"ANY NUMBER"
PROMPT
ANY NUMBER
```

Fig. 7-16. This printout traces the operation of the HP-41C as it runs the Test program.

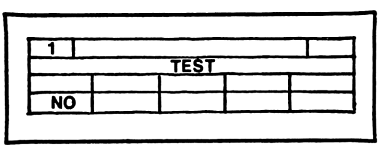


Fig. 7-17. A sample magnetic card for the TI-59.

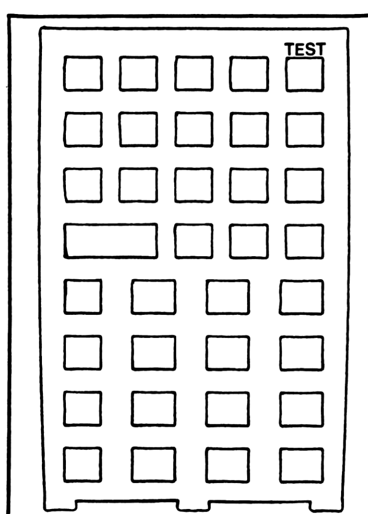


Fig. 7-18. A sample keyboard overlay for the HP-41C, to remind the operator what to do to run the program.

Chapter 8
Programs Using Arithmetic

This chapter contains programs that use arithmetic. Study them carefully, and you will have no trouble incorporating the basic methods into any program.

RECIPROCALS

This simple program tells us the reciprocals of degrees. That sounds like a simple matter, but it is not since it uses fractions or degrees, minutes and seconds. In terms of degrees and decimal fractions of degrees, determining the reciprocal of, say 193.84 (13.84) or 67.33 (247.33) is not awfully difficult, though, as any experienced pilot will tell you, in moments of stress it can turn into something much more complicated than it actually is.

Dealing with degrees, minutes, and seconds is pretty much the same, except that the way of presenting the input and the result is different. The program allows you to use either type of input and display.

The program for the HP is longer because it uses a different method to arrive at the result, and there are those little embellishments available because of the alphanumerics in the computer.

With either computer, to run the program simply key in the initial degree data and the display will immediately display the result in the same format in which it was put in, be it decimals or minutes and seconds. This is accomplished by the flags you'll see in the printouts of the program (Figs. 8-1 and 8-2).

Program Steps

TI-59

Cards: 1
Passes: 1

Partition: 479 59
1. Enter data in degrees,

minutes and seconds
2. Press **A**
Display: The reciprocal to the above in the same format

HP-41C

Cards: 1
Passes: 2
Size: 003
Display: *DEGREE?*
1. Enter data in degrees, minutes and seconds
2. Press **R/S**
Display: *DEGR.*= and the reciprocal in the same format

000	25	CLR	019	77	GE	038	00	00
001	47	CMS	020	12	B	039	88	DMS
002	76	LBL	021	91	R/S	040	42	STO
003	11	A	022	76	LBL	041	00	00
004	22	INV	023	12	B	042	86	STF
005	86	STF	024	85	+	043	00	00
006	00	00	025	03	3	044	61	GTO
007	42	STO	026	06	6	045	35	1/X
008	00	00	027	00	0	046	76	LBL
009	76	LBL	028	95	=	047	34	FX
010	35	1/X	029	42	STO	048	58	FIX
011	58	FIX	030	01	01	049	04	04
012	02	02	031	87	IFF	050	43	RCL
013	75	-	032	00	00	051	01	01
014	01	1	033	34	FX	052	22	INV
015	08	8	034	91	R/S	053	88	DMS
016	00	0	035	76	LBL	054	42	STO
017	95	=	036	16	A'	055	01	01
018	22	INV	037	42	STO	056	91	R/S

Fig. 8-1. The TI-59 version of Reciprocals.

USING PARENTHESES

Many mathematical formulas which you might want to use contain one or several pairs of parentheses. It is important to realize what has to be done to produce the correct results.

The TI is designed to deal with parentheses and include them as steps in a program. This is not so with the HP.

In this program I have taken an equation which involves one pair of parentheses and written it into a program for the two computers. The basic equation is this:

$$2(N+1)^2=$$

N stands for any number you might want to use.

```
 01↑LBL "RECIP"      22↑LBL 04          43 "D.MMSS="
02↑LBL "PPG 007"     23 180              44 ARCL 00
 03 "RECIPROCAL"     24 -                45 AVIEW
    04 AVIEW         25 STO 00           46 STOP
    05 PSE           26 ENTER↑           47 GTO 02
                     27 0
    06↑LBL 01        28 X>Y?             48↑LBL A
    07 CF 00         29 GTO A            49 RCL 00
    08 0             30↑LBL B            50 360
  09 "DEGREE ?"      31 FIX 2            51 +
    10 PROMPT        32 FS? 00           52 STO 00
    11 STO 01        33 GTO 05           53 GTO B
    12 X=0?          34 "DEGR="
    13 GTO 03        35 ARCL 00          54↑LBL 02
    14 GTO 04        36 AVIEW       55 "AGAIN? 0=YES"
                     37 STOP             56 PROMPT
    15↑LBL 03                            57 X=0?
    16 SF 00         38↑LBL 05           58 GTO 01
  17 "D.MMSS ?"      39 FIX 4            59 "END"
    18 PROMPT        40 RCL 00           60 AVIEW
    19 STO 01        41 HMS              61 STOP
    20 HR            42 STO 00           62 .END.
    21 STO 01
```

Fig. 8-2. The HP-41C version of Reciprocals.

Say that N equals 5; the equation adds 1 to 5, making the sum inside the parentheses 6. The 2 superscript at the end squares that amount: $6 \times 6=36$. The 2 preceding the parentheses multiples that figure by itself: $36 \times 2 = 72$. Thus, the answer to the equation is 72.

```
000   76 LBL  007   02   2   014   54   )
001   11  A   008   65   ×   015   33  X²
002   42 STO  009   53   (   016   95   =
003   00  00  010   43 RCL   017   42 STO
004   91 R/S  011   00  00   018   01  01
005   76 LBL  012   85   +   019   91 R/S
006   12  B   013   01   1   020   00   0
```

Fig. 8-3. The TI-59 can handle a mathematical equation which includes parentheses.

When we use the HP, a complete understanding of each section of an equation is of vital importance. The HP has no parentheses on its keyboard (the < and > in the printout take the place of parentheses, though they are actually the symbols which stand for greater than and smaller than). As a result, we often have to take a mathematical formula and turn it inside out to adopt it to the HP's reverse Polish notation system (Figures 8-3 and 8-4).

The following are the steps in each program, with an explanation of what they do.

TI-59

000 LBL
001 A Prepares the computer to accept data for storing.
002 STO
003 00 Store in R_{00}.
004 R/S Stop.
005 LBL
006 B This key will start the actual program.
007 2 The first digit in the equation.
008 x Commands to multiply.
009 (
010 RCL
011 00 Recalls the number stored in R_{00}.
012 +
013 1
014)
015 x^2 Squares sum inside ( )
016 = Performs the multiplication with 2 in step 007.
017 STO
018 01 Stores result in R_{01}.
019 R/S Stop. Hold the result in display.

```
01*LBL "PPG 042"      09 "ENTER NO."        17 STO 00
02 "PARENTHESES"         10 PROMPT          18 FIX 2
     03 AVIEW            11 STO 00          19 "EQUALS:"
     04 PSE              12 1               20 ARCL 00
     05*LBL 00           13 +               21 AVIEW
   06 "2(N+1)↑2"         14 X↑2             22 STOP
     07 AVIEW            15 2               23 GTO 00
     08 PSE              16 *               24 END
```

Fig. 8-4. The HP-41C, using reverse Polish notation, employs no parentheses (or = signs) in its calculations.

HP-41C

01 LBL PPG 042 My own file number for the program.
02 PARENTHESES The name of the program.
03 AVIEW Place into display.
04 PSE Hold in display.
05 LBL 00 Extra label to permit repeated use of the program without the need for renewed initialization.
06 2(N+1)↑2 Puts the equation into display.
07 AVIEW Places the equation into display.
08 PSE Hold in display.
09 ENTER NO. Request to enter number to be used in the equation.
10 PROMPT Holds the command until number has been entered.
11 STO 00 Stores number in R_{00}.
12 1 The 1 inside the parentheses.
13 + Adds the number stored in R_{00} to the 1 in the previous step.
14 X↑2 Squares the sum of the number in R_{00} and 1.
15 2 The 2 at the head of the equation.
16 × Multiply the preceding 2 by the squared contents of the parentheses.
17 STO 00 Put the new result into R_{00}.
18 FIX 2 Limits displayed decimals to two.
Steps 19 through 23 are used to display the result and step 23 (GTO 00) tells the computer to go back to LBL 00 (step 05).

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 001
1. Enter number	Display: *ENTER NO.*
2. Press **A**	1. Enter number; **R/S**
3. Press **B**	Display: *EQUALS:* and result
Display: Result	

MEASUREMENTS IN TWO DIMENSIONS

A good example of using one program to solve many related problems is this one, which measures a total area when some of the linear measurements are known (Fig. 8-5). Much of this kind of arithmetic can be done just as well in one's head. Few of us would have difficulty figuring the square-foot area of a room 10 by 15 feet (150 square feet), but, if that room should happen to be 10 feet, three and a quarter inches by 15 feet, 7 and 5/16 inches, we might wish there was a simple way to arrive at an exact answer. This program will do that.

The inches and their fractions must first be converted to decimals of feet. This could be incorporated into the program, but it would make the program unnecessarily long and cumbersome. Those conversions are eas-

ily made by using the calculator mode of the computers. The two sides of the room are 10.27 by 15.61 and the answer for the square footage of the area is 160.31. Since we already know that .31 equals 5/16, we can translate that into 160 feet and 5/16 inches square.

If you know the area and the length of one side of the room. This program will tell you the length of the other side.

TI-59. The section in the TI program, (Fig. 8-6) determining the area using the lengths of two sides is assigned to key **A**. The reverse portion of that program, which determines the length of one side given the length of the other side and the area, is assigned to key **A′** (steps 002 through 013 and 014 through 029 respectively). Both versions of the calculation involving triangles are assigned to keys **B** and **B′** in the same manner (steps 030 through 042 and 043 through 061). The calculation which defines the area of a circle based on the radius is assigned to key **C** (steps 062 through 071) and that defining the radius of a circle based on the area is assigned to key **D**

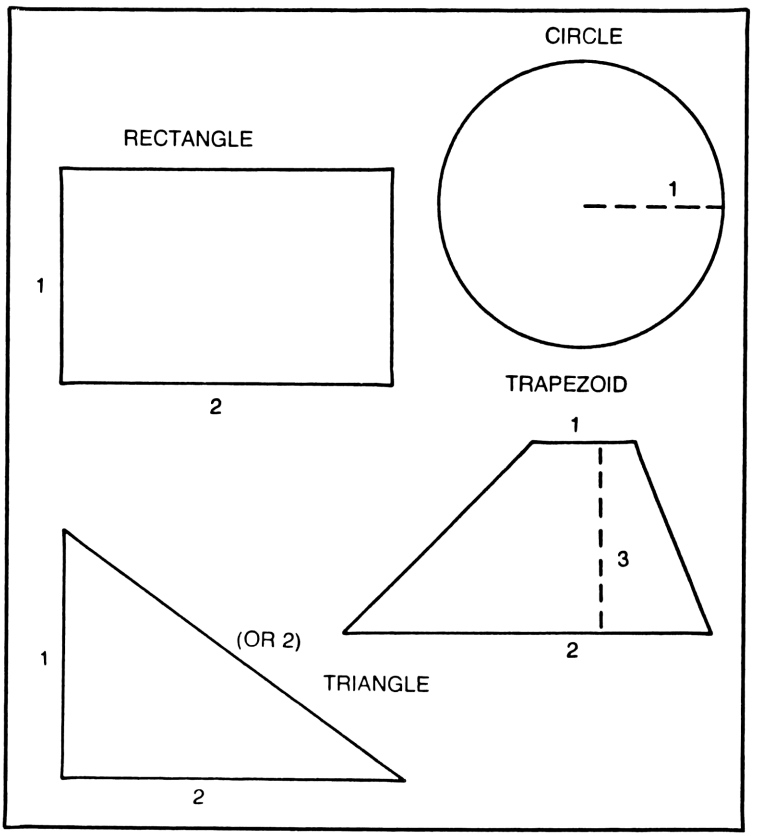


Fig. 8-5. The four shapes used in the Measurements in Two Dimensions program.

```
000  25 CLR  038  95  =   076  02  02
001  47 CMS  039  55  ÷   077  42 STO
002  76 LBL  040  02  2   078  00  00
003  11  A   041  95  =   079  55  ÷
004  58 FIX  042  91 R/S  080  89  π
005  02  02  043  76 LBL  081  95  =
006  42 STO  044  17 B'   082  34 ΓX
007  00  00  045  42 STO  083  95  =
008  65  ×   046  00  00  084  91 R/S
009  91 R/S  047  91 R/S  085  76 LBL
010  95  =   048  42 STO  086  15  E
011  42 STO  049  02  02  087  42 STO
012  00  00  050  43 RCL  088  00  00
013  91 R/S  051  00  00  089  91 R/S
014  76 LBL  052  65  ×   090  42 STO
015  16 A'   053  02  2   091  01  01
016  42 STO  054  95  =   092  91 R/S
017  00  00  055  42 STO  093  42 STO
018  91 R/S  056  01  01  094  02  02
019  42 STO  057  55  ÷   095  43 RCL
020  01  01  058  43 RCL  096  00  00
021  43 RCL  059  02  02  097  85  +
022  00  00  060  95  =   098  43 RCL
023  55  ÷   061  91 R/S  099  01  01
024  43 RCL  062  76 LBL  100  95  =
025  01  01  063  13  C   101  55  ÷
026  95  =   064  58 FIX  102  02  2
027  42 STO  065  02  02  103  95  =
028  00  00  066  42 STO  104  65  ×
029  91 R/S  067  00  00  105  43 RCL
030  76 LBL  068  33 X²   106  02  02
031  12  B   069  65  ×   107  95  =
032  58 FIX  070  89  π   108  42 STO
033  02  02  071  95  =   109  00  00
034  42 STO  072  91 R/S  110  91 R/S
035  00  00  073  76 LBL  111  00  0
036  65  ×   074  14  D
037  91 R/S  075  58 FIX
```

Fig. 8-6. Measurements in Two Dimensions for the TI-59.

(steps 073 through 084). The calculation that determines the area of a trapezoid is assigned to key **E** (steps 076 through 110).

When running the program in the HP-41C, press **R/S**, and the 160.31 measure will be accepted by the computer as the area measure. By answering the request *SIDE 1?* with a measure different than the one used before, say, **19** feet, pressing **R/S** again will give us the measurement *SIDE 2 = 8.44* feet, making it a rather long and narrow room.

HP-41. For the HP the different sections of the program defining the areas for rectangles, triangles, circles, and trapezoids are given separate labels and assigned to user-defined keys (Figs. 8-7, 8-8, and 8-9). Which of the keys are used to call up which section of the program is entirely up to the person using the program. The decision will often depend on what other programs, using user-defined keys, are also stored in the computer, since no such key can be used to perform a function in more than one program.

There is one peculiarity in the program for the HP which might require an explanation. The section labeled **A** (steps 02 through 06) is used by each of the four individual program segments. Each contains the command **XEQ A** (steps 20, 45, 56 and 86). It contains the sequence of commands necessary to place the results of the calculations into display. Since that involves a total of four separate steps (03 through 06), it seemed silly to repeat it at the appropriate place in each of the four programs. By simply telling the computer to **XEQ A** and then to **RTN** (return), a dozen program steps were saved, leaving more room for other programs in the computer.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

For Rectangle:
1. Enter length, side 1
2. Press **A**
3. Enter length, side 2
4. Press **R/S**

Display: Area measure for reverse:
1. Enter area measure
2. Press **2nd A′**
3. Enter length, one side
4. Press **R/S**

Display: Length, other side

For Triangle:
Same sequence, but use keys **B** and **B′**

For Circle:
1. Enter radius
2. Press **C**

Display: Area measure
or:
1. Enter area measure
2. Press **D**

Display: Radius

For Trapezoid:
1. Enter length, base 1
2. Press **E**
3. Enter length, base 2
4. Press **R/S**
5. Enter height
6. Press **R/S**

Display: Area measure

```
01↑LBL "AREA"
  02↑LBL A
  03 "AREA= "
  04 ARCL 00
  05 AVIEW
  06 STOP
  07 RTN

  08↑LBL "A3"
09 "RECTANGLE"
  10 AVIEW
  11 PSE
  12 FIX 2
13 "SIDE 1 :"
  14 PROMPT
  15 STO 00
16 "SIDE 2 :"
  17 PROMPT
  18 *
  19 STO 00
  20 XEQ A
  21 RCL 00
22 "SIDE 1 ?"
  23 PROMPT
  24 STO 01
  25 /
  26 STO 00
27 "SIDE 2="
  28 ARCL 00
  29 AVIEW
  30 STOP

  31↑LBL "A2"
32 "TRIANGLE"
  33 AVIEW
  34 PSE
  35 FIX 2
36 "SIDE 1 :"
  37 PROMPT
  38 STO 00
39 "SIDE 2 :"
  40 PROMPT
  41 *
  42 2
  43 /
  44 STO 00
  45 XEQ A

  46↑LBL "A1"
  47 "CIRCLE"
  48 AVIEW
  49 PSE
50 "RADIUS ?"
  51 PROMPT
  52 X↑2
  53 PI
  54 *
  55 STO 00
  56 XEQ A
  57 PI
  58 /
  59 SQRT
  60 STO 00

  61 "RADIUS="
  62 ARCL 00
  63 AVIEW
  64 STOP
  65↑LBL "A4"
66 "TRAPEZOID"
  67 AVIEW
  68 PSE
69 "BASE 1 ?"
  70 PROMPT
  71 STO 00
72 "BASE 2 ?"
  73 PROMPT
  74 STO 01
75 "HEIGHT ?"
  76 PROMPT
  77 STO 02
  78 RCL 00
  79 RCL 01
  80 +
  81 2
  82 /
  83 RCL 02
  84 *
  85 STO 00
  86 XEQ A
  87 "END"
  88 AVIEW
  89 STOP
  90 .END.
```

Fig. 8-7. Measurements in Two Dimensions for the HP-41C.

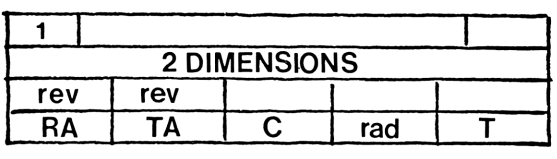


Fig. 8-8. The program card for the TI-59 with suggested legends.

Cards: 2
Passes: 3
Size: 003

For Rectangle:

1. Press **N** (row 4, key 1)
Display: *RECTANGLE*, then *SIDE 1?*
2. Enter length, side 1; **R/S**
Display: *SIDE 2?*
3. Enter length, side 2; **R/S**
Display: *AREA*= and measure
4. If desired, enter new area measure; **R/S**
Display: *SIDE ?*
5. Enter length of one side; **R/S**
Display: *SIDE 2*= length of other side.

For Triangle:

1. Press **shift 0** (row 4, key 2)
Display: *SIDE 1?*
2. Enter length, side 1; **R/S**
Display: *SIDE 2?*
3. Enter length, side 2; **R/S**
Display: *AREA*= and measure

For Circle:

1. Press **shift N** (row 4, key 1)
Display: *CIRCLE*, then *RADIUS?*
2. Enter radius; **R/S**
Display: *AREA*= and measure

For Trapezoid:

1. Press **shift P** (row 4, key 3)
Display: *TRAPEZOID*, then *BASE 1?*
2. Enter length, base 1; **R/S**
Display: *BASE 2?*
3. Enter length, base 2; **R/S**
Display: *HEIGHT?*
4. Enter height; **R/S**
Display: *AREA*= and measure

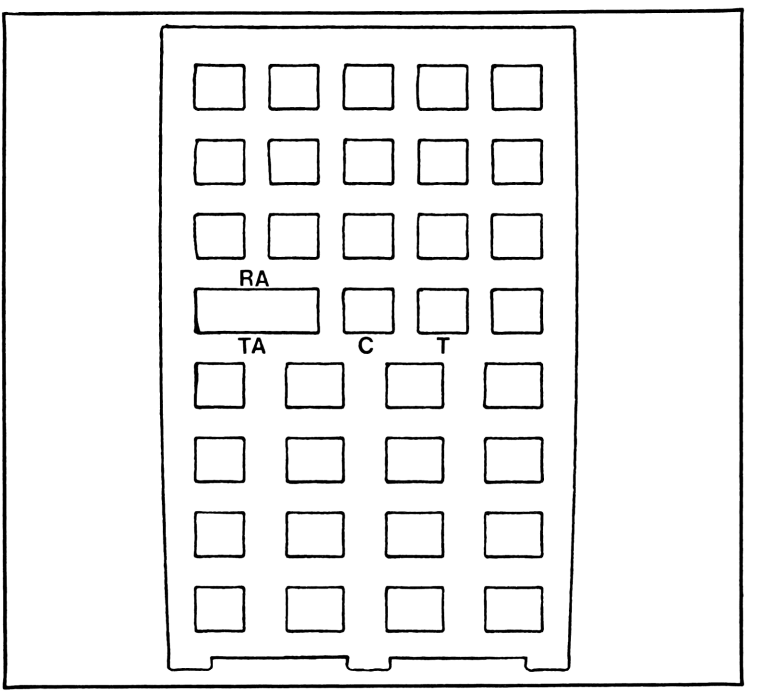


Fig. 8-9. The keyboard overlay for the HP-41C.

MEASUREMENTS IN THREE DIMENSIONS

This program requires repartitioning the TI-59, because the available 480 program steps won't do the trick. It is, in fact, a group of programs collected under one general heading. The purpose is to arrive at a series of measurements for a number of differently shaped three-dimensional objects. There is a rectangular box which could be a cube if the measurements for each side and the height are identical. There is a hexagon, a six-sided box; a cylinder; a sphere; a pyramid which can be four, six or eight sided; and a circular cone. In each case we determine the total surface area, the volume, and the surface area of the walls, excluding the top and bottom of all objects other than the sphere which, of course, has neither top nor bottom.

With both computers, any one of the program sections can be called up going through the entire program to arrive at, say, the circular cone which happens to be the last object on the list. With the TI, the data for the rectangular box is associated with key **A**, the data for the hexagon with key **B**, the cylinder with **C**, the sphere with **D**, the pyramid with **E** and the cone with **E'**. In the HP, the program can be called up and run from its beginning by pressing **shift 3.** Beyond that, the five keys in the top row and the first key in the second row control the programs for the different objects. For simplicity's sake I will use the letter designations even though the computer must **not** be in alpha mode (but it **must** be in the user mode) when the keys are used to call up their programs: **shift A**, the rectangular box; **shift B**, the hexagon; **shift C**, the cylinder; **shift D**, the sphere; **shift E**, the pyramid and **shift F**, the circular cone (see Figs. 8-10 and 8-11).

In each case, information about the measurements of the objects must be keyed into the computers for the programs to work. The HP tells us via its alphanumeric display what data are required in which order. Not so with the TI; anyone using this program with the TI-59 should make up a simple card which lists what information is called for in what order. Here is the information which can be copied for this purpose:

Rectangular Box:
Length, press **A**
Width, press **R/S**
Height, press **R/S**
for volume, press **R/S**
for surface area, press **R/S**
for walls only area, press **R/S.**

Hexagon:
Length, one side, press **B.**
Height press **R/S**
for volume, press **R/S**
for surface area, press **R/S**
for walls area only, press **R/S**

Cylinder:
Radius, press **C**

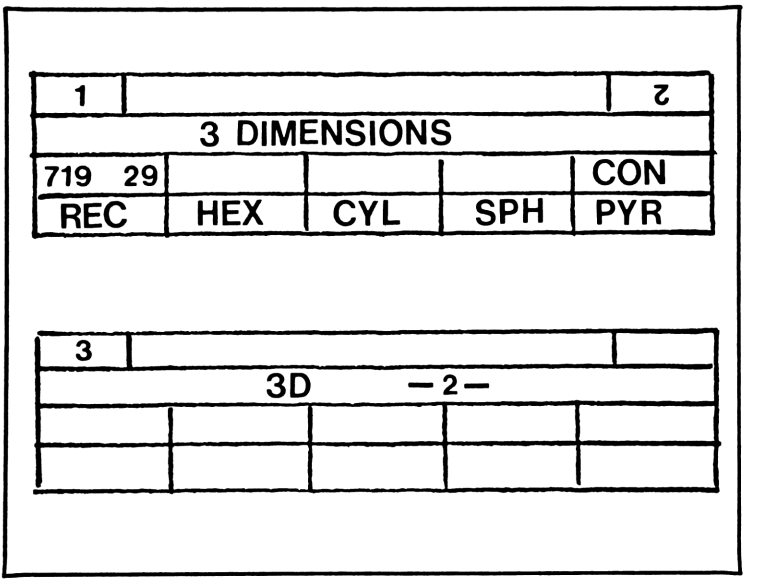


Fig. 8-10. Suggested legends for the magnetic cards on which Measurements in Three Dimensions is stored for the TI-59.

Height, Press **R/S**
for volume, press **R/S**
for surface area, press **R/S**
for walls only area, press **R/S**

Sphere:

Radius, press **D**
for volume, press **R/S**
for surface area, press **R/S**

Pyramid:

Length, one base side, press **E**
Number of sides, press **R/S**
Height, press **R/S**
for volume, press **R/S**
for surface area press **R/S**
for walls only area, press **R/S**

Circular Cone:

Base radius, press **2nd E'**
Height, press **R/S**
for volume, press **R/S**
for surface area, press **R/S**
for walls only area, press **R/S**

TI-59. In the program for the TI-59, the pyramid section is associated with key **E** (Fig. 8-13). It starts at step 251 and ends at step 459. Here the

section labeled **sin** (steps 261 through 283) is used to direct the computer to the next appropriate section, depending on the number of sides of the pyramid. If there are four sides, it is to go to **LBL INV**; if there are six sides it is to go to **LBL LNX**, and if there are eight sides it is to go to **LBL 1/X.**

Looking at those sections, we find that each starts off with **IFF 00** immediately after the **LBL** identification (steps 286, 287; 303, 302; and 322 and 323). **IFF 00** means that the programs ask the computer whether flag 0 has been set. If so, the computer will take the next step. If not, it will skip that step and go to the one after. Since flag 0 has not been set, the computer continues with the remaining steps in those sections until it is told to go to the next section, which is either **LBL (** or **LBL Y ˣ**. (Remember that most every key on the TI keyboard can be used as a user-defined label.)

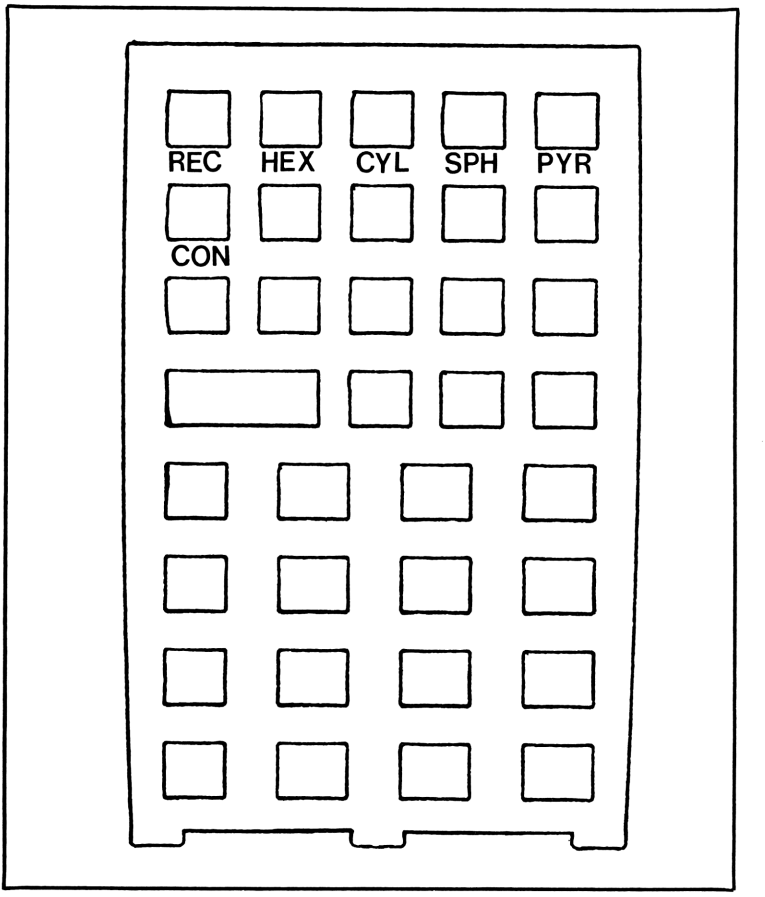


Fig. 8-11. Suggested legends for the keyboard overlay for Measurements in Three Dimensions for the HP-41C.

000	47	CMS	042	42	STO	084	91	R/S
001	25	CLR	043	05	05	085	42	STO
002	58	FIX	044	43	RCL	086	01	01
003	00	00	045	01	01	087	43	RCL
004	76	LBL	046	65	×	088	00	00
005	11	A	047	02	2	089	33	X^2
006	42	STO	048	65	×	090	55	÷
007	00	00	049	43	RCL	091	02	2
008	91	R/S	050	02	02	092	95	=
009	42	STO	051	95	=	093	65	×
010	01	01	052	85	+	094	06	6
011	91	R/S	053	43	RCL	095	95	=
012	42	STO	054	04	04	096	42	STO
013	02	02	055	85	+	097	02	02
014	65	×	056	43	RCL	098	91	R/S
015	43	RCL	057	05	05	099	43	RCL
016	01	01	058	95	=	100	01	01
017	65	×	059	42	STO	101	65	×
018	43	RCL	060	06	06	102	43	RCL
019	00	00	061	91	R/S	103	02	02
020	95	=	062	43	RCL	104	95	=
021	42	STO	063	00	00	105	42	STO
022	03	03	064	65	×	106	03	03
023	91	R/S	065	43	RCL	107	91	R/S
024	43	RCL	066	01	01	108	43	RCL
025	00	00	067	65	×	109	00	00
026	65	×	068	02	2	110	65	×
027	02	2	069	95	=	111	43	RCL
028	65	×	070	42	STO	112	01	01
029	43	RCL	071	07	07	113	65	×
030	01	01	072	75	-	114	06	6
031	95	=	073	43	RCL	115	95	=
032	42	STO	074	06	06	116	42	STO
033	04	04	075	95	=	117	04	04
034	43	RCL	076	94	+/-	118	43	RCL
035	00	00	077	42	STO	119	02	02
036	65	×	078	07	07	120	65	×
037	02	2	079	91	R/S	121	02	2
038	65	×	080	76	LBL	122	95	=
039	43	RCL	081	12	B	123	85	+
040	02	02	082	42	STO	124	43	RCL
041	95	=	083	00	00	125	04	04

Fig. 8-12. It took 529 steps to write the program for the TR-59.

126	95	=	168	04	04	210	00	00
127	42	STO	169	95	=	211	04	4
128	05	05	170	42	STO	212	55	÷
129	91	R/S	171	04	04	213	03	3
130	43	RCL	172	89	π	214	95	=
131	04	04	173	65	×	215	42	STO
132	91	R/S	174	02	2	216	03	03
133	76	LBL	175	95	=	217	65	×
134	13	C	176	65	×	218	89	π
135	42	STO	177	43	RCL	219	95	=
136	00	00	178	00	00	220	42	STO
137	91	R/S	179	65	×	221	03	03
138	42	STO	180	43	RCL	222	43	RCL
139	02	02	181	02	02	223	00	00
140	43	RCL	182	85	+	224	65	×
141	00	00	183	43	RCL	225	43	RCL
142	33	X^2	184	04	04	226	00	00
143	65	×	185	95	=	227	65	×
144	89	π	186	42	STO	228	43	RCL
145	95	=	187	04	04	229	00	00
146	42	STO	188	91	R/S	230	65	×
147	01	01	189	43	RCL	231	43	RCL
148	43	RCL	190	00	00	232	03	03
149	02	02	191	33	X^2	233	95	=
150	65	×	192	65	×	234	42	STO
151	43	RCL	193	89	π	235	01	01
152	01	01	194	65	×	236	91	R/S
153	95	=	195	02	2	237	89	π
154	42	STO	196	95	=	238	65	×
155	03	03	197	75	-	239	04	4
156	91	R/S	198	43	RCL	240	95	=
157	43	RCL	199	04	04	241	65	×
158	00	00	200	95	=	242	43	RCL
159	33	X^2	201	94	+/-	243	00	00
160	42	STO	202	42	STO	244	33	X^2
161	04	04	203	05	05	245	95	=
162	89	π	204	91	R/S	246	42	STO
163	65	×	205	76	LBL	247	02	02
164	02	2	206	14	D	248	58	FIX
165	95	=	207	58	FIX	249	00	00
166	65	×	208	02	02	250	91	R/S
167	43	RCL	209	42	STO	251	76	LBL

252	15	E	294	95	=	336	00	00
253	42	STO	295	42	STO	337	33	X²
254	00	00	296	03	03	338	65	×
255	91	R/S	297	61	GTO	339	02	2
256	42	STO	298	53	(	340	95	=
257	01	01	299	76	LBL	341	75	-
258	91	R/S	300	23	LNX	342	43	RCL
259	42	STO	301	87	IFF	343	06	06
260	02	02	302	00	00	344	95	=
261	76	LBL	303	44	SUM	345	94	+/-
262	38	SIN	304	43	RCL	346	55	÷
263	29	CP	305	00	00	347	03	3
264	43	RCL	306	33	X²	348	95	=
265	01	01	307	55	÷	349	42	STO
266	32	X:T	308	02	2	350	03	03
267	04	4	309	95	=	351	76	LBL
268	67	EQ	310	65	×	352	53	(
269	22	INV	311	06	6	353	43	RCL
270	29	CP	312	95	=	354	02	02
271	43	RCL	313	55	÷	355	65	×
272	01	01	314	03	3	356	43	RCL
273	32	X:T	315	95	=	357	03	03
274	06	6	316	42	STO	358	95	=
275	67	EQ	317	07	07	359	42	STO
276	23	LNX	318	61	GTO	360	03	03
277	29	CP	319	45	Y×	361	91	R/S
278	43	RCL	320	76	LBL	362	61	GTO
279	01	01	321	35	1/X	363	54	)
280	32	X:T	322	87	IFF	364	76	LBL
281	08	8	323	00	00	365	45	Y×
282	67	EQ	324	28	LOG	366	43	RCL
283	35	1/X	325	43	RCL	367	07	07
284	76	LBL	326	00	00	368	65	×
285	22	INV	327	65	×	369	43	RCL
286	87	IFF	328	02	2	370	02	02
287	00	00	329	93	.	371	95	=
288	34	FX	330	05	5	372	42	STO
289	43	RCL	331	95	=	373	03	03
290	00	00	332	33	X²	374	91	R/S
291	33	X²	333	42	STO	375	76	LBL
292	55	÷	334	06	06	376	54	)
293	03	3	335	43	RCL	377	43	RCL

Fig. 8-12. Continued from page 85.

378	02	02	420	02	02	462	42	STO
379	65	×	421	65	×	463	00	00
380	43	RCL	422	03	3	464	91	R/S
381	03	03	423	95	=	465	02	2
382	95	=	424	42	STO	466	01	1
383	42	STO	425	04	04	467	43	RCL
384	03	03	426	85	+	468	00	00
385	86	STF	427	43	RCL	469	33	X^2
386	00	00	428	07	07	470	65	×
387	61	GTO	429	95	=	471	89	π
388	38	SIN	430	42	STO	472	65	×
389	76	LBL	431	05	05	473	43	RCL
390	34	FX	432	61	GTO	474	01	01
391	43	RCL	433	39	COS	475	95	=
392	00	00	434	76	LBL	476	55	÷
393	65	×	435	28	LOG	477	03	3
394	43	RCL	436	43	RCL	478	95	=
395	02	02	437	00	00	479	42	STO
396	65	×	438	65	×	480	02	02
397	02	2	439	43	RCL	481	91	R/S
398	95	=	440	02	02	482	43	RCL
399	42	STO	441	65	×	483	00	00
400	04	04	442	04	4	484	33	X^2
401	43	RCL	443	95	=	485	42	STO
402	00	00	444	42	STO	486	04	04
403	33	X^2	445	04	04	487	89	π
404	85	+	446	85	+	488	65	×
405	43	RCL	447	43	RCL	489	02	2
406	04	04	448	02	02	490	65	×
407	95	=	449	95	=	491	43	RCL
408	42	STO	450	42	STO	492	04	04
409	05	05	451	05	05	493	95	=
410	91	R/S	452	76	LBL	494	42	STO
411	43	RCL	453	39	COS	495	04	04
412	04	04	454	43	RCL	496	89	π
413	91	R/S	455	05	05	497	65	×
414	76	LBL	456	91	R/S	498	02	2
415	44	SUM	457	43	RCL	499	65	×
416	43	RCL	458	04	04	500	43	RCL
417	00	00	459	91	R/S	501	00	00
418	65	×	460	76	LBL	502	65	×
419	43	RCL	461	10	E'	503	43	RCL

```
 01♦LBL "PPG 030"
 02 "3 DIMENSION"
      03 AVIEW
      04 PSE

      05♦LBL "V1"
      06 FIX 0
07 "RECTANG. BOX"
      08 AVIEW
      09 PSE
    10 "LENGTH ?"
      11 PROMPT
      12 STO 00
     13 "WIDTH ?"
      14 PROMPT
      15 STO 01
    16 "HEIGHT ?"
      17 PROMPT
      18 STO 02
      19 RCL 01
      20 *
      21 RCL 00
      22 *
      23 STO 03
      24 "CUBIC="
      25 ARCL 03
      26 AVIEW
      27 STOP
      28 RCL 00
      29 2
      30 *
      31 RCL 01
      32 *
      33 STO 04
      34 RCL 00
      35 2
      36 *
      37 RCL 02
      38 *
      39 STO 05
      40 RCL 01
      41 2
      42 *
      43 RCL 02
      44 *
      45 RCL 04
      46 +
      47 RCL 05
      48 +
      49 STO 06
50 "SURFACE AREA"
      51 AVIEW
      52 PSE
      53 "SQ.="
      54 ARCL 06
      55 AVIEW
      56 STOP
 57 "WALLS ONLY"
      58 AVIEW
      59 PSE
      60 RCL 00
      61 RCL 01
      62 *
      63 2
      64 *
      65 STO 07
      66 RCL 06
      67 -
      68 CHS
      69 STO 07
      70 "SQ.="
      71 ARCL 07
      72 AVIEW
      73 STOP

     74♦LBL "V2"
      75 FIX 0
    76 "HEXAGON"
      77 AVIEW
      78 PSE
   79 "ONE SIDE"
      80 PROMPT
      81 STO 00
      82 X↑2
      83 2
      84 /
      85 6
      86 *
      87 STO 02
  88 "HEIGHT ?"
      89 PROMPT
      90 STO 01
      91 RCL 02
      92 *
      93 STO 03
      94 "VOLUME"
      95 AVIEW
      96 PSE
      97 "CUBIC="
      98 ARCL 03
      99 AVIEW
     100 STOP
     101 RCL 00
     102 RCL 01
     103 *
     104 6
     105 *
     106 STO 04
     107 RCL 02
     108 2
     109 *
     110 RCL 04
     111 +
     112 STO 05
113 "SURFACE AREA"
     114 AVIEW
     115 PSE
     116 "SQ.="
     117 ARCL 05
     118 AVIEW
     119 STOP
120 "WALLS ONLY"
     121 AVIEW
```

Fig. 8-13. The 443-step program for the HP-41C.

```
122 PSE
123 "SQ.="
124 ARCL 04
125 AVIEW
126 STOP
127♦LBL "V3"
128 "CYLINDER"
129 AVIEW
130 PSE
131 "RADIUS"
132 PROMPT
133 STO 00
134 X↑2
135 PI
136 *
137 STO 01
138 "HEIGHT"
139 PROMPT
140 STO 02
141 RCL 01
142 *
143 STO 03
144 "VOLUME"
145 AVIEW
146 PSE
147 "CUBIC="
148 ARCL 03
149 AVIEW
150 STOP
151 "SURFACE AREA"
152 AVIEW
153 PSE
154 RCL 00
155 X↑2
156 STO 04
157 PI
158 2
159 *
160 RCL 04
161 *
162 STO 04
163 PI
164 2
165 *
166 RCL 00
167 *
168 RCL 02
169 *
170 RCL 04
171 +
172 STO 04
173 "SQ.="
174 ARCL 04
175 AVIEW
176 STOP
177 "WALL ONLY"
178 AVIEW
179 PSE
180 RCL 00
181 X↑2
182 PI
183 *
184 2
185 *
186 RCL 04
187 -
188 CHS
189 STO 05
190 "SQ.="
191 ARCL 05
192 AVIEW
193 STOP
194♦LBL "V4"
195 "SPHERE"
196 AVIEW
197 PSE
198 "RADIUS ?"
199 PROMPT
200 STO 00
201 4
202 3
203 /
204 STO 03
205 PI
206 *
207 STO 03
208 RCL 00
209 RCL 00
210 *
211 RCL 00
212 *
213 RCL 03
214 *
215 STO 01
216 FIX 2
217 "VOLUME"
218 AVIEW
219 PSE
220 "CUBIC="
221 ARCL 01
222 AVIEW
223 STOP
224 "SURFACE AREA"
225 AVIEW
226 PSE
227 PI
228 4
229 *
230 RCL 00
231 X↑2
232 *
233 STO 02
234 FIX 0
235 "SQ.="
236 ARCL 02
237 AVIEW
238 STOP
```

```
239*LBL "V5"
240 "PYRAMID"
241 AVIEW
242 PSE
243 CF 00
244 "BASE SIDE ?"
245 PROMPT
246 STO 00
247 "NO. OF SIDES?"
248 PROMPT
249 STO 01
250 "HEIGHT ?"
251 PROMPT
252 STO 02

253*LBL E
254 RCL 01
255 ENTER↑
256 4
257 X=Y?
258 GTO A
259 RCL 01
260 ENTER↑
261 6
262 X=Y?
263 GTO B
264 RCL 01
265 ENTER↑
266 8
267 X=Y?
268 GTO C

269*LBL A
270 FS? 00
271 GTO F
272 RCL 00
273 X↑2
274 3
275 /
276 STO 03
277 GTO D

278*LBL B
279 FS? 00
280 GTO G
281 RCL 00
282 X↑2
283 2
284 /
285 6
286 *
287 3
288 /
289 STO 07
290 GTO D

291*LBL C
292 FS? 00
293 GTO H
294 RCL 00
295 2.5
296 *
297 X↑2
298 STO 06
299 RCL 00
300 X↑2
301 2
302 *
303 RCL 06
304 -
305 CHS
306 3
307 /
308 STO 03

309*LBL D
310 RCL 02
311 *
312 STO 03
313 "VOLUME:"
314 AVIEW
315 PSE
316 FIX 0

317 "CUBIC="
318 ARCL 03
319 AVIEW
320 STOP
321 "SURFACE AREA"
322 AVIEW
323 PSE
324 SF 00
325 GTO E
326*LBL F
327 RCL 00
328 RCL 02
329 *
330 2
331 *
332 STO 04
333 RCL 00
334 X↑2
335 +
336 STO 05
337 GTO I

338*LBL G
339 RCL 00
340 RCL 02
341 *
342 3
343 *
344 STO 04
345 RCL 07
346 +
347 STO 05
348 GTO I

349*LBL H
350 RCL 00
351 RCL 02
352 *
353 4
354 *
355 STO 04
```

```
356 RCL 06
357 +
358 STO 05

359*LBL I
360 "SQ.="
361 ARCL 05
362 AVIEW
363 STOP
364 "WALLS ONLY"
365 AVIEW
366 PSE
367 "SQ.="
368 ARCL 04
369 AVIEW
370 STOP

371*LBL "V6"
372 "CIRC. CONE"
373 AVIEW
374 PSE

375 "BASE RADIUS?"
376 PROMPT
377 STO 00
378 "HEIGHT ?"
379 PROMPT
380 STO 01
381 RCL 00
382 X↑2
383 PI
384 *
385 RCL 01
386 *
387 3
388 /
389 STO 02
390 "VOLUME"
391 AVIEW
392 PSE
393 "CUBIC="
394 ARCL 02
395 AVIEW
396 STOP

397 "SURFACE AREA"
398 AVIEW
399 PSE
400 RCL 00
401 X↑2
402 STO 04
403 PI
404 2
405 *
406 RCL 04
407 *
408 STO 04
409 PI
410 2
411 *
412 RCL 00
413 *
414 RCL 01
415 *
416 RCL 04
417 +
418 3
419 /
420 STO 03
421 "SQ.="
422 ARCL 03
423 AVIEW
424 STOP

425 "WALL ONLY"
426 AVIEW
427 PSE
428 RCL 00
429 X↑2
430 PI
431 *
432 RCL 03
433 -
434 CHS
435 STO 03
436 "SQ.="
437 ARCL 03
438 AVIEW
439 STOP
440 "END"
441 AVIEW
442 STOP
443 .END.
```

Fig. 8-13. (Continued from page 89).

The routine continues along lines which are similar to those described for the HP. When the computer finally gets to step 385 it encounters **STF 00** which commands it to set flag 0. When the computer goes back to the sections which earlier asked **IFF 00**, the answer is affirmative and the computer then goes to the alternate sections of the program, which determine the surface area and the walls-only surface area (sections **LBL** $\sqrt{\mathbf{x}}$, **LBL SUM**, and **LBL LOG**).

Entering this program into the computer from magnetic cards requires two cards and three passes. But the computer will reject the program and

504	01	01	515	91	R/S	526	94	+/-
505	95	=	516	43	RCL	527	42	STO
506	85	+	517	00	00	528	03	03
507	43	RCL	518	33	X^2	529	91	R/S
508	04	04	519	65	×	530	00	0
509	95	=	520	89	π	531	00	0
510	55	÷	521	95	=	532	00	0
511	03	3	522	75	-	533	00	0
512	95	=	523	43	RCL	534	00	0
513	42	STO	524	03	03	535	00	0
514	03	03	525	95	=			

Fig. 8-13. It took 529 steps to write the program for the TI-59. (Continued from page 91.)

reply with a flashing *0* in the display unless you precede entering the cards by these keystrokes: **3 2nd Op 1 7**, which partitions the registers within the computer in such a manner that it can accept as many as 720 program steps, and 30 data storage registers remain available. After passing the key sequence the display will read *719.29*. Then press **CLR** and the computer is ready to accept the program from the magnetic cards.

HP-41C. When running the program with the HP comply with any displayed request by keying in the requested data and pressing **R/S**, and the computer will display the results in the same consecutive order shown above. It will always identify the forthcoming result by displaying such words as *VOLUME, SURFACE AREA,* or *WALLS ONLY* (Fig. 8-12).

All sections of the program are straight-forward. The computer simply makes the applicable computations once it is given the required input data. The only exception is the section dealing with the pyramid, where the computer can go in three directions depending on whether the pyramid has four, six or eight sides. This is accomplished in each of the computers by series of steps which compare the contents of the X register with those of another (T in the TI, Y in the HP), and which then tell the computer which way to go.

In the HP program the pyramid section is labeled V5 and consists of steps 239 through 370. Of this, the subsection labeled E, steps 253 through 268 is the section where the computer initially must decide which way to go, depending on the number of sides to the pyramid (four sides, steps 253 through 258, go to section A; six sides, steps 259 through 263, go to section B; eight sides, steps 264 through 268, go to section C).

Each of these next subsections, **A, B** and **C** start with the question, "Has flag number 00 been set?" (**FS? 00**). If the answer is yes, the computer must next go to **F, G,** or **H**, depending on whether it is coming out of section **A, B,** or **C.** The first time the computer faces these sections after completing section **E**, no flag has been set and it will skip the **GTO F, GTO G** or **GTO H** command and continue through the subsequent steps.

These next steps are used to calculate the volume, which is then displayed as the computer executes the commands contained in section D.

After having displayed the cubic measure for the volume, the computer puts *SURFACE AREA* into the display to tell us that will be the next display. You will notice that the instruction in step 324 reads SF 00, which stands for set flag 00. The computer is then told to go back once more to section E and go through that selection process (four, six, or eight sides). It is again told to go to sections **A, B,** or **C.** But this time, as soon as it gets to that new section, it is asked about the flag (**FS? 00**) and since the flag has been set the computer follows instructions and goes to sections **F, G** or **H.**

These are the sections which compute the surface area and the area covered by the base of the pyramid, permitting the computer to display the total surface area and the walls-only surface area in quick succession.

Program Steps

TI-59

Cards: 2
Passes: 3
Partition: 719 29

For Rectangular Box:
1. Enter length of side 1
2. Press **A**
3. Enter length of side 2
4. Press **R/S**
5. Enter height
6. Press **R/S**
Display: Cubic measure
7. Press **R/S**
Display: Surface area
8. Press **R/S**
Display: Area of walls only

For Hexagon:
1. Enter length of one side
2. Press **B**
3. Enter height
4. Press **R/S**
Display: Cubic measure
5. Press **R/S**
Display: Surface area
6. Press **R/S**
Display: Area of walls only

For Cylinder:
1. Enter radius
2. Press **C**
3. Enter height
4. Press **R/S**
Display: Cubic measure
5. Press **R/S**
Display: Surface area
6. Press **R/S**
Display: Area of walls only

For Sphere:
1. Enter radius
2. Press **D**
Display: Cubic measure
3. Press **R/S**
Display: Surface area

For Pyramid:
1. Enter length of one side at base
2. Press **E**
3. Enter number of sides
4. Press **R/S**
5. Enter height
6. Press **R/S**
Display: Cubic measure
7. Press **R/S**
Display: Surface area
8. Press **R/S**
Display: Area of walls only

For Circular Cone:
1. Enter radius at base
2. Press **2nd E'**
3. Enter height
4. Press **R/S**
Display: Cubic measure
5. Press **R/S**
Display: Surface area
6. Press **R/S**
Display: Area of walls only

HP-41C

Cards: 5
Passes: 9
Size: 006

For Rectangular Box:
1. Press **shift A** (top row, 1st key)
Display: *LENGTH?*
2. Enter length: **R/S**
Display: *WIDTH?*
3. Enter width; **R/S**
Display: *HEIGHT?*
4. Enter height; **R/S**
Display: *CUBIC*= and measure
5. Press **R/S**
Display: *SURFACE AREA*, then *SQ*= and surface are in square measure
6. Press **R/S**
Display: *WALLS ONLY*, then *SQ*= and measure

For Hexagon:
1. Press **shift B** (top row, 2nd key)
Display: *ONE SIDE?*
2. Enter length of one side; **R/S**
Display: *HEIGHT?*
3. Enter height; **R/S**
Display: *VOLUME*, then *CUBIC*=measure
4. Press **R/S**
Display: *SURFACE AREA*, then *SQ*=
5. Press **R/S**
Display: *WALLS ONLY*, then *SQ*=

For Cylinder:
1. Press **shift C** (top row, 3rd key)
Display: *RADIUS?*
2. Enter radius; **R/S**
Display: *HEIGHT?*
3. Enter height; **R/S**
Display: *VOLUME*, then *CUBIC*=
4. Press **R/S**
Display: *SURFACE AREA*, then *SQ*=
5. Press **R/S**
Display: *WALL ONLY*, then *SQ*=

For Sphere:
1. Press **shift D** (top row, 4th key)
Display: *RADIUS?*
2. Enter radius; **R/S**
Display: *VOLUME*, then *CUBIC*=
3. Press **R/S**
Display: *SURFACE AREA*, then *SQ*=

For Pyramid:
1. Press **shift E** (top row, 5th key)
Display: *BASE SIDE?*
2. Enter length of side at base; **R/S**
Display: *NO. OF SIDES?*
3. Enter number of sides (4,6 or 8); **R/S**
Display: *HEIGHT?*
4. Enter height; **R/S**
Display: *VOLUME*, then *CUBIC*=
5. Press **R/S**
Display: *SURFACE AREA*, then *SQ*=
6. Press **R/S**
Display: *WALLS ONLY*, then *SQ*=

For Circular Cone:
1. Press **shift F** (2nd row, 1st key)
Display: *BASE RADIUS?*
2. Enter radius at base· **R/S**
Display: *HEIGHT?*
3. Enter height; **R/S**
Display: *VOLUME*, then
CUBIC=
4. Press **R/S**
Display: *SURFACE AREA*, then
SQ=
5. Press **R/S**
Display: *WALL ONLY*, then
SQ=

ARITHMETIC INVOLVING TIME OR DEGREES

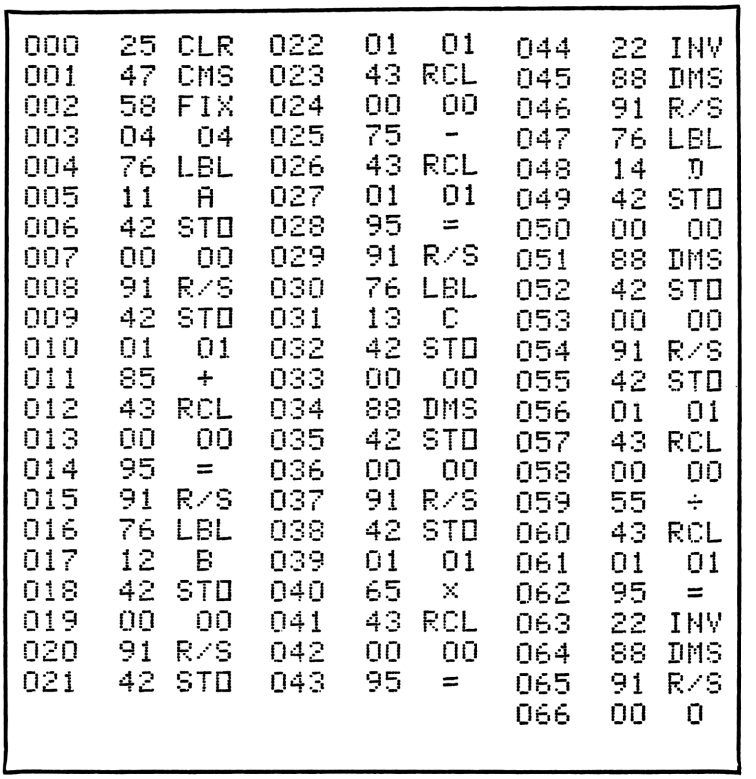
Since an hour has 60 minutes rather than 100, and a circle has 360 degrees, using either of these concepts in solving mathematical problems presents some difficulties.

It must be understood that the problem for time and degrees is the same. An hour is composed of 3600 seconds, and a circle is composed of 360 degrees. In order to use this data in addition, subtraction, and multiplication, it is often necessary to convert these figures to their decimal equivalents. The result must then be converted back to the H.MMSS or D.MMSS (hours:minutes:seconds or degrees:minutes:seconds) format. Most computers have provisions for these conversions as part of their built-in functions. This program takes advantage of these functions to perform four kinds of arithmetic with ease.

TI-59. Here are the four sections of the program (Fig. 8-14) are assigned to four of the letter keys: **A** for addition, **B** for subtraction, **C** for multiplication and **D** for division.

If we key in **5.4317** and press **A**, that figure will be in the display and the computer is now ready to add the next figure, for example: 3 hours, two minutes and 41 seconds. We key in **3.0241** and press **R/S** and the result is *8.4558*.

Now let's key in **5.4317** again but this time press **B**. (First press **RST R/S** to clear the register of unwanted data.) Key in **3.0241** and press **R/S** and the result is *2.4076*. Since there is no such thing as 76 seconds, we know this should have been 2.4116. To accomplish that change, two steps have to be added to the program, steps which are currently missing: These steps are **2nd D.MS** followed by **INV 2nd D.MS** (in the HP they would be **XEQ ALPHA HR ALPHA, XEQ ALPHA HMS ALPHA**). These steps transform 2.4076 into its decimal equivalent (2.6878) and the second step changes that back into the H.MMSS format 2.4116.



```
000  25 CLR   022  01  01   044  22 INV
001  47 CMS   023  43 RCL  045  88 DMS
002  58 FIX   024  00  00   046  91 R/S
003  04  04   025  75  -    047  76 LBL
004  76 LBL   026  43 RCL  048  14  D
005  11  A    027  01  01   049  42 STO
006  42 STO   028  95  =    050  00  00
007  00  00   029  91 R/S  051  88 DMS
008  91 R/S   030  76 LBL  052  42 STO
009  42 STO   031  13  C    053  00  00
010  01  01   032  42 STO  054  91 R/S
011  85  +    033  00  00   055  42 STO
012  43 RCL   034  88 DMS  056  01  01
013  00  00   035  42 STO  057  43 RCL
014  95  =    036  00  00   058  00  00
015  91 R/S   037  91 R/S  059  55  ÷
016  76 LBL   038  42 STO  060  43 RCL
017  12  B    039  01  01   061  01  01
018  42 STO   040  65  ×    062  95  =
019  00  00   041  43 RCL  063  22 INV
020  91 R/S   042  00  00   064  88 DMS
021  42 STO   043  95  =    065  91 R/S
                             066  00  0
```

Fig. 8-14. The Time/Degree arithmetic program for the TI-59.

Key in **5.4317** once more (after pressing **RST R/S**) and press **C**. This time a new figure shows up in the display, *5.7214*, the decimal equivalent of the figure which we keyed in (it is not possible to multiply or divide time or degree data without first converting it into its decimal equivalent). Now key in **17.83** and press **R/S**. The result is displayed as *102.0045*, this time in the time format.

Now for the last exercise, let's press **RST R/S** and key in **5.4317** and then press **D**. Again the decimal equivalent, *5.7214*, appears in the display. Divide that by, say **2.79** and the result will be *2.0303* or two hours, three minutes and three seconds.

HP-41C. Initiating the program (Fig. 8-15) produces its identification, *TIME ARITHM*, followed by *FUNCTION?* followed by *ADD 0=YES*. If we want to add two values, then we press **0** followed by **R/S**. But if we don't want to add we will press **R/S**. What happens is this: In step 10 we have placed a 9 into the X register (it can be any number other than a zero). When the computer asks in step 13 if the value in the X register is equal to zero, the answer is no (unless we pressed **0** after being asked if we wanted to). If the answer had been yes, the computer would have executed step 14 and

gone to **LBL A** which is the section of the program which performs the addition.

But, since we want to multiply, the answer to *X=0?* was no and the computer went on to step 15 which asks the same question about subtraction. The routine is repeated and the next question in the display will be *MULTIP. 0=YES*. We now press **0** and **R/S** and the display comes up with *TIME: H/M/S*, which means we want to know the time value to be multiplied. Enter **5.4317**, which stands for five hours, 43 minutes and 17 seconds. The display now asks the *FACTOR*, requesting the figure to

```
      01♦LBL "TQ"
      02 FIX 4
 03 "TIME ARITHM."
      04 AVIEW
      05 PSE

      06♦LBL "T"
  07 "FUNCTION ?"
      08 AVIEW
      09 PSE
      10 9
   11 "ADD 0=YES"
      12 PROMPT
      13 X=0?
      14 GTO A
 15 "SUBTR. 0=YES"
      16 PROMPT
      17 X=0?
      18 GTO "S"
19 "MULTIP. 0=YES"
      20 PROMPT
      21 X=0?
      22 GTO "M"
 23 "DIVIDE 0=YES"
      24 PROMPT
      25 X=0?
      26 GTO D

      27♦LBL "M"
  28 "TIME: H/M/S"
      29 PROMPT
      30 HR
      31 STO 00
      32 "FACTOR"
      33 PROMPT
      34 STO 01
      35 RCL 00
      36 *
      37 HMS

      38♦LBL "R"
      39 STO 02
      40 "H/M/S="
      41 ARCL 02
      42 AVIEW
      43 STOP
      44 GTO "Q"

      45♦LBL D
 46 "TIME: H/M/S"
      47 PROMPT
      48 HR
      49 STO 00
      50 "FACTOR"
      51 PROMPT
      52 STO 01
      53 RCL 00
      54 RCL 01
      55 /
      56 HMS
      57 GTO "R"

      58♦LBL A
      59 "TIME: H/M/S"
          60 PROMPT
          61 STO 00
   62 "FACTOR:H/M/S"
          63 PROMPT
          64 STO 01
          65 +
          66 GTO "R"

          67♦LBL "S"
    68 "TIME: H/M/S"
          69 PROMPT
          70 STO 00
   71 "FACTOR:H/M/S"
          72 PROMPT
          73 STO 01
          74 -
          75 GTO "R"

          76♦LBL "Q"
    77 "AGAIN 0=NO"
          78 PROMPT
          79 X=0?
          80 GTO "Z"
          81 GTO "T"

          82♦LBL "Z"
          83 "END"
          84 AVIEW
          85 STOP
          86 .END.
```

Fig. 8-15. This program performs arithmetic calculations using time or degree data on the HP-41C.

multiply by. Let's really make it difficult and enter **17.83**. The display instantly comes up with 102.0045 or 102 hours, no minutes and 45 seconds. Actually, the display will be *H/M/S=102.0045*. If we use that figure and run the program again, this time using division, we'd be entering *102.0045*; reply to *FACTOR* again with **17.83** and the result will show up as *H/M/S=5.4317*.

After the result has been displayed, press **R/S** which, produces the question *AGAIN? 0=NO*. Now, pressing **R/S** will start the program over with the question *FUNCTION*. But pressing **0** followed by **R/S** will *END* the display.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

For Addition:
1. Enter time or degree value in H.MMSS (or D.MMSS) format
2. Press **A**
3. Enter amount to be used in addition
4. Press **R/S**

Display: Result in H.MMSS or D.MMSS

For Subtraction:
Same, but use **B** key

For Multiplication:
Same, but use **C** key

For Division:
Same, but use **D** key

Passes: 3
Size: 003
Display: *FUNCTION?*
ADD 0=YES:
SUBTR. 0=YES; MULTIPL. 0=YES. DIVIDE 0= YES;

1. When appropriate function is in display, press **0** and **R/S**

Display: *TIME: H/M/S*

2. Enter time (or degrees) in H/MMSS (or D.MMSS) format; **R/S**

Display: *FACTOR?*

3. Enter factor to be used in calculation; **R/S**

Display: The result in H.MMSS or D.MMSS

All four functions operate in the identical manner.

HP-41C

Cards: 2

FRACTIONS

This is a simple program that is handy when you must use fractions in mathematical calculations. Just try to figure out what the result of the following might be:

$$2.78 \times \frac{7}{9} + \left(\frac{11}{17} \; / \frac{3}{5} \right) = ?$$

Obviously, the only way to do something like this is to change those

TI-59	HP-41C
000 CMS	01 LBL PPG 031
001 CLR	02 FRACTIONS
002 FIX	03 AVIEW
003 04	04 PSE
004 LBL	05 LBL 01
005 A	06 NUMBER
006 STO	07 PROMPT
007 00	08 STO 00
008 R/S	09 DIVIDE BY ?
009 STO	10 PROMPT
010 01	11 STO 01
011 RCL	12 RCL 00
012 00	13 RCL 01
013 ÷	14 /
014 RCL	15 STO 02
015 01	16 DECIMAL:
016 =	17 AVIEW
017 STO	18 PSE
018 02	19 VIEW 02
019 R/S	20 STOP
	21 GTO 01
	22 END
01: Is simply an identification for my files	000: Clears all memories

Fig. 8-16. The program steps for Divide By.

fractional values into decimals. The short program is given in Fig. 8-16. Here is what happens when we try to solve the above calculation.

TI-59. Turn the computer on and feed the program card into the card-reader slot. The display shows *0.0000*. Press **7**, then **A** and the display reads *7.0000*.

Press **9** and **R/S** and the display shows the result: *0.7778*. For the next calculation enter **11**, press **A**, then **17**, then **R/S** and the display responds with *0.6471*. Using the same sequence of steps the final result, 0.6000 will be in display.

With these figures, it is relatively easy to perform the calculation. Key in **2.78**, followed by these keystrokes: × **0.7778** =+ (*0.6471* ÷ **0.6**) = *3.2408*.

HP-41C. Press **ON** to turn the computer on. Press ÷, the user-defined key and start the program. The display, *FRACTIONS* changes to *NUMBER*. Enter **7**, press **R/S**. The display reads *DIVIDE BY?* Enter **9**, **R/S** and the display reads *DECIMAL* and changes to *0.7778*.

Press **R/S** and the program starts over again. The other results are 0.6471 and 0.6000.

With these figures, it is relatively easy to perform the calculations. Find the result for the data inside the parentheses: **0.6471 ENTER 0.6 /**, which results in a display of *1.0785*. Store this in memory (**STO 00**). Now enter the balance: **2.78 ENTER 0.7778 × RCL 00 +**, and the display shows: *3.2408*.

There is a simple way to reduce the length of this program, which is important if it is to be kept in the HP along with a number of other programs. As it is now, steps 11, 12, 14, and 15 in the TI and steps 12 and 13 in the HP are needed to place the two values into the appropriate consecutive order for the required division. The consecutive order is important when you divide, but it is of no consequence when you multiply. We can save extra steps that recall data from memory by multiplying by using the reciprocal of the value used for division. Both computers have a single key (**1/×**) which produces the reciprocal of the number in display.

To convert $\frac{7}{9}$ to the decimal equivalent, instead of dividing 7 by 9, we multiply 7 by the reciprocal of 9 which is 0.1111 and the result is again 0.7778. The two programs are given in Fig. 8-17.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 003
1. Enter principal number of fraction	1. Enter principal number of fraction
2. Press **A**	2. Press **R/S**
3. Enter below-the-line number of fraction	3. Enter below-the-line number of fraction
4. Press **R/S**	4. Press **R/S**
Display: Decimal equivalent	Display: *DECIMAL:*
	Display: Decimal equivalent

DIVIDE BY?

The purpose of this program is to figure out if one number can be divided by another number without a fraction remainder. Enter a number, then enter another number, and press **R/S**. If the first number can be divided by the second number, the HP will display *YES* and the TI will display *555555*. If it cannot be divided evenly by that number the HP will display **NO** and the TI will show a flickering *99999999999 99*.

There are times when it isn't easy to know the answer by doing calculations in the head. For instance, pick a year with an even number and find out whether it is, was or is going to be a leap year: 1980: Yes; 1776 YES; 1066: NO. (Actually they didn't have leap years before the year 1582—not a leap year—when the Gregorian calendar was adopted by the Western nations.)

TI-59	HP-41C
000 CMS	01 PPG 031
001 CLR	02 FRACTIONS
002 FIX	03 AVIEW
003 04	04 PSE
004 LBL	05 LBL 01
005 A	06 NUMBER
006 STO	07 PROMPT
007 00	08 STO 00
008 R/S	09 DIVIDE BY ?
009 STO	10 PROMPT
010 01	11 STO 01
011 1/x	12 1/x
012 x	13 RCL 00
013 RCL	14 x
014 00	15 STOP
015 =	16 GTO 01
016 R/S	

Fig. 8-17. Program steps for Divide By, Second Method.

Figures 8-18 and 8-19 show the two programs.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

1. Enter number to be divided
2. Press **A**
3. Enter number by which to divide

```
000  25 CLR   015  01  01   030  05  5
001  47 CMS   016  95  =    031  05  5
002  58 FIX   017  22 INV   032  05  5
003  00  00   018  59 INT   033  05  5
004  76 LBL   019  29 CP    034  91 R/S
005  11  A    020  42 STO   035  76 LBL
006  42 STO   021  02  02   036  13  C
007  00  00   022  67  EQ    037  58 FIX
008  91 R/S   023  12  B    038  09  09
009  42 STO   024  61 GTO   039  01  1
010  01  01   025  13  C    040  55  ÷
011  43 RCL   026  76 LBL   041  00  0
012  00  00   027  12  B    042  95  =
013  55  ÷    028  05  5    043  91 R/S
014  43 RCL   029  05  5    044  00  0
```

Fig. 8-18. The TI-59 program designed to determine the divisibility of a number.

```
01♦LBL "/"      14 /            27 AVIEW
02 "DIVIDE"     15 FRC          28 STOP
03 AVIEW        16 STO 02       29♦LBL "W"
04 PSE          17 X=0?         30 1
05♦LBL "V"      18 GTO "Z"   31 "AGAIN 0=NO"
06 "NUMBER"     19 GTO "Y"      32 PROMPT
07 PROMPT       20♦LBL "Z"      33 X=0?
08 STO 00       21 "  YES"      34 GTO "X"
09 "DIVIDE BY"  22 AVIEW        35 GTO "V"
10 PROMPT       23 STOP         36♦LBL "X"
11 STO 01       24 GTO "W"      37 "END"
12 RCL 00                       38 AVIEW
13 RCL 01       25♦LBL "Y"      39 STOP
                26 "  NO"       40 .END.
```

Fig. 8-19. The HP-41C program designed to determine whether a number is divisible without developing fractions.

TI-59

4. Press **R/S**
Display: Either *5555555* in a steady display or *99.9999999* in a flashing display. The first means that the first figure can be divided by the second without leaving a fraction. The second display means that there will be a fraction left over

HP-41C

Cards: 1
Passes: 2
Size: 003
Display: *NUMBER*
1. Enter number to be divided; **R/S**
Display: *DIVIDE BY*
2. Enter number by which to divide; **R/S**
Display: *YES* or *NO*, depending on whether the first number can be divided by the second, leaving no fractions

ENDLESS LOOP

One capability both computers has is the ability to perform a given series of calculations over and over again. The computer runs in circles, performing one or a series of calculations during each full circle, changing the data input each time. This is very useful, as we'll see when we examine some of the more complicated programs. For the moment, the purpose of this program is to demonstrate how such a loop works and why it is possible to "get hung up in a loop."

The program, illustrated in Figs. 8-20 and 8-21 consists of four separate parts. Each part will perform a certain function; addition, subtraction, multiplication, or division. I could have added squaring and taking square roots, but they tend to produce such astronomically large or small results

```
000  25 CLR  024  13  C   048  95  =
001  47 CMS  025  42 STO 049  42 STO
002  76 LBL  026  00  00  050  00  00
003  11  A   027  76 LBL 051  66 PAU
004  42 STO  028  35 1/X 052  66 PAU
005  01  01  029  75  -   053  61 GTO
006  91 R/S  030  43 RCL 054  34 fX
007  76 LBL  031  01  01  055  76 LBL
008  12  B   032  95  =   056  15  E
009  42 STO  033  42 STO 057  42 STO
010  00  00  034  00  00  058  00  00
011  76 LBL  035  66 PAU 059  76 LBL
012  23 LNX  036  66 PAU 060  33 X²
013  85  +   037  61 GTO 061  55  ÷
014  43 RCL  038  35 1/X 062  43 RCL
015  01  01  039  76 LBL 063  01  01
016  95  =   040  14  D   064  95  =
017  42 STO  041  42 STO 065  42 STO
018  00  00  042  00  00  066  00  00
019  66 PAU  043  76 LBL 067  66 PAU
020  66 PAU  044  34 fX  068  66 PAU
021  61 GTO  045  65  ×   069  61 GTO
022  23 LNX  046  43 RCL 070  33 X²
023  76 LBL  047  01  01  071  00  0
```

Fig. 8-20. The Endless Loop program for the TI-59.

that a few turns through the loop would produce a number as meaningless as the national debt.

To run the program, put in a base number and a factor number. The base number is the one which we want to change by using the factor number, in repeated addition, subtraction, multiplication or division. In each case, the program is designed to display the result from each calculation for a brief period. This is accomplished by a double **PAU** (pause) command in the TI-59, while in the HP-41C we use **VIEW 00** and **PSE** (pause). This lets us see the result long enough to decide whether to go on with the program or stop it.

Stop the program by pressing **R/S** for either computer. With both, the display returns to the factor number, so we can repeat the experiment with a different factor number. This way we can simply punch in a new factor number, press **R/S** and the program will take off again.

With the TI all we need do is key in a new factor number, press **A**, and then use any of the other four letter keys to determine the type of calculation we want.

Here is what happens with the two computers:

TI-59. Turn the computer on and feed in the program card. The display will show *1* when the card has been read. Press **CLR** and the display changes to *0*. Now the program is ready to go. If the computer has been used for calculations prior to starting the program, it is always a good idea to play safe and press **RST R/S**, which takes the program through steps 000 and 001, designed to remove all left over data from any memories.

Now enter the factor number and press **A**. The number is now in display Fig. 8-22. Key in the base number and press the letter key which coincides with the type of calculation you want (**B**= addition; **C**= subtraction; **D**= multiplication; **E**= division). The computer will display the successive results and will continue to do so indefinitely until you press **R/S** or the computer is shut off, in which case the program is lost until the magnetic card is again fed into the card-reader slot.

HP-41C. Press **ON** and start the program by pressing the appropriate user-defined key. The display shows *ENDLESS LOOP*, then changes to *FACTOR?* Enter the factor number and press **R/S**.

```
      01♦LBL "LL"
      02 0
      03 STO 00
04 "ENDLESS LOOP"
      05 AVIEW
      06 PSE
    07 "FACTOR ?"
      08 PROMPT
      09 STO 01
      10 1
   11 "ADD 0=YES"
      12 PROMPT
      13 X=0?
     14 GTO "LL2"
15 "SUBTR. 0=YES"
      16 PROMPT
      17 X=0?
     18 GTO "LL3"
19 "MULTIP. 0=YES"
      20 PROMPT
      21 X=0?
     22 GTO "LL4"
23 "DIVIDE 0=YES"
      24 PROMPT
      25 X=0?
     26 GTO "LL5"
     27♦LBL "LL2"
28 "BASE NO. ?"
      29 PROMPT
      30 STO 00
     31♦LBL "LL6"
      32 RCL 01
      33 ST+ 00
      34 VIEW 00
      35 PSE
     36 GTO "LL6"
     37♦LBL "LL3"
38 "BASE NO. ?"
      39 PROMPT
      40 STO 00
     41♦LBL "LL7"
      42 RCL 01
      43 ST- 00
      44 VIEW 00
      45 PSE
     46 GTO "LL7"
     47♦LBL "LL4"
48 "BASE NO. ?"
      49 PROMPT
      50 STO 00
     51♦LBL "LL8"
      52 RCL 01
      53 ST* 00
      54 VIEW 00
      55 PSE
     56 GTO "LL8"
     57♦LBL "LL5"
58 "BASE NO. ?"
      59 PROMPT
      60 STO 00
     61♦LBL "LL9"
      62 RCL 01
      63 ST/ 00
      64 VIEW 00
      65 PSE
     66 GTO "LL9"
      67 .END.
```

Fig. 8-21. Four endless-loop programs combined in one for the HP-41C.

```
5.  STO  5.  STO  5.  STO  5.  STO
      1        1        1        1
5.       5.       5.       5.
    R/S      R/S      R/S      R/S
5.  STO  5.  STO  5.  STO  5.  STO
5.       5.       5.       5.
5.  +    5.  -    5.  ×    5.  ÷
5.  RCL  5.  RCL  5.  RCL  5.  RCL
      1        1        1        1
5.       5.       5.       5.
5.  =    5.  =    5.  =    5.  =
10.      0.       25.      1.
10.  STO 0.  STO  25.  STO  1.  STO
      0        0        0        0
10.      0.       25.      1.
10.  GTO 0.  GTO  25.  GTO  1.  GTO
     LNX      1/X      √X       X=
10.  -   0.  +    25.  ×    1.  =
10.  RCL 0.  RCL  25.  RCL  1.  RCL
      1        1        1        1
5.   5.       5.       5.
5.   =   5.   =   5.   =   5.   =
15.      -5.      125.     0.2
15.  STO -5.  STO  125.  STO  0.2  STO
      0        0        0        0
15.      -5.      125.     0.2
15.  GTO -5.  GTO  125.  GTO  0.2  GTO
     LNX      1/X      √X       X2
15.  +   -5.  -    125.  x    0.2  ÷
15.  RCL -5.  RCL  125.  RCL  0.2  RCL
      1        1        1        1
5.       5.       5.       5.
5.   =   5.   =   5.   =   5.   =
20.      -10.     625.     0.04
20.  STO -10. STO  625.  STO  0.04 STO
      0        0        0        0
20.      -10.     625.     0.04
20.  GTO -10. GTO  625.  GTO  0.04 GTO
     LNX      1/X      √X       X2
20.  +   -10. -    625.  ×    0.04 ÷
20.  RCL -10. RCL  625.  RCL  0.04 RCL
      1        1        1        1
5.       5.       5.       5.
5.   =   5.   =   5.   =   5.   =
25.      -15.     3125.    0.008
25.  STO -15. STO  3125.  STO  0.008 STO
      0        0        0        0
25.      -15.     3125.    0.008
25.  GTO -15. GTO  3125.  GTO  0.008 GTO
     LNX      1/X      √X       X2
25.  +   -15. -    3125.  ×    0.008 ÷
25.  RCL -15. RCL  3125.  RCL  0.008 RCL
      1        1        1        1
5.       5.       5.       5.
5.   =   5.   =   5.   =   5.   =
30.      -20.     15625.   0.0016
```

Fig. 8-22. The printouts tracing the actions of the TI-59 when running the four versions of the Endless Loop program.

```
          01♦LBL "LL"
                   0
               STO 00
        "ENDLESS LOOP"
               AVIEW
ENDLESS LOOP
                 PSE
          "FACTOR ?"
               PROMPT
FACTOR ?
        5.0000    RUN
               STO 01
                   1
           "ADD 0=YES"
               PROMPT
ADD 0=YES
        0.0000    RUN
               X=0?
           GTO "LL2"

          27♦LBL "LL2"
```

```
                "BASE NO. ?"
                     PROMPT
BASE NO. ?
              5.0000    RUN
                     STO 00
          31♦LBL "LL6"
                     RCL 01
              5.0000    ***
                     ST+ 00
                    VIEW 00
                   10.0000
                       PSE
                 GTO "LL6"

          31♦LBL "LL6"
                     RCL 01
              5.0000    ***
                     ST+ 00
                    VIEW 00
                   15.0000
```

```
                       PSE
                GTO "LL6"

          31♦LBL "LL6"
                     RCL 01
              5.0000    ***
                     ST+ 00
                    VIEW 00
                   20.0000
                       PSE
                 GTO "LL6"

          31♦LBL "LL6"
                     RCL 01
              5.0000    ***
                     ST+ 00
                    VIEW 00
                   25.0000
                       PSE
                 GTO "LL6"
```

Fig. 8-23. The printout tracing the action of the HP-41C when running the additional portion of the Endless Loop.

The display gives us a chance to decide on the type of calculation we want to perform. It asks *ADD 0=YES*. This means that if we want addition, press **0** and **R/S**. If we don't simply press **R/S** and the display changes to: *SUBTR. 0=YES*.

The same routine is followed for multiplication and division. This is one of the simplest ways the computer can prompt the operator to make a decision. When the decision has been made and **0** followed by **R/S** has been pressed, the display asks *BASE NO.?* Enter it, press **R/S** and the program is off and running and will continue to do so until you press **R/S** or press **OFF** (Figs. 8-23, 8-24, 8-25 and 8-26).

The accompanying illustrations show how the computers go through the same routine over and over again. In this example I have purposely designed a loop that never stops on its own accord. If this loop were included in a longer and more complicated program, it would simply sit there and go around and around with the display indicating by a flickering C in the TI or that erratically moving bird figure in the HP that it is busy computing. When that happens, the program itself must be stopped and it is up to the operator to take it out of the hung-up loop condition.

You would never want to do this on purpose in a program, but it does occasionally happen by accident, and can become a frustrating problem,

especially if no printer is available to determine what went wrong where. Whenever you use a loop as part of a program, a means must be provided to permit the program to get out of that loop. Usually this is done by comparing each successive sum with a fixed number, and when the result of the calculation is "equal or greater" or "equal or lesser" (depending on the type of calculation being used) than that number, the next instruction would be a GTO (go to) instructor in the HP or simply a certain label in the TI. The computer would accept that new command and be disentangled from the loop.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

1. Enter number to be used as factor in repeated calculations
2. Press **A**

```
                01♦LBL "LL"
                         0
                    STO 00
             "ENDLESS LOOP"
                    AVIEW
ENDLESS LOOP
                       PSE
                 "FACTOR ?"
                    PROMPT
FACTOR ?
              5.0000    RUN
                    STO 01
                         1
               "ADD 0=YES"
                    PROMPT
ADD 0=YES
                       RUN
                     X=0?
            "SUBTR. 0=YES"
                    PROMPT
SUBTR. 0=YES
              0.0000    RUN
                     X=0?
                  GTO "LL3"
              37♦LBL "LL3"
               "BASE NO. ?"
                    PROMPT
BASE NO. ?
```

```
5.0000    RUN
      STO 00

 41♦LBL "LL7"
      RCL 01
5.0000    ***
      ST- 00
     VIEW 00
      0.0000
         PSE
   GTO "LL7"

 41♦LBL "LL7"
      RCL 01
5.0000    ***
      ST- 00
     VIEW 00
     -5.0000
         PSE
   GTO "LL7"

 41♦LBL "LL7"
      RCL 01
5.0000    ***
```

```
     ST- 00
    VIEW 00
   -10.0000
       PSE
  GTO "LL7"

 41♦LBL "LL7"
      RCL 01
5.0000    ***
     ST- 00
    VIEW 00
   -15.0000
       PSE
  GTO "LL7"

 41♦LBL "LL7"
      RCL 01
5.0000    ***
     ST- 00
    VIEW 00
   -20.0000
       PSE
  GTO "LL7"
```

Fig. 8-24. The printout tracing the actions of the HP-41C when running the subtraction portion of the Endless Loop program.

TI-59

3. Enter base number to be used in conjunction with the factor number
4. Press **B** for add
   C for subtract
   D for multiply
   E for divide

Display: One after another, the results of repeated calculations involving the factor and base numbers appear in the display
5. Press **R/S** to stop the loop

HP-41C

Cards: 2

Passes: 3
Size: 002
Display: *FACTOR?*
1. Enter number to be used
2. Press **R/S**
Display: *ADD 0=YES;*
R/S: *SUBTR. 0=YES*
R/S: *MULTIP. 0=YES*
R/S: *DIVIDE 0=YES*
3. For desired function, press **0** then **R/S**
Display: *BASE NO.?*
4. Enter number to be used in conjunction with the original number (step 1)
5. **R/S**

```
01♦LBL "LL" MULTIP. 0=YES
                  0
            STO 00
      "ENDLESS LOOP"
             AVIEW
ENDLESS LOOP
               PSE
        "FACTOR ?"
      PROMPT BASE NO. ?
FACTOR ?
      5.0000    RUN
            STO 01
                  1
        "ADD 0=YES"
            PROMPT
ADD 0=YES
               RUN
             X=0?
     "SUBTR. 0=YES"
            PROMPT
SUBTR. 0=YES
               RUN
             X=0?
   "MULTIP. 0=YES"
            PROMPT
    0.0000    RUN
             X=0?
        GTO "LL4"
47♦LBL "LL4"
    "BASE NO. ?"
          PROMPT
    5.0000    RUN
          STO 00
51♦LBL "LL8"
          RCL 01
5.0000    ***
          ST* 00
         VIEW 00
     25.0000
           PSE
        GTO "LL8"
51♦LBL "LL8"
          RCL 01
5.0000    ***
          ST* 00
         VIEW 00
    125.0000
           PSE
        GTO "LL8"
51♦LBL "LL8"
          RCL 01
5.0000    ***
          ST* 00
         VIEW 00
    625.0000
           PSE
        GTO "LL8"
51♦LBL "LL8"
          RCL 01
5.0000    ***
          ST* 00
         VIEW 00
  3,125.0000
           PSE
        GTO "LL8"
```

Fig. 8-25. The printout tracing of the action of the HP-41C when running the multiplication section of the Endless Loop program.

```
                XEQ "LL"                      PROMPT   61♦LBL "LL9"
                          MULTIP. 0=YES                      RCL 01
             01♦LBL "LL"                        RUN 5.0000     ***
                      0                        X=0?        ST/ 00
                  STO 00            "DIVIDE 0=YES"       VIEW 00
          "ENDLESS LOOP"                      PROMPT       0.2000
                   AVIEW DIVIDE 0=YES                         PSE
ENDLESS LOOP                        0.0000    RUN      GTO "LL9"
                     PSE                      X=0?
               "FACTOR ?"               GTO "LL5"   61♦LBL "LL9"
                  PROMPT                                     RCL 01
FACTOR ?                             57♦LBL "LL5" 5.0000     ***
            5.0000    RUN             "BASE NO. ?"        ST/ 00
                  STO 01                   PROMPT       VIEW 00
                       1 BASE NO. ?                       0.0400
              "ADD 0=YES"            5.0000    RUN           PSE
                  PROMPT                   STO 00      GTO "LL9"
ADD 0=YES
                     RUN             61♦LBL "LL9"   61♦LBL "LL9"
                   X=0?                    RCL 01         RCL 01
           "SUBTR. 0=YES"            5.0000     *** 5.0000     ***
                  PROMPT                   ST/ 00         ST/ 00
SUBTR. 0=YES                             VIEW 00        VIEW 00
                     RUN                   1.0000         0.0080
                   X=0?                      PSE            PSE
         "MULTIP. 0=YES"                 GTO "LL9"      GTO "LL9"
```

Fig. 8-26. The printout of the action of the HP-41C when running the divide section of the Endless Loop program.

HP-41C

Display: One after another, the results of repeated calculations appear in the display

6. **R/S** stops loop

AVERAGES

Averaging of a number of figures is not particularly complicated; it's simply a nuisance. This program simplifies the procedure.

Enter all the data, add them all together, and divide the total. The program does this for us, and at the same time tells us how many individual figures were used in the averaging process.

TI-59. The program (Fig. 8-27) for the TI uses the **A** and the **B** keys; enter the values one after the other, pressing **A** after each entry. By doing so, two things being accomplished: the entries are added to one another and the last total will appear in the display, and the number of entries is counted and stored in an appropriate register for future use.

When all entries have been made, press **B** and the first display will show the number of entries which have been made. Press **R/S** and the

grand total of the entries will appear in the display. Press **R/S** again and the average will be displayed.

Steps 000 and 001 (Fig. 8-27) are used to clear all leftover data out of the computer. To make sure that actually takes place, always start this (and all) programs in the TI with **RST R/S**. Failing that, the computer would skip the first two steps.

Steps 002 through 015 collect and add the value entries in R_{00}, accomplished by using **SUM 00** (steps 004 and 005) instead of the more cumbersome **STO 00 + RCL 01 = STO 01** which would have accomplished the same result.

Steps 006 through 012 place the number of entries into R_{01} by adding 1 when each entry is made.

Steps 013 and 014 recall the contents of R_{00} into the display. Thus, when **A** is pressed after an entry, the sum up to that point appears in the display.

Steps 016 through 027 cause the number of entries now stored in R_{03} to appear in the display when key **B** is pressed.

Steps 028 through 032 recall the contents of R_{00}, the total sum of the entries, into the display.

Steps 033 through 039 divide that last displayed value by the number of entries and then, upon pressing **R/S**, displays the result which is the average.

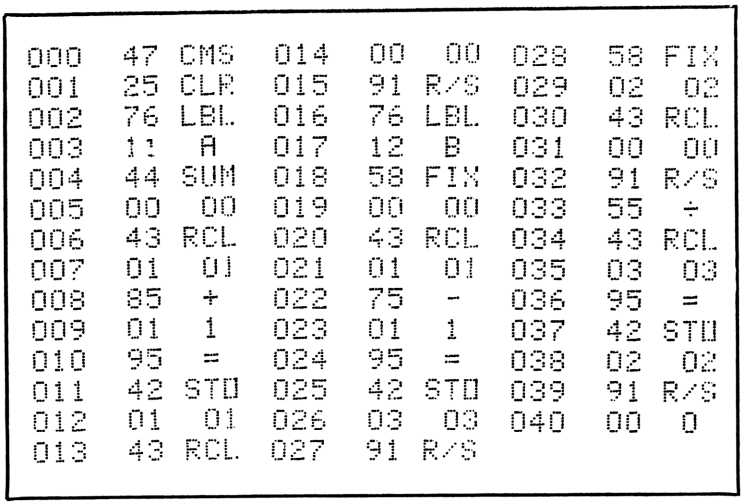
HP-41C. Once initialized, the display, (Fig. 8-28) identifies the program with *AVERAGES* and then requests *ENTER VALUE*. Enter **87.05** and press **R/S**. The display again says *ENTER VALUE*. In this manner we enter all the values involved. Let's enter: **15.2, 87.03, 0.147, 125** and **11.9**. Then, when the display again requests *ENTER VALUE*, press **0** to inform the computer that is all. Press **R/S** and the computer tells us *NO. ENTRIES= 6*. Pressing **R/S** produces *TOTAL= 326.33*, and pressing **R/S** again results in *AVERAGE= 54.39*.

Steps 01 through 05 identify the program and step 05 (**CLRG**) clears any data leftover in the registers. Steps 06 through 12 enter the keyed-in value in registers R_{03} and R_{00}. R_{03} simply retains the last entry, while R_{00} adds it to all previous entries. Steps 11 and 12 add a 1 to whatever is in register R_{01}. Thus, R_{01} keeps track of the number of entries made, while R_{00} keeps the total of those entries.

Step 13 recalls the last entry into the X register to compare it with 0. If the last entry was 0, the computer goes to **LBL 01**. If anything other than 0 was entered, it goes back to **LBL 00** for additional entries.

Steps 18 through 21 in **LBL 01** reduce the number of entries by one, because the computer did count the last entry of a 0 as a legitimate entry, which it was not (it was simply the signal that all entries had, in fact, been made), and stores the result again in R_{01}. In steps 22 through 25 the total of all values is divided by the number of entries and the result is stored in R_{02}.

Steps 27 through 39 are used to display the results, first the number of entries, then the total of the values entered, and then the average.



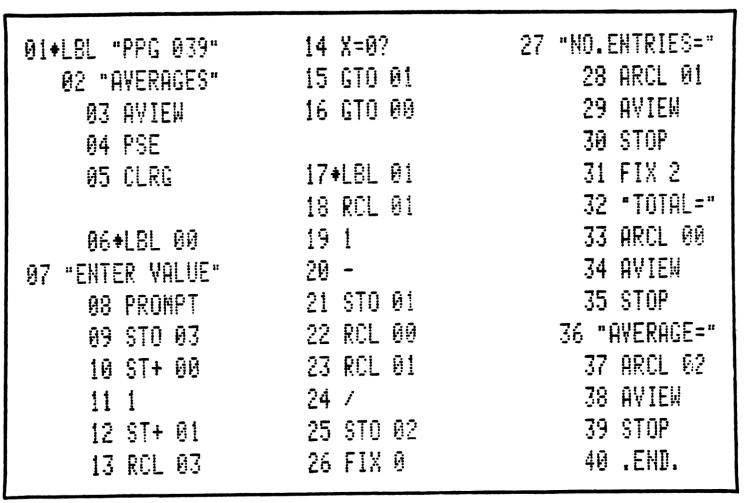
```
000   47 CMS   014   00   00   028   58 FIX
001   25 CLR   015   91 R/S   029   02   02
002   76 LBL   016   76 LBL   030   43 RCL
003   11   A   017   12   B   031   00   00
004   44 SUM   018   58 FIX   032   91 R/S
005   00   00   019   00   00   033   55   ÷
006   43 RCL   020   43 RCL   034   43 RCL
007   01   01   021   01   01   035   03   03
008   85   +   022   75   -   036   95   =
009   01   1   023   01   1   037   42 STO
010   95   =   024   95   =   038   02   02
011   42 STO   025   42 STO   039   91 R/S
012   01   01   026   03   03   040   00   0
013   43 RCL   027   91 R/S
```

Fig. 8-27. The TI-59 version of Averages.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter value
2. Press **A**
3. Repeat steps 1 and 2 to enter as many values as are to be averaged
4. Press **B**
Display: The number of entries
5. Press **R/S**
Display: Grand total of all entries
6. Press **R/S**



```
01*LBL "PPG 039"      14 X=0?          27 "NO.ENTRIES="
   02 "AVERAGES"      15 GTO 01           28 ARCL 01
     03 AVIEW         16 GTO 00           29 AVIEW
     04 PSE                               30 STOP
     05 CLRG          17*LBL 01           31 FIX 2
                      18 RCL 01           32 "TOTAL="
     06*LBL 00        19 1                33 ARCL 00
07 "ENTER VALUE"      20 -                34 AVIEW
     08 PROMPT        21 STO 01           35 STOP
     09 STO 03        22 RCL 00         36 "AVERAGE="
     10 ST+ 00        23 RCL 01           37 ARCL 02
     11 1             24 /                38 AVIEW
     12 ST+ 01        25 STO 02           39 STOP
     13 RCL 03        26 FIX 0            40 .END.
```

Fig. 8-28. The Averages program, for the HP-41C.

TI-59

Display: The average

HP-41C

Cards: 1
Passes: 1
Size: 004
Display: *ENTER VALUE*

1. Enter value; **R/S**
Display: *ENTER VALUE*
2. Repeat the above as often as desired. When last entry has been made, press **0** and **R/S**
Display: *NO ENTRIES=* and number
3. Press **R/S**
Display: *TOTAL=* all entries
4. Press **R/S**
Display: *AVERAGE=* and average

LIMITED RHUMBLINE

Here we ask the computer to tell us not only the distance between two points, but also the direction from one point to the other in degrees.

Let's assume the two points are New York, New York, and Houston, Texas. The accuracy of the distance reply will be degraded somewhat by the fact that Houston lies below the 30th parallel, but it won't be enough to make any serious difference.

Enter **40.38** for the latitude and **73.46** for the longitude at New York City. Then enter **29.57** and **95.21** for the latitude and longitude respectively at Houston. When the computer gets through computing, it will display *MILES=1,360* (or simply *1360* in the TI). When we press **R/S** (or **2nd E'** in the TI) the display tells us *DIR.DEGR.=238* (or simply *238*) in the TI-59), representing the straight-line direction from New York City to Houston (Figs. 8-29 and 8-30). (By the way, you might have noticed that when you enter the coordinates into the TI and then press the letter key, the display is at variance from the input data. For instance, keying in **29.57** and pressing **C** results in a display of *29.95*. The reason, of course, is that the computer converts the input to the decimal equivalent, and 29 degrees and 57 minutes are equal to 29.95 degrees.)

This program is limited to areas between the 30th and 50th parallel, with accuracy decreasing the farther we venture from the 35th parallel.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter start latitude
2. Press **A**
3. Enter start longitude
4. Press **B**
5. Enter destination latitude
6. Press **C**
7. Enter destination longitude
8. Press **D**
9. Press **E**
Display: Distance
10. Press **R/S**
Display: Degrees of direction

HP-41C

Cards: 2

Passes: 3
Size: 017
Display: *START LAT.?*
1. Start latitude; **R/S**
Display: **START LONG.?**
2. Enter start longitude; **R/S**
Display: *DES. LAT.?*
3. Enter destination latitude; **R/S**
Display: *DEST. LONG.?*
4. Enter destination longitude; **R/S**
Display: *MILES=* and distance
5. Press **R/S**
Display: *DIR.DEGR.* = and direction in degrees

RHUMBLINE II

This program is designed to provide distance information with great accuracy from the equator to the 70th parallel. If asked to compute data for a point above that parallel, the computer will respond with *NOT POSSIBLE*. This is not the completely scientific approach to the subject used in rhumbline programs designed for the long distance operation of aircraft, but it is simpler to dissect than one of those overly obscure mathematical formulas.

TI-59. The program for the TI-59 (Fig. 8-31) is a carbon copy of that for the HP, using 387 steps rather than 260, but each section is the same as described in that section.

The program works like this: Enter the starting latitude and press **A**. Then enter the starting longitude and press **B**. Next the destination latitude followed by **C**, and the destination longitude followed by **D**. Pressing **E** will, after some computing time, produce the distance figure in the display, and pressing **R/S**, will change that to the direction in degrees.

HP-41C. Upon initialization the program (Fig. 8-32) will display *DIRCT DIST.* followed by *START LAT.?* Let's enter the coordinates for Santa Fe, **35.37** for the latitude and **106.03** for the longitude. Use the coordinates for Los Angeles for the destination, **33.56** and **118.26**, and the computer replies with *MILES=735*. Pressing **R/S** results in *DIR.DEGR.=245*.

The program includes a routine which permits the computer to calculate distance and direction information for two points which lie on either side of the international dateline. (It can't deal with two points on either side of the equator. It can be used in the northern or southern hemisphere, but not simultaneously in both.)

Let's make the computer calculate the distance and direction information for Los Angeles and Nagoya, Japan. The coordinates for Nagoya are approximately 35.31 and 135.25, but this latter is an Eastern longitude and it must therefore be entered as – 135.25. Key in **33.56** and **118.26** for Los Angeles, and then **35.31** and **135.25** followed by **CHS**. The display reads *MILES=5,711* and after that *DIR.DEGR.=271*.

Steps 01 through 04 identify the program.

Steps 05 through 10 accept and store the starting latitude data.

```
000  47 CMS   040  01  01   080  10  10
001  25 CLR   041  75  -    081  43 RCL
002  76 LBL   042  43 RCL   082  05  05
003  11  A    043  03  03   083  65  ×
004  58 FIX   044  95  =    084  43 RCL
005  02  02   045  50 I×I   085  10  10
006  88 DMS   046  42 STO   086  95  =
007  42 STO   047  06  06   087  42 STO
008  01  01   048  43 RCL   088  11  11
009  91 R/S   049  01  01   089  43 RCL
010  76 LBL   050  85  +    090  11  11
011  12  B    051  43 RCL   091  32 X:T
012  88 DMS   052  03  03   092  43 RCL
013  42 STO   053  95  =    093  08  08
014  02  02   054  55  ÷    094  22 INV
015  91 R/S   055  02  2    095  37 P/R
016  76 LBL   056  95  =    096  42 STO
017  13  C    057  42 STO   097  13  13
018  88 DMS   058  07  07   098  58 FIX
019  42 STO   059  43 RCL   099  00  00
020  03  03   060  06  06   100  32 X:T
021  91 R/S   061  65  ×    101  42 STO
022  76 LBL   062  06  6    102  12  12
023  14  D    063  07  7    103  91 R/S
024  88 DMS   064  95  =    104  76 LBL
025  42 STO   065  42 STO   105  10 E'
026  04  04   066  08  08   106  43 RCL
027  91 R/S   067  43 RCL   107  02  02
028  76 LBL   068  07  07   108  32 X:T
029  15  E    069  65  ×    109  43 RCL
030  43 RCL   070  93  .    110  04  04
031  02  02   071  03  3    111  67  EQ
032  75  -    072  08  8    112  28 LOG
033  43 RCL   073  95  =    113  38 SIN
034  04  04   074  75  -    114  76 LBL
035  95  =    075  06  6    115  28 LOG
036  50 I×I   076  07  7    116  86 STF
037  42 STO   077  95  =    117  00  00
038  05  05   078  50 I×I   118  43 RCL
039  43 RCL   079  42 STO   119  02  02
```

Fig. 8-29. The TI-59 version of Rhumbline.

```
120  32 X:T
121  43 RCL
122  04  04
123  77  GE
124  39 COS
125  30 TAN
126  76 LBL
127  39 COS
128  02  2
129  07  7
130  00  0
131  42 STO
132  14  14
133  87 IFF
134  00  00
135  33 X²
136  23 LNX
137  76 LBL
138  30 TAN
139  09  9
140  00  0
141  42 STO
142  15  15
143  87 IFF
144  00  00
145  34 ΓX
146  54  )
147  76 LBL
148  38 SIN
149  86 STF
150  00  00
151  43 RCL
152  04  04
153  32 X:T
154  43 RCL
155  02  02
156  77  GE
157  39 COS
158  30 TAN
159  76 LBL
160  33 X²
161  43 RCL
162  03  03
163  32 X:T
164  43 RCL
165  01  01
166  77  GE
167  35 1/X
168  45 Y^x
169  76 LBL
170  34 ΓX
171  43 RCL
172  03  03
173  32 X:T
174  43 RCL
175  01  01
176  77  GE
177  44 SUM
178  52 EE
179  76 LBL
180  45 Y^x
181  43 RCL
182  14  14
183  85  +
184  43 RCL
185  13  13
186  95  =
187  42 STO
188  16  16
189  50 I×I
190  91 R/S
191  76 LBL
192  35 1/X
193  43 RCL
194  13  13
195  75  -
196  43 RCL
197  14  14
198  95  =
199  42 STO
200  16  16
201  50 I×I
202  91 R/S
203  76 LBL
204  44 SUM
205  43 RCL
206  15  15
207  85  +
208  43 RCL
209  13  13
210  95  =
211  42 STO
212  16  16
213  50 I×I
214  91 R/S
215  76 LBL
216  52 EE
217  43 RCL
218  13  13
219  75  -
220  43 RCL
221  15  15
222  95  -
223  42 STO
224  16  16
225  50 I×I
226  91 R/S
227  76 LBL
228  23 LNX
229  43 RCL
230  14  14
231  91 R/S
232  76 LBL
233  54  )
234  43 RCL
235  15  15
236  91 R/S
237  00  0
```

```
      01*LBL "RL"
 02 "DIRECT DIST."
      03 AVIEW
      04 PSE

      05*LBL 00
 06 "START LAT. ?"
      07 PROMPT
      08 STO 01
      09 HR
      10 STO 01
11 "START LONG. ?"
      12 PROMPT
      13 STO 02
      14 HR
      15 STO 02
 16 "DEST. LAT. ?"
      17 PROMPT
      18 STO 03
      19 HR
      20 STO 03
21 "DEST. LONG. ?"
      22 PROMPT
      23 STO 04
      24 HR
      25 STO 04
      26 RCL 02
      27 RCL 04
      28 -
      29 ABS
      30 STO 05
      31 RCL 01
      32 RCL 03
      33 -
      34 ABS
      35 STO 06
      36 RCL 01
      37 RCL 03
      38 +
      39 2
      40 /
      41 STO 07
      42 RCL 06
      43 67
      44 *
      45 STO 08
      46 RCL 07
      47 0.38
      48 *
      49 67
      50 -
      51 ABS
      52 STO 10
      53 RCL 05
      54 RCL 10
      55 *
      56 STO 11
      57 RCL 08
      58 ENTER↑
      59 RCL 11
      60 R-P
      61 STO 12
      62 X<>Y
      63 STO 15
      64 FIX 0
      65 "MILES="
      66 ARCL 12
      67 AVIEW
      68 STOP
      69 RCL 02
      70 ENTER↑
      71 RCL 04
      72 X=Y?
      73 GTO A
      74 GTO B

     75*LBL A
      76 SF 00
      77 RCL 02
      78 ENTER↑
      79 RCL 04
      80 X>Y?
      81 GTO C
      82 GTO D

     83*LBL C
      84 270
      85 STO 13
      86 FS? 00
      87 GTO F
      88 GTO E

     89*LBL D
      90 90
      91 STO 14
      92 FS? 00
      93 GTO F
      94 GTO G

     95*LBL B
      96 RCL 02
      97 ENTER↑
      98 RCL 04
      99 X>Y?
     100 GTO C
     101 GTO D

    102*LBL E
     103 RCL 03
     104 ENTER↑
     105 RCL 01
     106 X>Y?
     107 GTO H
     108 GTO I
    109*LBL G
     110 RCL 03
     111 ENTER↑
     112 RCL 01
     113 X>Y?
     114 GTO J
```

Fig. 8-30. Rhumbline for the HP- 41C.

```
115 GTO "K"
116♦LBL I
117 RCL 13
118 RCL 15
119 +
120 STO 16
121 GTO F
122♦LBL H
123 RCL 15
124 RCL 13
125 -
126 STO 16
127 GTO F
128♦LBL J
129 RCL 14
130 RCL 15
131 +
132 STO 16
133 GTO F
134♦LBL "K"
135 RCL 15
136 RCL 14
137 -
138 STO 16
139 GTO F
140♦LBL F
141 RCL 16
142 ABS
143 STO 16
144 "DIR.DEGR.="
145 ARCL 16
146 AVIEW
147 STOP
148 "AGAIN? 0=YES"
149 PROMPT
150 X=0?
151 GTO 00
152 "END"
153 AVIEW
154 STOP
155 END
```

Steps 11 through 19 accept and store the starting longitude data. After storing the figure in R_{02} the program compares it to zero. If the figure is a negative figure (meaning an East longitude), it considers that figure to be smaller than zero and tells the computer to **GTO EE** where that negative figure is added to 360 before the computer goes on with the balance of the program. If it is not a negative figure, the computer goes directly to the next section.

Steps 20 through 25 accept and store the destination latitude data.

Steps 26 through 34 repeat what took place in steps 11 through 19.

Steps 35 through 52 relate the two sets of coordinates to one another, determining the number of latitudes and longitudes covered, and determine the average latitude for the entire distance being covered.

Steps 53 through 58 and steps 59 through 64 are the sections in which negative longitude data are turned into positive values which the computer can deal with.

Steps 65 through 137 determine at what latitudes the two points are located and what distance between longitudes is to be used for the problem.

Steps 138 through 142 place *NOT POSSIBLE* into the display and tell the computer to go on to the section which contains the *END* legend.

Steps 143 through 146 place the *END* legend into the display.

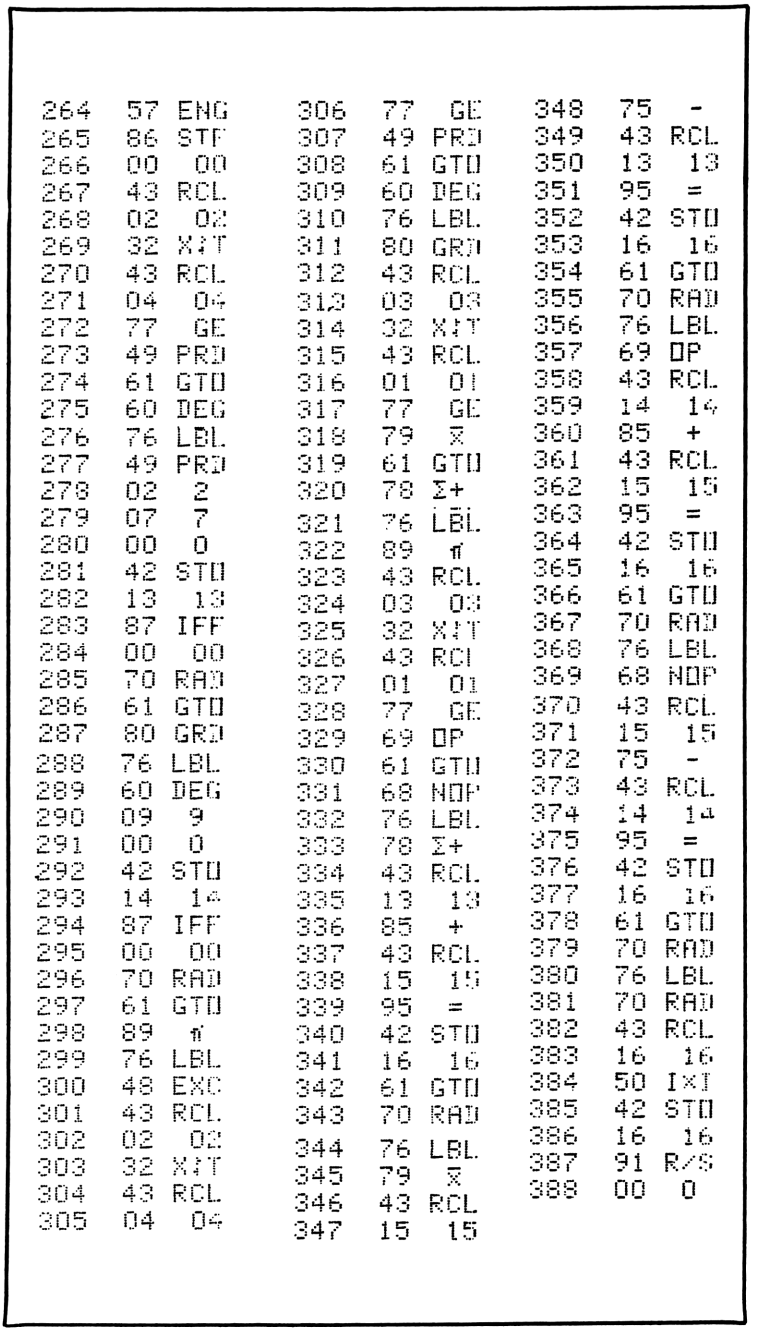
Steps 147 through 179 construct the theoretical rectangle with the diagonal line which determines the distance from point to point, and the degree in terms of direction. It then places the distance into the display as *MILES* = **xxx**. The degree figure must be related to 270 or 90 to be correct.

Steps 180 through 187 determine whether the direction is east-to-west or west-to-east.

```
000  47 CMS    044  76 LBL    088  42 STO
001  25 CLR    045  14  D     089  06  06
002  76 LBL    046  42 STO    090  43 RCL
003  11  A     047  04  04    091  01  01
004  42 STO    048  88 DMS    092  85  +
005  01  01    049  42 STO    093  43 RCL
006  88 DMS    050  04  04    094  03  03
007  42 STO    051  32 X:T    095  95  =
008  01  01    052  00  0     096  55  ÷
009  91 R/S    053  77  GE    097  02  2
010  76 LBL    054  38 SIN    098  95  =
011  12  B     055  43 RCL    099  42 STO
012  42 STO    056  04  04    100  07  07
013  02  02    057  91 R/S    101  32 X:T
014  88 DMS    058  76 LBL    102  01  1
015  42 STO    059  38 SIN    103  00  0
016  02  02    060  43 RCL    104  77  GE
017  32 X:T    061  04  04    105  39 COS
018  00  0     062  85  +     106  43 RCL
019  77  GE    063  03  3     107  07  07
020  28 LOG    064  06  6     108  32 X:T
021  43 RCL    065  00  0     109  02  2
022  02  02    066  95  =     110  00  0
023  91 R/S    067  42 STO    111  77  GE
024  76 LBL    068  04  04    112  30 TAN
025  28 LOG    069  91 R/S    113  43 RCL
026  43 RCL    070  76 LBL    114  07  07
027  02  02    071  15  E     115  32 X:T
028  85  +     072  43 RCL    116  03  3
029  03  3     073  02  02    117  00  0
030  06  6     074  75  -     118  77  GE
031  00  0     075  43 RCL    119  23 LN:
032  95  =     076  04  04    120  43 RCL
033  42 STO    077  95  =     121  07  07
034  02  02    078  50 I×I    122  32 X:T
035  91 R/S    079  42 STO    123  04  4
036  76 LBL    080  05  05    124  00  0
037  13  C     081  43 RCL    125  77  GE
038  42 STO    082  01  01    126  35 1/X
039  03  03    083  75  -     127  43 RCL
040  88 DMS    084  43 RCL    128  07  07
041  42 STO    085  03  03    129  32 X:T
042  03  03    086  95  =     130  05  5
043  91 R/S    087  50 I×I    131  00  0
```

Fig. 8-31. Rhumbline II as written for the TI-59.

132	77	GE
133	34	FX
134	43	RCL
135	07	07
136	32	X↕T
137	06	6
138	00	0
139	77	GE
140	33	X²
141	43	RCL
142	07	07
143	32	X↕T
144	07	7
145	00	0
146	77	GE
147	45	Y^x
148	76	LBL
149	39	COS
150	06	6
151	07	7
152	76	LBL
153	44	SUM
154	65	×
155	43	RCL
156	05	05
157	95	=
158	42	STO
159	10	10
160	61	GTO
161	52	EE
162	76	LBL
163	30	TAN
164	06	6
165	05	5
166	61	GTO
167	44	SUM
168	76	LBL
169	23	LNX
170	06	6
171	00	0
172	61	GTO
173	44	SUM
174	76	LBL
175	35	1/X
176	05	5
177	03	3
178	93	.
179	93	.
180	08	8
181	61	GTO
182	44	SUM
183	76	LBL
184	34	FX
185	04	4
186	09	9
187	61	GTO
188	44	SUM
189	76	LBL
190	33	X²
191	04	4
192	02	2
193	21	2ND
194	44	SUM
195	76	76
196	45	Y^x
197	03	3
198	05	5
199	61	GTO
200	44	SUM
201	76	LBL
202	53	(
203	01	1
204	55	÷
205	00	0
206	95	=
207	91	R/S
208	76	LBL
209	54	)
210	43	RCL
211	06	06
212	25	CLR
213	06	6
214	07	7
215	95	=
216	02	2
217	08	8
218	43	RCL
219	07	07
220	65	×
221	00	0
222	93	.
223	03	3
224	08	8
225	95	=
226	75	-
227	06	6
228	07	7
229	95	=
230	50	I×I
231	42	STO
232	10	10
233	76	LBL
234	52	EE
235	43	RCL
236	10	10
237	42	STO
238	11	11
239	32	X↕T
240	43	RCL
241	08	08
242	22	INV
243	37	P/R
244	42	STO
245	15	15
246	32	X↕T
247	42	STO
248	12	12
249	58	FIX
250	00	00
251	43	RCL
252	12	12
253	91	R/S
254	43	RCL
255	02	02
256	32	X↕T
257	43	RCL
258	04	04
259	67	EQ
260	57	ENG
261	61	GTO
262	48	EXC
263	76	LBL



```
264  57 ENG     306  77  GE     348  75  -
265  86 STF     307  49 PRD     349  43 RCL
266  00  00     308  61 GTO     350  13  13
267  43 RCL     309  60 DEG     351  95  =
268  02  02     310  76 LBL     352  42 STO
269  32 X↕T     311  80 GRD     353  16  16
270  43 RCL     312  43 RCL     354  61 GTO
271  04  04     313  03  03     355  70 RAD
272  77  GE     314  32 X↕T     356  76 LBL
273  49 PRD     315  43 RCL     357  69 OP
274  61 GTO     316  01  01     358  43 RCL
275  60 DEG     317  77  GE     359  14  14
276  76 LBL     318  79  x̄      360  85  +
277  49 PRD     319  61 GTO     361  43 RCL
278  02  2      320  78 Σ+      362  15  15
279  07  7      321  76 LBL     363  95  =
280  00  0      322  89  π      364  42 STO
281  42 STO     323  43 RCL     365  16  16
282  13  13     324  03  03     366  61 GTO
283  87 IFF     325  32 X↕T     367  70 RAD
284  00  00     326  43 RCL     368  76 LBL
285  70 RAD     327  01  01     369  68 NOP
286  61 GTO     328  77  GE     370  43 RCL
287  80 GRD     329  69 OP      371  15  15
288  76 LBL     330  61 GTO     372  75  -
289  60 DEG     331  68 NOP     373  43 RCL
290  09  9      332  76 LBL     374  14  14
291  00  0      333  78 Σ+      375  95  =
292  42 STO     334  43 RCL     376  42 STO
293  14  14     335  13  13     377  16  16
294  87 IFF     336  85  +      378  61 GTO
295  00  00     337  43 RCL     379  70 RAD
296  70 RAD     338  15  15     380  76 LBL
297  61 GTO     339  95  =      381  70 RAD
298  89  π      340  42 STO     382  43 RCL
299  76 LBL     341  16  16     383  16  16
300  48 EXC     342  61 GTO     384  50 I×I
301  43 RCL     343  70 RAD     385  42 STO
302  02  02     344  76 LBL     386  16  16
303  32 X↕T     345  79  x̄      387  91 R/S
304  43 RCL     346  43 RCL     388  00  0
305  04  04     347  15  15
```

Fig. 8-31. Rhumbline for the TI-41C. (Continued from page 119.)

Steps 188 through 193 relate the direction data to 270 degrees, because it is the section used only if the direction is from east to west.

Steps 194 through 199 perform the same function for west-to-east directions.

Steps 200 through 206 repeat steps 180 through 187 and might just as well be deleted from the program. I have left them in as a good example of how stray program steps sometimes remain in a program where they are not needed. These steps do not interfere with the execution of the program.

Steps 207 through 220 determine which of four possible variations are to be used to calculate the final result.

Steps 221 through 244 represent the four different possibilities, each ending with the command to **GTO F**.

Steps 245 through 260, the section labeled **LBL F**, place the direction into the display and ask whether you want to use the program again for another calculation, or whether you're done, in which case it will place *END* into the display.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 3
Passes: 2	Passes: 5
Partition: 479 59	Size: 019
1. Enter start latitude	Display: *START LAT.?*
2. Press **A**	1. Enter start latitude: **R/S**
3. Enter start longitude	Display: *START LONG.?*
4. Press **B**	2. Enter start longitude; **R/S**
5. Enter destination latitude	Display: *DEST. LAT.?*
6. Press **C**	3. Enter destination latitude; **R/S**
7. Press destination longitude	Display: *DEST. LONG?*
8. Press **D**	4. Enter destination longitude; **R/S**
9. Press **E**	Display: *MILES*= and distance
Display: Distance	5. Press **R/S**
10. Press **R/S**	Display: *DIR.DEGR.* = and direction in degrees
Display: Direction in degrees	

```
      01♦LBL "RL"
 02 "DIRECT DIST."
      03 AVIEW
      04 PSE

      05♦LBL 00
 06 "START LAT. ?"
      07 PROMPT
      08 STO 01
      09 HR
      10 STO 01
11 "START LONG. ?"
      12 PROMPT
      13 STO 02
      14 HR
      15 STO 02
      16 ENTER↑
      17 0
      18 X>Y?
      19 GTO "EE"

      20♦LBL "CC"
 21 "DEST. LAT. ?"
      22 PROMPT
      23 STO 03
      24 HR
      25 STO 03
26 "DEST. LONG. ?"
      27 PPOMPT
      28 STO 04
      29 HR
      30 STO 04
      31 ENTER↑
      32 0
      33 X>Y?
      34 GTO "FF"

      35♦LBL "DD"
      36 RCL 02
      37 RCL 04
      38 -
      39 ABS
      40 STO 05
      41 RCL 01
      42 RCL 03
      43 -
      44 ABS
      45 STO 06
      46 RCL 01
      47 RCL 03
      48 +
      49 2
      50 /
      51 STO 07
      52 GTO "L"

      53♦LBL "EE"
      54 RCL 02
      55 360
      56 +
      57 STO 02
      58 GTO "CC"
      59♦LBL "FF"
      60 RCL 04
      61 360
      62 +
      63 STO 04
      64 GTO "DD"

      65♦LBL "L"
      66 ENTER↑
      67 10
      68 X>Y?
      69 GTO 11
      70 GTO 12

      71♦LBL 12
      72 RCL 07
      73 ENTER↑
      74 20
      75 X>Y?
      76 GTO 13
      77 GTO 14

      78♦LBL 14
      79 RCL 07
      80 ENTER↑
      81 30
      82 X>Y?
      83 GTO 15
      84 GTO 16

      85♦LBL 16
      86 RCL 07
      87 ENTER↑
      88 40
      89 X>Y?
      90 GTO 17
      91 GTO 18

      92♦LBL 18
      93 RCL 07
      94 ENTER↑
      95 50
      96 X>Y?
      97 GTO 19
      98 GTO 20

      99♦LBL 20
     100 RCL 07
     101 ENTER↑
     102 60
     103 X>Y?
     104 GTO 21
     105 GTO 22

     106♦LBL 22
     107 RCL 07
     108 ENTER↑
     109 70
     110 X>Y?
     111 GTO 23
     112 GTO 24

     113♦LBL 11
     114 67

     115♦LBL "M"
     116 RCL 05
     117 *
     118 STO 10
     119 GTO 25

     120♦LBL 13
     121 65
     122 GTO "M"

     123♦LBL 15
     124 60
     125 GTO "M"

     126♦LBL 17
     127 53.8
     128 GTO "M"
```

Fig. 8-32. Rhumbine II for the HP-41C.

```
129♦LBL 19
130 49
131 GTO "M"

132♦LBL 21
133 42
134 GTO "M"

135♦LBL 23
136 35
137 GTO "M"

138♦LBL 24
139 "NOT POSSIBLE"
140 AVIEW
141 STOP
142 GTO 26

143♦LBL 26
144 "END"
145 AVIEW
146 STOP

147♦LBL 27
148 RCL 06
149 67
150 *
151 STO 08
152 RCL 07
153 0.38
154 *
155 67
156 -
157 ABS
158 STO 10
159♦LBL 25
160 RCL 10
161 STO 11
162 RCL 08
163 ENTER↑
164 RCL 11
165 R-P
166 STO 12
167 X<>Y
168 STO 15
169 FIX 0
170 "MILES="
171 ARCL 12
172 AVIEW
173 STOP
174 RCL 02
175 ENTER↑
176 RCL 04
177 X=Y?
178 GTO A
179 GTO B

180♦LBL A
181 SF 00
182 RCL 02
183 ENTER↑
184 RCL 04
185 X>Y?
186 GTO C
187 GTO D

188♦LBL C
189 270
190 STO 13
191 FS? 00
192 GTO F
193 GTO E

194♦LBL D
195 90
196 STO 14
197 FS? 00
198 GTO F
199 GTO G

200♦LBL B
201 RCL 02
202 ENTER↑
203 RCL 04
204 X>Y?
205 GTO C
206 GTO D

207♦LBL E
208 RCL 03
209 ENTER↑
210 RCL 01
211 X>Y?
212 GTO H
213 GTO I
214♦LBL G
215 RCL 03
216 ENTER↑
217 RCL 01
218 X>Y?
219 GTO J
220 GTO "K"

221♦LBL I
222 RCL 13
223 RCL 15
224 +
225 STO 16
226 GTO F
227♦LBL H
228 RCL 15
229 RCL 13
230 -
231 STO 16
232 GTO F
233♦LBL J
234 RCL 14
235 RCL 15
236 +
237 STO 16
238 GTO F
239♦LBL "K"
240 RCL 15
241 RCL 14
242 -
243 STO 16
244 GTO F
245♦LBL F
246 RCL 16
247 ABS
248 STO 16
249 "DIR.DEGR.="
250 ARCL 16
251 AVIEW
252 STOP
253 "AGAIN? 0=YES"
254 PROMPT
255 X=0?
256 GTO 00
257 "END"
258 AVIEW
259 STOP
260 END
```

Chapter 9
Programs Dealing with Units

The programs in this chapter show how to make conversions or find unit measures. Study them carefully and you will be able to use the general theory in many programs.

UNIT CONVERSIONS

Often we find we can use a program to provide answers to many different questions, questions which are related and which require an identical or similar mathematical treatment for their solution.

A good example are unit conversions. With most of the rest of the world on the metric system and the U.S. in the process of converting to that system, we are frequently confronted with weights and measures which must be converted in one direction or the other. This program is an example of how such a multipurpose program can be constructed.

All conversions can be accomplished by multiplying one unit with a figure, available in dictionaries, encyclopedias, and other research literature.

For our example, I have devised two nearly identical programs, one dealing with linear and area measurements and the other with volume and weight measurements. Since some of the factors by which units must be multiplied consist of a considerable number of digits, and since the TI-59 considers each individual digit and decimal point a program step, I have placed all the multiplication factors into data memories; from there they can then be recalled as needed.

TI-59. Figures 9-1 and 9-2 show the program, easier to understand if we look at the program while reading the description.

Each user-defined key (**A** through **E** and **A'** through **E'**) represents one of 10 unit conversions contained in this program. **A** converts inches to

centimeters; **B** converts feet to meters; **C** converts statute miles to kilometers; **D** converts 1/64 of an inch into centimeters (the reason for this is that it is often quite difficult to convert fractions of inches into metric units; this step can be a great help): **E** converts yards to meters; **A'** converts square inches to square centimeters; **B'** converts square feet to square meters; **C'** converts square statute miles to square kilometers; **D'** converts 1/64 square inches to square centimeters; and **E'** converts square yards to square meters.

In addition, each of the 10 programs also performs the reciprocal conversion, from the metric unit to the standard unit. With reference to the

UNIT CONVERSION PROGRAM I (linear and area measures)

Data stored in registers:

02 - 2.54	06 - 0.0929
03 - 0.3048	07 - 2.59
04 - 1.6093	08 - 0.9144
05 - 6.4516	09 - 0.83613

The program steps:

000 Lbl	026 R/S	052 06	078 06	104 E'
001 A	027 00	053 04	079 GTO	105 STO
002 STO	028 R/S	054 =	080 EE	106 00
003 00	029 Lbl	055 GTO	081 Lbl	107 RCL
004 RCL	030 B	056 EE	082 C'	108 09
005 02	031 STO	057 Lbl	083 STO	109 GTO
006 Lbl	032 00	058 E	084 00	110 EE
007 EE	033 RCL	059 STO	085 RCL	0
008 STO	034 03	060 00	086 07	0
009 01	035 GTO	061 RCL	087 GTO	0
010 x	036 EE	062 08	088 EE	
011 RCL	037 Lbl	063 GTO	089 Lbl	
012 00	038 C	064 EE	090 D'	
013 =	039 STO	065 Lbl	091 STO	
014 R/S	040 00	066 A'	092 00	
015 x	041 RCL	067 STO	093 RCL	
016 RCL	042 04	068 00	094 05	
017 01	043 GTO	069 RCL	095 /	
018 1/x	044 EE	070 05	096 4	
019 =	045 Lbl	071 GTO	097 0	
020 R/S	046 D	072 EE	098 9	
021 x	047 STO	073 Lbl	099 6	
022 RCL	048 00	074 B'	100 =	
023 01	049 RCL	075 STO	101 GTO	
024 1/x	050 02	076 00	102 EE	
025 =	051 /	077 RCL	103 Lbl	

Fig. 9-1. The first Unit Conversion program for the TI-59.

storage registers, in addition to those used to store the multiplication factors, we'll be using 00 and 01 to execute each program.

Start writing the program by labeling the first section, pressing **LRN** to place the computer into program mode, and when the display shows 000 00, press **2nd Lbl A,** followed by **STO 00,** which will later store the variable value we will enter into register 00. Then press **RCL 02** to recall the multiplication factor which we previously stored in register 02. Now press **Lbl EE.** From here on all the programs follow the same routine, and there is no need to repeat each program step for each program. Instead, each of the other nine programs will contain a command to go to label **EE** (that key was arbitrarily chosen—it could have been any other) by the command **GTO EE.** Back to the first program, press **STO 01,** which will store the value recalled from register 02 in register 01. Now press **x RCL 00 =,** telling the computer to multiply what we placed into register 01 by the contents of register 00 and to display the result. Pressing **R/S** at this point stops the program execution and displays the result.

To perform the opposite calculation, to convert centimeters to inches, first recall our initial variable into display. This time, instead of representing inches, it will simply represent centimeters. To accomplish this, press **x RCL 01 1/x = R/S.** This multiplies the previous result by the reciprocal of the multiplication factor in register 01, and the display will be the variable we entered originally. Press **x RCL 01 1/x = R/S** once more and the value displayed will be the number of inches represented by the previously displayed number of centimeters.

To avoid accidentally pressing **R/S** too often, which could result in some confusing and meaningless displays, press **0** to put a zero into the display, and follow it by pressing R/S once again. If the operator should press **R/S** after the program is completed, the display will show *0,* reminding him he has reached the end of this program section.

The next program portion will convert feet to meters. Start by pressing **2nd Lbl B** followed by **STO 00 RCL 03 GTO EE.** That's all there is; all the other sections are identical except for the number of the **RCL** register which contains the multiplication factor and changes for each program section.

The only deviation from this format is found in the sections dealing with fractions of an inch. In the section where 1/64 of an inch is converted to centimeters, the program steps include dividing the multiplication factor by 64; in the section in which 1/64 square inch is converted to square centimeters, the multiplication factor is divided by 4096, the number of square 1/64 inches in one square inch.

To elaborate on the reason for including the fractions, we often find we are confronted with the problem of performing some type of mathematical function, possibly as simple as addition or subtraction, involving, fractions of an inch, for instance and 3 7/16 of an inch. If we want to convert that to a decimal value, the program makes it simple: First press **7 + 3 = STO 21** (be sure to use a register that is not used by the program!). Then press **40**

Running the Unit Conversion Program I:
(using arbitrary figures)

Press:	Display:	
CLR	0	
5	5	(inches)
A	12.70	(centimeters)
R/S	5.00	(centimeters)
R/S	1.97	(inches)
CLR	0	
5	5	(feet)
B	1.52	(meters)
R/S	5.00	(meters)
R/S	16.40	(feet)
CLR	0	
5	5	(statute miles)
C	8.05	(kilometers)
R/S	5.00	(kilometers)
R/S	3.11	(statute miles)
CLR	0	
5	5	(5/64 inches)
D	0.20	(centimeter)
R/S	5.00	(centimeter)
R/S	125.98	(125.98/64 inches)
CLR	0	
5	5	(yards)
E	4.57	(meters)
R/S	5.00	(meters)
R/S	5.47	(yards)
CLR	0	
5	5	(square inches)
2nd A'	32.26	(square centimeters)
R/S	5.00	(square centimeters)
R/S	0.78	(square inches)
CLR	0	
5	5	(square feet)
2nd B'	0.46	(square meters)
R/S	5.00	(square meters)
R/S	53.82	(square feet)
CLR	0	
5	5	(square statute miles)
2nd C'	12.95	(square kilometers)
R/S	5	(square kilometers)
R/S	1.9305	(square statute miles)
CLR	0	
5	5	(5/64 square inches)
2nd D'	0.01	(square centimeters)(.0078754883)
R/S	5.00	(square centimeters)
R/S	3174.41	(3174.41/64 square inches)
CLR	0	
5	5	(square yards)
2nd E'	4.18	(square meters)
R/S	5.00	(square meters)
R/S	5.98	(square yards)
R/S	0	

Fig. 9-2. The keystrokes and resulting displays when running the Unit Conversion I program in the TI-59.

UNIT CONVERSION PROGRAM II
(volume and weight measures)

Data stored in registers:

02 - 16,387	06 - 0.9072022789
03 - 0.02831	07 - 29.57
04 - 0.0648	08 - 0.94607
05 - 28.35	09 - 0.473035
11 - 0.4536011394	10 - 3.7999696

The program steps:

000 Lbl	025 01	050 00	075 RCL	100 10
001 A	026 1/x	051 RCL	076 07	101 GTO
002 Fix	027 =	052 05	077 GTO	102 EE
003 04	028 R/S	053 GTO	078 EE	0
004 STO	029 0	054 EE	079 Lbl	0
005 00	030 R/S	055 Lbl	080 B′	0
006 RCL	031 Lbl	056 E	081 STO	
007 02	032 B	057 STO	082 00	
008 Lbl	033 STO	058 00	083 RCL	
009 EE	034 00	059 RCL	084 08	
010 STO	035 RCL	060 06	085 GTO	
011 01	036 03	061 GTO	086 EE	
012 x	037 GTO	062 EE	087 Lbl	
013 RCL	038 EE	063 Lbl	088 C′	
014 00	039 Lbl	064 E′	089 STO	
015 =	040 C	065 STO	090 00	
016 R/S	041 STO	066 00	091 RCL	
017 x	042 00	067 RCL	092 09	
018 RCL	043 RCL	068 11	093 GTO	
019 01	044 04	069 GTO	094 EE	
020 1/x	045 GTO	070 EE	095 Lbl	
021 =	046 EE	071 Lbl	096 D′	
022 R/S	047 Lbl	072 A′	097 STO	
023 x	048 D	073 STO	098 00	
024 RCL	049 STO	074 00	099 RCL	

Fig. 9-3. The Unit Conversion II program accommodating 10 sets of conversions, for the TI-59.

(5/8 equals 40/64) and **D** and the display shows that 40/64 of an inch represent *1.5875* centimeters. Press **STO 22** to save this value. Now press **CLR** and **28** (7/16 equal 28/64) followed by **D** and the display shows *1.1113*. Press **STO 23,** then press **RCL 21 A** and the display shows *25.4000*. Now press **+ RCL 22 + RCL 23 =** and the display responds with *28.0988*. To determine what that is in inches and decimal fractions of inches, press **A, R/S, R/S,** and the display gives the result of *11.0625*.

Running the Unit Conversion Program II
(using arbitrary figures)

Press:	Display:	
CLR	0	
5	5	(cubic inches)
A	81935.0000	(cubic centimeters)
R/S	5.0000	(cubic centimeters)
R/S	0.00031	(cubic inches)
R/S	0	
5	5	(cubic feet)
B	0.1416	(cubic meters)
R/S	5.0000	(cubic meters)
R/S	176.6160	(cubic feet)
R/S	0	
5	5	(grains)
C	0.3240	(grams)
R/S	5.0000	(grams)
R/S	77.1605	(grains)
R/S	0	
5	5	(ounces)
D	141.7500	(grams)
R/S	5.0000	(grams)
R/S	0.1764	(ounces)
R/S	0	
5	5	(U.S. pounds)
E	4.5360	(metric pounds)
R/S	5.0000	(metric pounds)
R/S	5.5115	(U.S. pounds)
R/S	0	
5	5	(U.S. pounds)
2nd E'	2.2680	(kilograms)
R/S	5.0000	(kilograms)
R/S	11.0229	(U.S. pounds)
R/S	0	
5	5	(fluid ounces)
2nd A'	147.8500	(cubic centimeters)
R/S	5.0000	(cubic centimeters)
R/S	0.1691	(fluid ounces)
R/S	0	
5	5	(quarts)
2nd B'	4.7304	(liters)
R/S	5.0000	(liters)
R/S	5.2850	(quarts)
R/S	0	
5	5	(pints)
2nd C'	2.3652	(liters)
R/S	5.0000	(liters)
R/S	10.5700	(pints)
R/S	0	
5	5	(gallons)
2nd D'	18.9998	(liters)
R/S	5.0000	(liters)
R/S	1.3158	(gallons)
R/S	0	

Fig. 9-4. The keystrokes and displays when running the Unit Conversion II program in the TI-59.

000	76	LBL	055	04	4	110	09	09
001	11	A	056	95	=	111	61	GTO
002	58	FIX	057	61	GTO	112	52	EE
003	04	04	058	52	EE	113	00	0
004	42	STO	059	76	LBL			
005	00	00	060	15	E			
006	43	RCL	061	42	STO			
007	02	02	062	00	00	2.54		02
008	76	LBL	063	43	RCL	0.3048		03
009	52	EE	064	08	08	1.6093		04
010	42	STO	065	61	GTO	6.4516		05
011	01	01	066	52	EE	0.0929		06
012	65	×	067	76	LBL	2.59		07
013	43	RCL	068	16	A'	0.9144		08
014	00	00	069	42	STO	0.83613		09
015	95	=	070	00	00			
016	91	R/S	071	43	RCL			
017	65	×	072	05	05			
018	43	RCL	073	61	GTO			
019	01	01	074	52	EE			
020	35	1/X	075	76	LBL			
021	95	=	076	17	B'			
022	91	R/S	077	42	STO			
023	65	×	078	00	00			
024	43	RCL	079	43	RCL			
025	01	01	080	06	06			
026	35	1/X	081	61	GTO			
027	95	=	082	52	EE			
028	91	R/S	083	76	LBL			
029	00	0	084	18	C'			
030	91	R/S	085	42	STO			
031	76	LBL	086	00	00			
032	12	B	087	43	RCL			
033	42	STO	088	07	07			
034	00	00	089	61	GTO			
035	43	RCL	090	52	EE			
036	03	03	091	76	LBL			
037	61	GTO	092	19	D'			
038	52	EE	093	42	STO			
039	76	LBL	094	00	00			
040	13	C	095	43	RCL			
041	42	STO	096	05	05			
042	00	00	097	55	÷			
043	43	RCL	098	04	4			
044	04	04	099	00	0			
045	61	GTO	100	09	9			
046	52	EE	101	06	6			
047	76	LBL	102	95	=			
048	14	D	103	61	GTO			
049	42	STO	104	52	EE			
050	00	00	105	76	LBL			
051	43	RCL	106	10	E'			
052	02	02	107	42	STO			
053	55	÷	108	00	00			
054	06	6	109	43	RCL			

Fig. 9-5. Printout of the Unit Conversion I program for the TI-59.

The program, shown in Figs. 9-3 and 9-4, is in fact, identical to the one we just discussed, except that it deals with volume measurements and weights.

The programs in Figs. 9-5 and 9-6 can be used for any conversion which deals with a single multiplication factor. To accommodate other conversions, simply replace the values stored in the registers with those applicable to the problem at hand, and bingo—you're ready to go.

```
000  76 LBL     042  42 STO     084  43 RCL
001  11  A      043  00  00     085  08  08
002  58 FIX     044  43 RCL     086  61 GTO
003  04  04     045  04  04     087  52 EE
004  42 STO     046  61 GTO     088  76 LBL
005  00  00     047  52 EE      089  18 C'
006  43 RCL     048  76 LBL     090  42 STO
007  02  02     049  14  D      091  00  00
008  76 LBL     050  42 STO     092  43 RCL
009  52 EE      051  00  00     093  09  09
010  42 STO     052  43 RCL     094  61 GTO
011  01  01     053  05  05     095  52 EE
012  65  ×      054  61 GTO     096  76 LBL
013  43 RCL     055  52 EE      097  19 D'
014  00  00     056  76 LBL     098  42 STO
015  95  =      057  15  E      099  00  00
016  91 R/S     058  42 STO     100  43 RCL
017  65  ×      059  00  00     101  10  10
018  43 RCL     060  43 RCL     102  61 GTO
019  01  01     061  06  06     103  52 EE
020  35 1/X     062  61 GTO     104  00  0
021  95  =      063  52 EE
022  91 R/S     064  76 LBL          16387.  02
023  65  ×      065  10 E'       0.02831  03
024  43 RCL     066  42 STO        0.0648  04
025  01  01     067  00  00         28.35  05
026  35 1/X     068  43 RCL   .90720222789  06
027  95  =      069  06  06         29.57  07
028  91 R/S     070  61 GTO       0.94607  08
029  00  0      071  52 EE     0.4730375  09
030  91 R/S     072  76 LBL      3.7999696  10
031  76 LBL     073  16 A'    .4536011394  11
032  12  B      074  42 STO
033  42 STO     075  00  00
034  00  00     076  43 RCL
035  43 RCL     077  07  07
036  03  03     078  61 GTO
037  61 GTO     079  52 EE
038  52 EE      080  76 LBL
039  52 EE      081  17 B'
040  76 LBL     082  42 STO
041  13  C      083  00  00
```

Fig. 9-6. Printout of the Unit Conversion II program for the TI-59.

UNIT CONVERSION PROGRAM (for HP-41)

Data stored in registers:

02 - 2.540000	03 - 0.304800
04 - 1.609300	05 - 6.451600
07 - 2.590000	08 - 0.914400
09 - 0.836130	10 - 16.387.0000
11 - 0.028310	12 - 0.064800
13 - 0.304800	14 - 0.907202
15 - 29.57000	16 - 0.946070
17 - 0.473038	18 - 3.799970
19 - 0.453601	

The program steps:

01 LBL X	21 LBL AA	41 PROMPT
02 FIX 4	22 INCH/CM	42 STO 00
03 LBL Z	23 PROMPT	43 RCL 03
04 STO 01	24 STO 00	44 XEQ Z
05 RCL 00	25 RCL 02	45 METER=
06 x	26 XEQ Z	46 ARCL 20
07 STO 20	27 CM=	47 AVIEW
08 RCL 01	28 ARCL 00	48 STOP
09 1/x	29 AVIEW	49 METER=
10 x	30 STOP	50 ARCL 21
11 STO 21	31 CM=	51 AVIEW
12 RCL 01	32 ARCL 21	52 STOP
13 1/x	33 AVIEW	53 FEET=
14 x	34 STOP	54 ARCL 22
15 STO 22	35 INCH=	55 AVIEW
16 RTN	36 ARCL 22	56 STOP
17 LBL UC	37 AVIEW	57 LBL CC
18 CONVERT UNIT	38 STOP	58 MILES/KM
19 AVIEW	39 LBL BB	59 PROMPT
20 PSE	40 FEET/METER	60 STO 00
61 RCL 04	96 YARD/METER	131 LBL bb
62 XEQ Z	97 PROMPT	132 SQ.FT/SQ.M
63 KM=	98 STO 00	133 PROMPT
64 ARCL 20	99 RCL 08	134 STO 00
65 AVIEW	100 XEQ Z	135 RCL 06
66 STOP	101 M=	136 XEQ Z
67 KM=	102 ARCL 20	137 SQ.M=
68 ARCL 21	103 AVIEW	138 ARCL 20
69 AVIEW	104 STOP	139 AVIEW
70 STOP	105 M=	140 STOP
71 MILES=	106 ARCL 21	141 SQ.M=
72 ARCL 22	107 AVIEW	142 ARCL 21
73 AVIEW	108 STOP	143 AVIEW
74 STOP	109 YARDS=	144 STOP
75 LBL DD	110 ARCL 22	145 SQ.FT=
76 1/64 IN/CM	111 AVIEW	146 ARCL 22
77 PROMPT	112 STOP	147 AVIEW
78 STO 00	113 LBL aa	148 STOP
79 RCL 02	114 SQ.IN/SQ.CM	149 LBL cc
80 64	115 PROMPT	150 SQ.MI/SQ.KM
81 /	116 STO 00	151 PROMPT

Fig. 9-7. All 20 programs can be written as one Unit Conversion program for the HP-41C.

The program steps:

82 XEQ Z	117 RCL 05	152 STO 00
83 CM=	118 XEQ Z	153 RCL 07
84 ARCL 20	119 SQ.CM=	154 XEQ Z
85 AVIEW	120 ARCL 20	155 SQ.KM=
86 STOP	121 AVIEW	156 ARCL 20
87 CM=	122 STOP	157 AVIEW
88 ARCL 21	123 SQ.CM=	158 STOP
89 AVIEW	124 ARCL 21	159 SQ.KM=
90 STOP	125 AVIEW	160 ARCL 21
91 /64 IN=	126 STOP	161 AVIEW
92 ARCL 22	127 SQ.IN=	162 STOP
93 AVIEW	128 ARCL 22	163 SQ.MI=
94 STOP	129 AVIEW	164 ARCL 22
95 LBL EE	130 STOP	165 AVIEW
166 STOP	201 SQ.YD=	236 STOP
167 LBL dd	202 ARCL 22	237 CU.FT=
168 1/64 S.IN/S.CM	203 AVIEW	238 ARCL 22
169 PROMPT	204 STOP	239 AVIEW
170 STO 00	205 LBL FF	240 STOP
171 RCL 05	206 CU.IN/CU.CM	241 LBL HH
172 4096	207 PROMPT	242 GRAIN/GRAM
173 /	208 STO 00	243 PROMPT
174 XEQ Z	209 RCL 10	244 STO 00
175 SQ.CM=	210 XEQ Z	245 RCL 12
175 ARCL 20	211 CU.CM=	246 XEQ Z
177 AVIEW	212 ARCL 20	247 GRAM=
178 STOP	213 AVIEW	248 ARCL 20
179 SQ.CM=	214 STOP	249 AVIEW
180 ARCL 21	215 CU.CM=	250 STOP
181 AVIEW	216 ARCL 21	251 GRAM=
182 STOP	217 AVIEW	252 ARCL 21
183 /64 S.IN=	218 STOP	253 AVIEW
184 ARCL 22	219 CU.IN=	254 STOP
185 AVIEW	220 ARCL 22	255 GRAIN=
186 STOP	221 AVIEW	256 ARCL 22
187 LBL ee	222 STOP	257 AVIEW
188 SQ.YRD/SQ.M	223 LBL GG	258 STOP
189 PROMPT	224 CU.FT/CU.M	259 LBL II
190 STO 00	225 PROMPT	260 OZ/GRAM
191 RCL 09	226 STO 00	261 PROMPT
192 XEQ Z	227 RCL 11	262 STO 00
193 SQ.M=	228 XEQ Z	263 RCL 13
194 ARCL 20	229 CU.M=	264 XEQ Z
195 AVIEW	230 ARCL 20	265 GRAM=
196 STOP	231 AVIEW	266 ARCL 20
197 SQ.M=	232 STOP	267 AVIEW
198 ARCL 21	233 CU.M=	268 STOP
199 AVIEW	234 ARCL 21	269 GRAM=
200 STOP	235 AVIEW	270 ARCL 21
271 AVIEW	306 ARCL 21	341 LIT=
272 STOP	307 AVIEW	342 ARCL 21
273 OZ=	308 STOP	343 AVIEW
274 ARCL 22	309 LB=	344 STOP
275 AVIEW	310 ARCL 22	345 QURT=

The program steps:

276 STOP	311 AVIEW	346 ARCL 22
277 LBL JJ	312 STOP	347 AVIEW
278 LB/KG	313 LBL %	348 STOP
279 PROMPT	314 FL.OZ/CU.CM	349 LBL <
280 STO 00	315 PROMPT	350 PINTS/LITER
281 RCL 19	316 STO 00	351 PROMPT
282 XEQ Z	317 RCL 15	352 STO 00
283 KG=	318 XEQ Z	353 RCL 17
284 ARCL 20	319 CU.CM=	354 XEQ Z
285 AVIEW	320 ARCL 20	355 LIT=
286 STOP	321 AVIEW	356 ARCL 20
287 KG=	322 STOP	357 AVIEW
288 ARCL 21	323 CU.CM=	358 STOP
289 AVIEW	324 ARCL 21	359 LIT=
290 STOP	325 AVIEW	360 ARCL 21
291 LB=	326 STOP	361 AVIEW
292 ARCL 22	327 FL.OZ=	362 STOP
293 AVIEW	328 ARCL 22	363 PNT=
294 STOP	329 AVIEW	364 ARCL 22
295 LBL Σ	330 STOP	365 AVIEW
296 LB/POUND	331 LBL <	366 STOP
297 PROMPT	332 QUART/LITER	367 LBL >
298 STO 00	333 PROMPT	368 GAL/LITER
299 RCL 14	334 STO 00	369 PROMPT
300 XEQ Z	335 RCL 16	370 STO 00
301 PND=	336 XEQ Z	371 RCL 18
302 ARCL 20	337 LIT=	372 XEQ Z
303 AVIEW	338 ARCL 20	373 LIT=
304 STOP	339 AVIEW	374 ARCL 20
305 PND=	340 STOP	375 AVIEW
376 STOP		
377 LIT=		
378 ARCL 21		
379 AVIEW		
380 STOP		
381 GAL=		
382 ARCL 22		
383 AVIEW		
384 STOP		
385 END		
386 AVIEW		
387 STOP		
388 END		

Key Assignments

For the purpose of clarity the keys are identified here by their alpha designations, even though the computer must NOT be in ALPHA mode when these user-defined keys are used.

Program from the top	P
inch/centimeter	A
feet/meter	B
miles/kilometers	C
1/64 inch/centimeters	D
yard/meter	E
square inch/sq. centimeters	shift A
square feet/sq. meter	shift B
square mile/sq. kilometer	shift C
1/64 sq. inch/sq. centimeter	shift D
square yard/sq. meter	shift E
cubic inch/cu. centimeter	F
cubic foot/cu. meter	G
grain/gram	H
ounce/gram	I
U.S. pound/kilogram	J
U.S. pound/metric pound	shift F
fluid ounce/cu. centimeter	shift G
quart/liter	shift H
pint/liter	shift I
gallon/liter	shift J

Fig. 9-7. Continued from page 133.

Running the Unit Conversion Program in the HP-41
(the input figures used are arbitrary)

Press:	Display:	
ON	3.927962	something left over from before
USER	3.927962 (USER)	
P (EEX)	CONVERT UNIT	identifies the program, then
	INCH/CM	this legend appears; sub-program
8	8	inches
R/S	CM= 20.3200	centimeters
R/S	CM= 8.0000	centimeters
R/S	INCH= 3.1496	inches
R/S	FEET/METER	sub-program
8	8	feet
R/S	METER=2.4384	meters
R/S	METER=8.0000	meters
R/S	FEET= 26.2467	feet
R/S	MILES/KM	sub-program
8	8	miles
R/S	KM=12.8744	kilometers
R/S	KM=8.0000	kilometers
R/S	MILES= 4.9711	miles
R/S	1/64 IN/CM	sub-program
8	8	8/64 inch
R/S	CM= 0.3175	centimeters
R/S	CM= 8.0000	centimeters
R/S	/64 IN=201.5748	201.5748/64" inches
64	64	
/	3.1496	inches
R/S	YARD/METER	sub-program
8	8	yards
R/S	M= 7.3152	meters
R/S	M= 8.0000	meters
R/S	YARDS= 8.7489	yards
R/S	SQ.IN/SQ.CM	sub-program (sq.inch/sq. centimeter)
8	8	square inches
R/S	SQ.CM= 51.6128	square centimeters
R/S	SQ.CM= 8.0000	square centimeters
R/S	SQ.IN= 1.2400	square inches
R/S	SQ.FT/SQ.M	sub-program (sq.foot/sq.meter)
8	8	square feet
R/S	SQ.M= 0.7432	square meters
R/S	SQ.M= 8.0000	square meters
R/S	SQ.FT= 86.1141	square feet
R/S	SQ.MI/SQ.KM	sub-program (sq.mile/sq.kilometer)
8	8	square miles
R/S	SQ.KM= 20.7200	square kilometers
R/S	SQ.KM= 8.0000	square kilometers
R/S	SQ.MI= 3.0888	square miles
R/S	1/64 S.IN/S.CM	sub-program (1/64 sq.inch/sq. centimeter.)
8	8	8/64 square inch
R/S	SQ.CM= 0.0126	square centimeters
R/S	SQ.CM= 8.0000	square centimeters
R/S	/64 S.IN=5,079.0502	
64	64	
x^2	4096	
/	1.2400	square inches
R/S	SQ.YRD/SQ.M	sub-program (sq.yard/sq.meter)

Fig. 9-8. The keystrokes and displays when running the 20-section Unit Conversion program in the HP-41C.

Press:	Display:	
8	8	square yards
R/S	SQ.M= 6.6890	square meters
R/S	SQ.M= 8.0000	square meters
R/S	SQ.YD=9.5679	square yards
R/S	CU.IN/CU.CM	sub-program (cubic inch/cu.centimeter)
8	8	cubic inches
R/S	CU.CM=131,096.0000	cubic centimeters
R/S	CU.CM= 8.0000	cubic centimeters
R/S	CU.IN= 0.0005	cubic inches
R/S	CU.FT/CU.M	sub-program (cubic feet/cu. meters)
8	8	cubic feet
R/S	CU.M = 0.2265	cubic meters
R/S	CU.M= 8.0000	cubic meters
R/S	CU.FT=282.5857	cubic feet
R/S	GRAIN/GRAM	sub-program
8	8	grains
R/S	GRAM= 0.5184	grams
R/S	GRAM= 8.0000	grams
R/S	GRAIN=123.4568	grains
R/S	OZ/GRAMS	sub-program (ounces/grams)
8	8	ounces
R/S	GRAM=226.8000	grams
R/S	GRAM= 8.0000	grams
R/S	OZ= 0.2822	ounces
R/S	LB/KG	sub-program (U.S.pounds/kilogram)
8	8	U.S. pounds
R/S	KG= 3.6288	kilograms
R/S	KG= 8.0000	kilograms
R/S	LB= 17.6366	U.S. pounds
R/S	LB/POUND	sub-program (U.S.pound/metric pound)
8	8	U.S. pounds
R/S	PND= 7.2576	metric pounds
R/S	PND= 8.0000	metric pounds
R/S	LB= 8.8183	U.S. pounds
R/S	FL.OZ/CU.CM	sub-program (fluid ounces/cu.centim.)
8	8	fluid ounces
R/S	CU.CM=236.5600	cubic centimeters
R/S	CU.CM= 8.0000	cubic centimeters
R/S	FL.OZ= 0.2705	fluid ounces
R/S	QUART/LITER	sub-program
8	8	quarts
R/S	LIT= 7.5686	liters
R/S	LIT= 8.0000	liters
R/S	QUART= 8.4560	quarts
R/S	PINTS/LITER	sub-program
8	8	pints
R/S	LIT= 3.7843	liters
R/S	LIT= 8.0000	liters
R/S	PNT= 16.9120	pints
R/S	GAL/LITER	sub-program (gallons/liters)
8	8	U.S. gallons
R/S	LIT= 30.3998	liters
R/S	LIT= 8.0000	liters
R/S	GAL= 2.1053	U.S. gallons
R/S	END	

(Note: Any sub program can be called up at the press of a single key. See key assignment table in Fig.9-7)

Fig. 9-8. Continued from page 135.

```
 01♦LBL "X"
FS? 00  GTO "UC"
000.025  XROM 30,03
FIX 4  SF 00

          03♦LBL "Z"
          04 STO 01
          05 RCL 00
          06 *
          07 STO 20
          08 RCL 01
          09 1/X
          10 *
          11 STO 21
          12 RCL 01
          13 1/X
          14 *
          15 STO 22
          16 RTN

          17♦LBL "UC"
       18 "CONVERT UNIT"
          19 AVIEW
          20 PSE

          21♦LBL "AA"
          22 "INCH/CM"
          23 PROMPT
          24 STO 00
          25 RCL 02
          26 XEQ "Z"
          27 "CM= "
          28 ARCL 20
          29 AVIEW
          30 STOP
          31 "CM= "
          32 ARCL 21
          33 AVIEW
          34 STOP
          35 "INCH= "
          36 ARCL 22
          37 AVIEW
          38 STOP

          39♦LBL "BB"
       40 "FEET/METER"
          41 PROMPT
          42 STO 00
          43 RCL 03
          44 XEQ "Z"
          45 "METER= "
          46 ARCL 20
          47 AVIEW
          48 STOP
          49 "METER= "
          50 ARCL 21
          51 AVIEW
          52 STOP
             "EET= "
          ARCL 22
          55 AVIEW
          56 STOP

          57♦LBL "CC"
       58 "MILES/KM"
          59 PROMPT
          60 STO 00
          61 RCL 04
          62 XEQ "Z"
          63 "KM="
          64 ARCL 20
          65 AVIEW
          66 STOP
          67 "KM= "
          68 ARCL 21
          69 AVIEW
          70 STOP
          71 "MILES= "
          72 ARCL 22
          73 AVIEW
          74 STOP

          75♦LBL "DD"
       76 "1/64 IN/CM"
          77 PROMPT
          78 STO 00
          79 RCL 02
          80 64
          81 /
          82 XEQ "Z"
          83 "CM= "
          84 ARCL 20
          85 AVIEW
          86 STOP
          87 "CM= "
          88 ARCL 21
          89 AVIEW
          90 STOP
          91 "/64 IN= "
          92 ARCL 22
          93 AVIEW
          94 STOP
          95♦LBL "EE"
       96 "YARD/METER"
          97 PROMPT
          98 STO 00
          99 RCL 08
         100 XEQ "Z"
         101 "M= "
         102 ARCL 20
         103 AVIEW
         104 STOP
         105 "M= "
         106 ARCL 21
         107 AVIEW
         108 STOP
         109 "YARDS= "
         110 ARCL 22
         111 AVIEW
         112 STOP

         113♦LBL "aa"
      114 "SQ.IN/SQ.CM"
         115 PROMPT
         116 STO 00
         117 RCL 05
         118 XEQ "Z"
         119 "SQ.CM= "
         120 ARCL 20
         121 AVIEW
         122 STOP
         123 "SQ.CM= "
         124 ARCL 21
         125 AVIEW
         126 STOP
         127 "SQ.IN= "
         128 ARCL 22
         129 AVIEW
         130 STOP

         131♦LBL "bb"
      132 "SQ.FT/SQ.M"
         133 PROMPT
         134 STO 00
         135 RCL 06
         136 XEQ "Z"
         137 "SQ.M= "
         138 ARCL 20
         139 AVIEW
         140 STOP
         141 "SQ.M= "
         142 ARCL 21
         143 AVIEW
```

Fig. 9-9. Printout of the HP-41C version of the Unit Conversion program.

```
144 STOP
145 "SQ.FT= "
146 ARCL 22
147 AVIEW
148 STOP

149♦LBL "cc"
150 "SQ.MI/SQ.KM"
151 PROMPT
152 STO 00
153 RCL 07
154 XEQ "Z"
155 "SQ.KM= "
156 ARCL 20
157 AVIEW
158 STOP
159 "SQ.KM= "
160 ARCL 21
161 AVIEW
162 STOP
163 "SQ.MI= "
164 ARCL 22
165 AVIEW
166 STOP

167♦LBL "dd"
168 "1/64 S.IN/S.CM"
169 PROMPT
170 STO 00
171 RCL 05
172 4096
173 /
174 XEQ "Z"
175 "SQ.CM="
176 ARCL 20
177 AVIEW
178 STOP
179 "SQ.CM= "
180 ARCL 21
181 AVIEW
182 STOP
183 "/64 S.IN="
184 ARCL 22
185 AVIEW
186 STOP

187♦LBL "ee"
188 "SQ.YRD/SQ.M"
189 PROMPT
190 STO 00
191 RCL 09
192 XEQ "Z"
193 "SQ.M= "
194 ARCL 20
195 AVIEW
196 STOP
197 "SQ.M= "
198 ARCL 21
199 AVIEW
200 STOP
201 "SQ.YD= "
202 ARCL 22
203 AVIEW
204 STOP

205♦LBL "FF"
206 "CU.IN/CU.CM"
207 PROMPT
208 STO 00
209 RCL 10
210 XEQ "Z"
211 "CU.CM= "
212 ARCL 20
213 AVIEW
214 STOP
215 "CU.CM= "
216 ARCL 21
217 AVIEW
218 STOP
219 "CU.IN= "
220 ARCL 22
221 AVIEW
222 STOP

223♦LBL "GG"
224 "CU.FT/CU.M"
225 PROMPT
226 STO 00
227 RCL 11
228 XEQ "Z"
229 "CU.M= "
230 ARCL 20
231 AVIEW
232 STOP
233 "CU.M= "
234 ARCL 21
235 AVIEW
236 STOP
237 "CU.FT= "
238 ARCL 22
239 AVIEW
240 STOP
241♦LBL "HH"
242 "GRAIN/GRAM"
243 PROMPT
244 STO 00
245 RCL 12
246 XEQ "Z"
247 "GRAM= "
248 ARCL 20
249 AVIEW
250 STOP
251 "GRAM= "
252 ARCL 21
253 AVIEW
254 STOP
255 "GRAIN="
256 ARCL 22
257 AVIEW
258 STOP

259♦LBL "II"
260 "OZ/GRAMS"
261 PROMPT
262 STO 00
263 RCL 13
264 XEQ "Z"
265 "GRAM= "
266 ARCL 20
267 AVIEW
268 STOP
269 "GRAM= "
270 ARCL 21
271 AVIEW
272 STOP
273 "OZ= "
274 ARCL 22
275 AVIEW
276 STOP

277♦LBL "JJ"
278 "LB/KG"
279 PROMPT
280 STO 00
281 RCL 19
282 XEQ "Z"
283 "KG= "
284 ARCL 20
285 AVIEW
286 STOP
287 "KG= "
288 ARCL 21
```

Fig. 9-9. Continued from page 137.

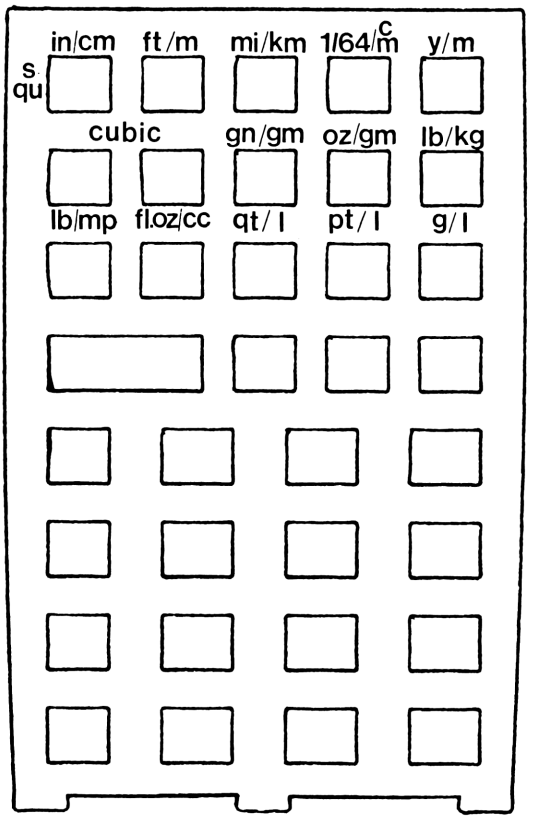
```
  289 AVIEW
  290 STOP
  291 "LB= "
  292 ARCL 22
  293 AVIEW
  294 STOP
  295♦LBL "Σ"
296 "LB/POUND"
  297 PROMPT
  298 STO 00
  299 RCL 14
  300 XEQ "Z"
  301 "PND= "
  302 ARCL 20
  303 AVIEW
  304 STOP
  305 "PND= "
  306 ARCL 21
  307 AVIEW
  308 STOP
  309 "LB= "
  310 ARCL 22
  311 AVIEW
  312 STOP
  313♦LBL "%"
314 "FL.OZ/CU.CM"
  315 PROMPT
  316 STO 00
  317 RCL 15
  318 XEQ "Z"
319 "CU.CM= "
  320 ARCL 20
  321 AVIEW
  322 STOP
323 "CU.CM= "
  324 ARCL 21
  325 AVIEW
  326 STOP
327 "FL.OZ= "
  328 ARCL 22
  329 AVIEW
  330 STOP
  331♦LBL "≠"
332 "QUART/LITER"
  333 PROMPT
  334 STO 00
  335 RCL 16
  336 XEQ "Z"
  337 "LIT.= "
  338 ARCL 20
  339 AVIEW
  340 STOP
  341 "LIT.= "
  342 ARCL 21
  343 AVIEW
  344 STOP
  345 "QURT= "
  346 ARCL 22
  347 AVIEW
  348 STOP

  349♦LBL "<"
350 "PINTS/LITER"
  351 PROMPT
  352 STO 00
  353 RCL 17
  354 XEQ "Z"
  355 "LIT.= "
  356 ARCL 20
  357 AVIEW
  358 STOP
  359 "LIT.= "
  360 ARCL 21
  361 AVIEW
  362 STOP
  363 "PNT.= "
  364 ARCL 22
  365 AVIEW
  366 STOP

  367♦LBL ">"
368 "GAL/LITER"
  369 PROMPT
  370 STO 00
  371 RCL 18
  372 XEQ "Z"
  373 "LIT.= "
  374 ARCL 20
  375 AVIEW
  376 STOP
  377 "LIT.= "
  378 ARCL 21
  379 AVIEW
  380 STOP
  381 "GAL.= "
  382 ARCL 22
  383 AVIEW
  384 STOP
  385 "END"
  386 AVIEW
  387 STOP
  388 .END.

R00=  5.00
R01=  3.80
R02=  2.54
R03=  0.30
R04=  1.61
R05=  6.45
R06=  0.09
R07=  2.59
R08=  0.91
R09=  0.84
R10=  16,387.00
R11=  0.03
R12=  0.06
R13=  28.35
R14=  0.91
R15=  29.57
R16=  0.95
R17=  0.47
R18=  3.80
R19=  0.45
R20=  19.00
R21=  5.00
R22=  1.32
R23=  0.00
R24=  0.00

USER KEYS:
 11 "AA"
-11 "aa"
 12 "BB"
-12 "bb"
 13 "CC"
-13 "cc"
 14 "DD"
-14 "dd"
 15 "EE"
-15 "ee"
 21 "FF"
-21 "Σ"
 22 "GG"
-22 "%"
 23 "HH"
-23 "≠"
 24 "II"
-24 "<"
 25 "JJ"
-25 ">"
 43 "UC"
```



1				4
UNIT CONVERSION I				
		square		
in : cm	ft : m	mi : km	1/64 : cm	y : m

1				4
UNIT CONVERSION II				mt
fl.oz : cc	qt : l	pt : l	gal : l	ton :
ci : cc	cf : cm	grn : gm	oz : gm	lb : kg

Fig. 9-10. With so many user-defined keys, a means of remembering which key does what is important.

HP-41C. This same set of problems can be solved with the HP-41. We not only have the added ability to use words to identify the variables which the computer needs to solve each problem and the meaning of each of the result, we also have a virtually unlimited number of keys to be used as user-defined keys. We can therefore expect to combine all 20 unit-conversion programs into one over-all program (Figs. 9-7 and 9-8).

The program for the HP consists of a brief first step which asks whether flag 00 is set. If it is not, it displays *CARD*, meaning that a card containing the data to be used in the conversions must be fed into the computer before the program can work. If the flag is set (a tiny *0* at the bottom of the display window would be in sight), the computer skips the request for the data card because the flag indicates that the card has already been read (Fig. 9-9).

The first portion of the program is identified as LBL Z and contains the actual calculation which will be used by every one of the 20 conversions which the program is designed to perform.

The section labeled LBL UC simply identifies the program by placing its name into the display.

From then on, there are 20 sections labeled LBL AA, BB, CC and so on, each with its own identifying display, and each designed to convert a different set of units. Each of these sections can be called up directly without wading through a lot of other sections, simply by using the appropriate user-defined key. The program uses the top 10 keys in their direct and shift modes. To remember which key does what, it would be advisable to prepare a keyboard overlay that identifies the user-defined keys (Fig. 9-10).

Program Steps

TI-59

Cards: 2 (1 per program)
Passes: 1 program and 1 data for each of two programs
Partition: 479 59
1. Enter amount of units to be converted
2. Press the key that controls the desired conversion.
Display: The result

Key assignments: Program I:
A: Inch-centimeter; vice versa
B: Feet-meter; vice versa
C: Miles-kilometers; vice versa
D: 1/64 inch-centimeter; vice versa
E: Yard-meter; vice versa
A′: Sq. inch-sq. cm; vice versa
B′: Sq.ft-sq.meter; vice versa
C′: Sq.mile-sq.kilometer; vice versa
D′: 1/64 sq.inch-sq.cm; vice versa
E′: Sq.yard-sq.meter; vice versa

Key assignment; program II:
A: Cu.inch-cu.cm; vice versa
B: Cu.ft.-cu.meter; vice versa
C. Grain-gram; vice versa
D. Ounce-gram; vice versa
E. US pound-metric pound; vice versa
A′: Fluid ounce-cu.cm; vice versa

B': Quart-liter; vice versa
C': Pint-liter; vice versa
D': Gal.-liter; vice versa
E': U.S. ton-metric ton; vice versa

Note: To obtain the vice versa function, press **R/S**

HP-41C

Cards: 6 program, 1 data
Passes: 11 program, 2 data
Size: 025

Display: *INCH/CM* (or any other designation; see below)

1. Enter number of inches to be converted; **R/S**
Display: *CM=25.4000* or whatever the result would be.
2. Press **R/S**
Display: *CM= 10.0000*
3. If you want a different input for the centimeter conversion, enter it; **R/S**

Display: *INCH= 3.9370* or whatever the result might be.

Key assignments:
1st row, key 1 through 5:
1: Inch/centimeter and vice versa
2: Feet/meter and vice versa
3: Miles/km and vice versa
4: 1/64 inch/cm and vice versa
5: Yard/meter and vice versa
1st row, *shift* key 1 through 5:
1: Sq.in/sq.cm and vice versa
2: Sq.ft/sq.meter and vice versa
3: Sq.mile/sq.km and vice versa
4: 1/64 sq.in/sq.cm and vice versa
5: Sq.yard/sq.meter and vice versa

2nd row, keys 1 through 5:
1: Cu.in/cu.cm and vice versa
2: Cu.ft/cu.meter and vice versa
3: Grain/gram and vice versa
4: Ounce/gram and vice versa
5: Pound/kilogram
2nd row, *shift* key 1 through 5:
1: U.S. pound/metric pound and vice versa
2: Fluid ounce/cu.cm and vice versa
3: Quart/liter and vice versa
4: Pint/liter and vice versa
5: Gallon/liter and vice versa

DIAGONAL OR POLAR CONVERSION

This program may not be of much practical use, but it makes use of the polar-to-rectangular and rectangular-to-polar conversion function, one of the built-in capabilities of both computers.

What it does is this: Given the lengths of two sides of a right angle, it will compute the location of the opposite corner, assuming that right angle were to be transformed into a rectangular box, and then figures out the angle of the diagonal to the side of the rectangle, and the length of the diagonal from corner to corner (see Fig. 9-11).

Conversely, given the angles and length of the diagonal, it will construct the rectangular box into which that diagonal would fit.

The ability to conduct these calculations proves to be very helpful once we get involved writing more complicated programs.

The two computers do not function in exactly the same manner.

To determine the length and angle of the diagonal with the TI, first key in side 2 (the x side), then press **x ≷ t** (Fig. 9-11). Key in the other distance (side 1 or y) and then press **INV 2nd P-R** and the computer will display the angle. Press **x≷t** again, producing the display of the length of the diagonal.

With this data about the diagonal, first key in the length of the diagonal, then press **x≷t** and after that key in the angle between the diagonal and the horizontal side. Press **2nd R-P** and the display will show first the length of side 1 (y) and then, after pressing **x≷t** again, it will show the length of side 2 (x).

With the HP, first enter side 1 (usually expressed as the y distance). Move that into the Y register by pressing **ENTER,↑** then key in side 2 (referred to as the x distance) and press **shift R-P.** The display will show the length of the diagonal. If we then press **x≷y,** the display will show the angle between the diagonal and side 2 (the x side) in degrees.

Conversely, if we know the length of the diagonal, key that in followed by **ENTER↑.** Then key in the angle between the diagonal and the horizontal side (x) and press **shift P-R.** The display will show the length of side 2 (x). Pressing **x≷y** will then change that to the length of side 1 (y).

Let's walk through these programs to become more familiar with them

TI-59: the **A** key is used to enter the lengths of sides 1 and 2, while pressing the **B** key performs the calculations. The **C** key is used to enter the data related to the diagonal, and the **D** key will produce the lengths of sides 1 and 2 (Fig. 9-12). Here is how it works:

Enter **20** and press **A.** Then enter **40** and press **R/S.** Then press **B** and the display first reads *63.43,* the angle of the diagonal. When you press **R/S** this time, *44.72* is in the display, the length of the diagonal.

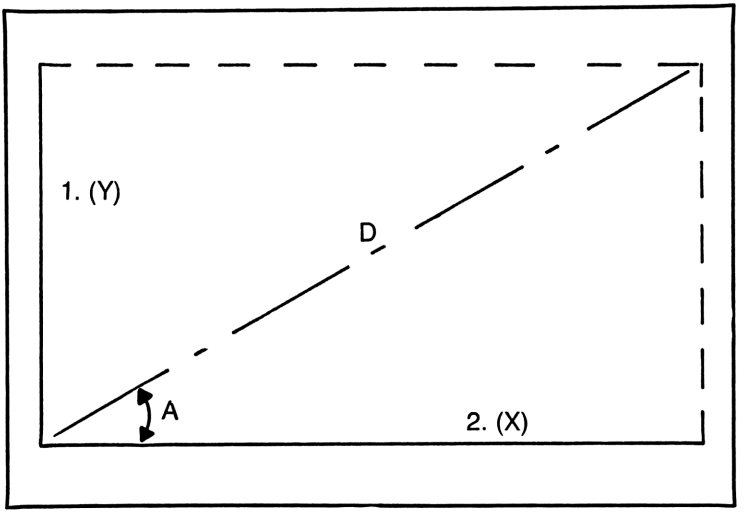


Fig. 9-11. The polar/rectangular conversion capability of the TI-59 and HP-41C.

```
000  76 LBL   015  22 INV   030  76 LBL
001  11  A    016  37 P/R   031  14  D
002  42 STO   017  58 FIX   032  43 RCL
003  00  00   018  02  02   033  02  02
004  91 R/S   019  91 R/S   034  32 X↕T
005  42 STO   020  32 X↕T   035  43 RCL
006  01  01   021  91 R/S   036  03  03
007  91 R/S   022  76 LBL   037  37 P/R
008  76 LBL   023  13  C    038  91 R/S
009  12  B    024  42 STO   039  32 X↕T
010  43 RCL   025  02  02   040  91 R/S
011  00  00   026  91 R/S   041  00  0
012  32 X↕T   027  42 STO
013  43 RCL   028  03  03
014  01  01   029  91 R/S
```

Fig. 9-12. How the TI-59 deals with polar/rectangular conversions.

If we enter that figure, **44.72,** and press **C** and then key in *63.43* and press **R/S**, pressing **D** will produce *40,* the length of side 2, and pressing **R/S** again will now produce *20,* the length of side 1.

HP: We'll use simple figures (Fig. 9-13). When the display asks for *SIDE 1?* enter **20.** Then when it asks for *SIDE 2?* enter **40.** Pressing **R/S**

```
 01↑LBL "S"          17 FIX 1          33 ENTER↑
02 "DIAGONAL"      18 "DISTANCE:"      34 RCL 05
 03 AVIEW            19 ARCL 02        35 P-R
 04 PSE              20 AVIEW          36 STO 06
05 "SIDE 1 ?"        21 STOP           37 X<>Y
 06 PROMPT           22 "ANGLE:"       38 STO 07
 07 STO 00           23 ARCL 03      39 "SIDE 1:"
08 "SIDE 2 ?"        24 AVIEW          40 ARCL 06
 09 PROMPT           25 STOP           41 AVIEW
 10 STO 01      26 "ANGLE DEGREE"      42 STOP
 11 ENTER↑           27 PROMPT       43 "SIDE 2:"
 12 RCL 00           28 STO 04         44 ARCL 07
 13 R-P            29 "DISTANCE ?"     45 AVIEW
 14 STO 02           30 PROMPT         46 STOP
 15 X<>Y             31 STO 05         47 END
 16 STO 03           32 RCL 04
```

Fig. 9-13. This program illustrates how the HP-41C handles the polar/rectangular conversion.

results in *DISTANCE:44.7* and pressing **R/S** again produces *ANGLE:63.4*.

To do the reverse, when the computer asks *ANGLE DEGREE* enter **63.4.** In response to *DISTANCE?* enter **44.7.** The displayed result will be *SIDE 1: 20.0* and *SIDE 2: 40.0.*

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 008
1). Enter length, side 1	Display: *SIDE 1?*
2). Press **A**	1). Enter length: **R/S**
3). Enter length, side 2	Display: *SIDE 2?*
4). Press **R/S**	2). Enter length: **R/S**
5). Press **B**	Display: *DISTANCE:* and result
Display: Angle of the diagonal	3). Press **R/S**
6). Press **R/S**	Display: *ANGLE*; and degrees or;
Display: Length of diagonal	Display: *ANGLE DEGREE?*
Or:	1). Enter degrees; **R/S**
1). Enter length of diagonal	Display: *DISTANCE?*
2). Press **C**	2). Enter length, diagonal; **R/S**
3). Enter angle, diagonal	Display: *SIDE 1:* and length
4). Press **R/S**	3). Press **R/S**
5). Press **D**	Display: *SIDE 2:* and length
Display: Length, side 2	
6). Press **R/S**	
Display: Length, side 1	

WORD COUNT

This is a program for writers, editors, and others who deal with the printed word. It figures out the number of printed pages that will result from typesetting a manuscript of a certain length. The input data consists of the average number of words found on a type-written page. This number varies depending on the type size used, the margins, etc. Also needed as input is information about the approximate number of words on the printed page, printed column or whatever form the final printed material will take.

Figures 9-14 and 9-15 show the programs for the two computers and Figs. 9-16 and 9-17 represent a trace of what happens when the program is run. For the purposes of this example I assumed that the average type-written page contains 250 words and that there will be 400 words on the printed page. I have also made allowances for possible illustrations, meaning that illustrated pages, the number to be entered, is not the number of pages on which illustrations will appear, but rather the approximate number

```
000  58 FIX   023  91 R/S   046  75  -
001  00  00   024  55  ÷    047  43 RCL
002  76 LBL   025  43 RCL   048  02  02
003  11  A    026  00  00   049  95  =
004  42 STO   027  95  =    050  94 +/-
005  00  00   028  42 STO   051  55  ÷
006  91 R/S   029  07  07   052  43 RCL
007  76 LBL   030  91 R/S   053  00  00
008  12  B    031  76 LBL   054  95  =
009  42 STO   032  14  D    055  42 STO
010  01  01   033  42 STO   056  04  04
011  91 R/S   034  03  03   057  91 R/S
012  76 LBL   035  91 R/S   058  65  ×
013  13  C    036  76 LBL   059  43 RCL
014  42 STO   037  15  E    060  00  00
015  02  02   038  43 RCL   061  95  =
016  91 R/S   039  03  03   062  42 STO
017  65  ×    040  65  ×    063  05  05
018  43 RCL   041  43 RCL   064  91 R/S
019  01  01   042  01  01   065  00  0
020  95  =    043  95  =
021  42 STO   044  42 STO
022  02  02   045  03  03
```

Fig. 9-14. The TR-59 version of the World Count program.

of pages which would be involved if all illustrations with their captions were lumped together without any text.

TI-59: Enter the words per manuscript page and press **A.** Then enter the words per printed page, column or what have you and press **B.** Enter the number of printed pages you need and press **C.** Now press **R/S,** and the display shows *89,600* as the total word count. Press **R/S** again and it reads *358* as the required number of manuscript pages, assuming no illustrations. Now enter **55** for the number of pages that will be occupied by illustrations and press **D.** Press **E** and the display shows *270* for the required number of manuscript pages. Press **R/S** and it displays the new word count: *67,600.*

HP-41C. When the program has been activated the display first identifies it with *WORD COUNT* and then changes to *WORDS: MSS:PGE.* Enter **250** and press **R/S.** The display now asks *WORDS: BOOK:PGE.* Enter **400** and **R/S.** It now asks *BOOK PAGES?* requesting the number of book pages to be filled. Enter **224** and press **R/S,** which produces the display *WORDS= 89,600.* After pressing **R/S** again the display reads *NO ILLUSTR.* and then changes to *MSS:PGE= 358.* Press **R/S** again and the display requests *ILLUSTR.PAGES.* Enter **55** and press **R/S** and the display reads *W.ILLUSTR* then changes to *MSS PGE = 270.* Press **R/S** once more and it will display the new word count, including the illustrations: *WORDS= 67,600.*

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1). Enter words on ms page
2). Press **A**
3). Enter words on printed page
4). Press **B**
5). Enter number of printed pages needed
6). Press **C**, then **R/S**
Display: words needed to fill printed pages
7). Press **R/S**
Display: the number of ms pages
8). Enter space used for illustrations in pages
9). Press **D**, then press **E**
Display: The ms pages need considering illustrations
10). Press **R/S**
11). Total words needed, considering illustrations

HP-41C

Cards: 1
Passes: 2
Size: 006
Display: *WORDS:MSS:PAGE*
1). Enter average word count on ms page; **R/S**
Display: *WORDS:BOOK:PGE*

```
   01♦LBL "PPG 050"        22 STOP         43 RCL 02
    02 "WORD COUNT"        23 RCL 02       44 -
       03 AVIEW            24 RCL 00       45 CHS
       04 PSE              25 /            46 RCL 00
05 "WORDS:MSS:PAGE"        26 STO 04       47 /
       06 PROMPT       27 "NO ILUSTR."     48 STO 04
       07 STO 00           28 AVIEW        49 RCL 00
08 "WORDS:BOOK:PGE"        29 PSE          50 *
       09 PROMPT         30 "MSS:PGE="     51 STO 05
       10 STO 01           31 ARCL 04   52 "MSS PGE="
  11 "BOOK PAGES ?"        32 AVIEW        53 ARCL 04
       12 PROMPT           33 STOP         54 AVIEW
       13 STO 02      34 "ILLUSTR.PAGES"   55 STOP
       14 RCL 01           35 PROMPT       56 "WORDS="
       15 RCL 02           36 STO 03       57 ARCL 05
       16 *             37 "W. ILUSTR."    58 AVIEW
       17 STO 02           38 AVIEW        59 STOP
       18 FIX 0            39 PSE          60 "END"
       19 "WORDS="         40 RCL 03       61 AVIEW
       20 ARCL 02          41 RCL 01       62 STOP
       21 AVIEW            42 *            63 END
```

Fig. 9-15. This program for the HP-41C relates manuscript pages to printed pages.

2). Enter average words on printed page; **R/S**
Display: *BOOK PAGES?*
3). Enter number of pages called for, **R/S**
Display: *WORDS=* total number of words required.
4). Press **R/S**
Display: *NO ILLUSTR.*, then *MSS:PGE=* number of ms pages
5). Press **R/S**
Display: *ILLUSTR.PAGES*
6). Enter space used by illustrations in form of number of pages, **R/S**
Display: *MSS:PGE=* number ms pages
7). Press **R/S**
Display: *WORDS=* number of words considering illustrations

250.	STO	55.	RCL
	0		3
250.		55.	
	R/S	55.	×
400.	STO	55.	RCL
	1		1
400.		400.	
	R/S	400.	=
224.	STO	22000.	
	2	22000.	STO
224.			3
	R/S	22000.	
224.	×	22000.	-
224.	RCL	22000.	RCL
	1		2
400.		89600.	
400.	=	89600.	=
89600.		-67600.	
89600.	STO	67600.	÷
	2	67600.	RCL
89600.			0
	R/S	250.	
89600.	÷	250.	=
89600.	RCL	270.	
	0	270.4	STO
250.			4
250.	=	270.	
358.			R/S
358.4	STO		
	7		
358.			
	R/S		
55.	STO		
	3		
55.			
	R/S		

Fig. 9-16. The Word Count program as run by the TI-59.

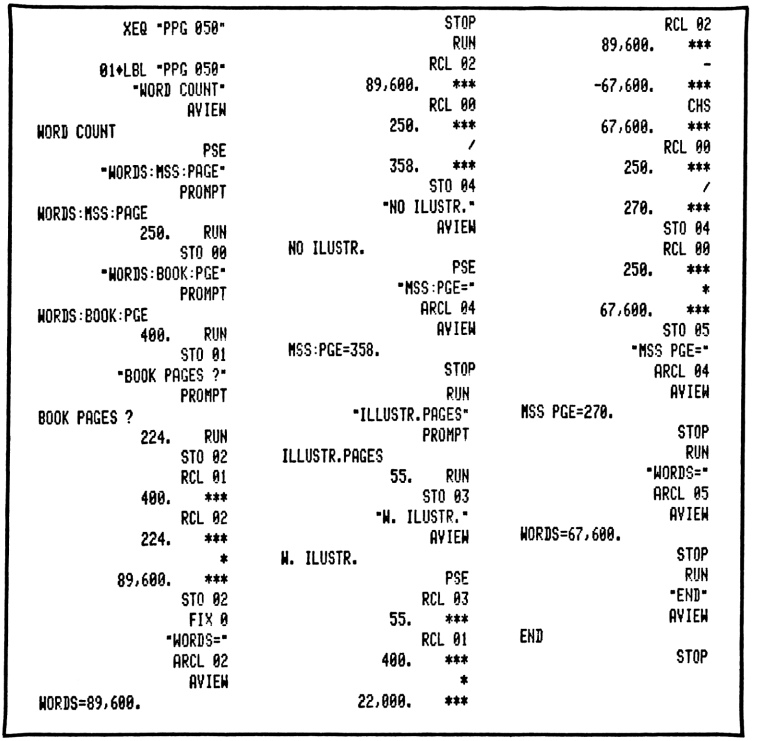
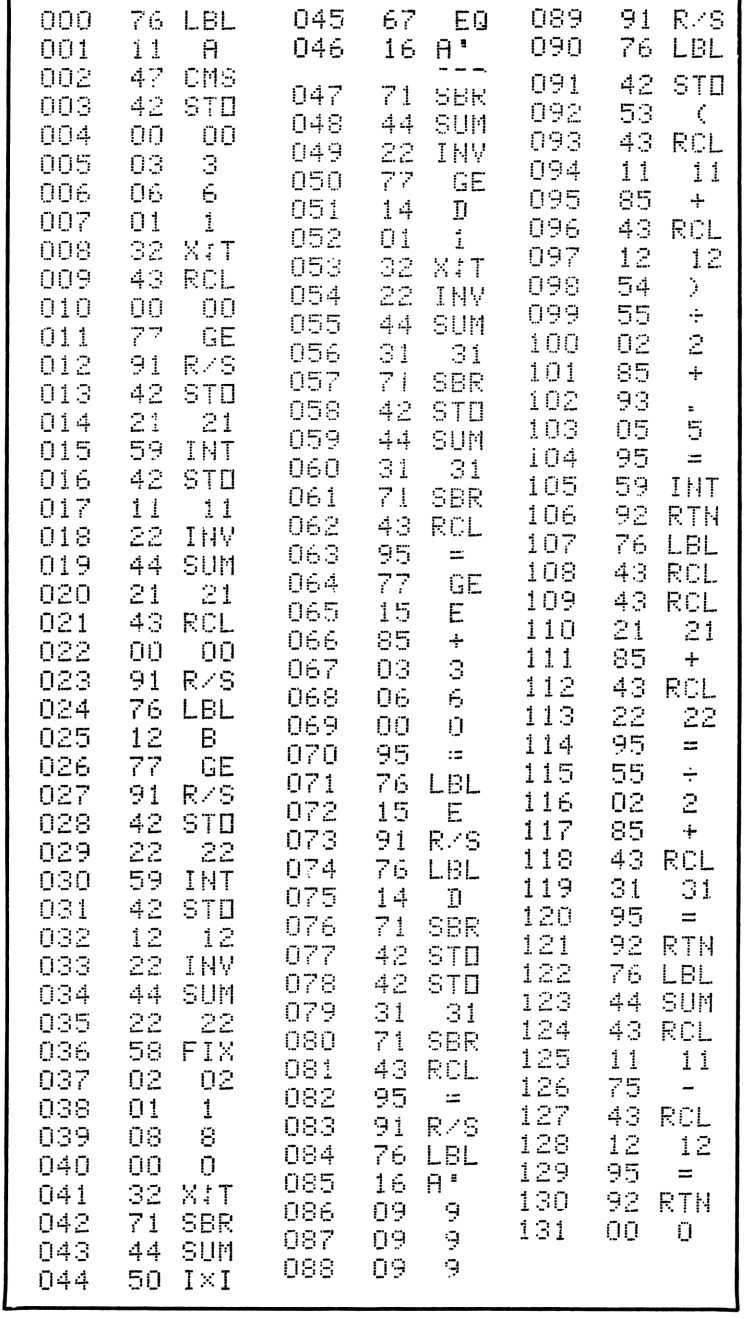


Fig. 9-17. The HP-41C trace of Word Count.

AVERAGE WIND

Pilots of airplanes, and balloonists, as well as captains of boats and ships, often need to figure out the likely direction and velocity of wind half-way between two altitudes or locations for which wind data are available. Here the common means of averaging several figures, dividing the sum by the number of individual entries, doesn't work because we're dealing with 360 degrees. If the wind at one place is blowing from 355 degrees and in the other from 35 degrees, adding those two figures and then dividing by two would result in 195 degrees, which is wrong.

The programs in Figs. 9-18 and 9-19 are designed to determine the correct average direction, the average velocity and, in the event that an impossible combination of figures is entered, such as 90 and 270 degrees, the HP will display *NOT POSSIBLE* and the TI display will read *999.* The wind data is entered in the direction-decimal point-velocity format for both computers, meaning that wind blowing from 290 degrees at 35 knots is entered as **290.35.** The TI will present the result in that same format, while the HP will produce the following displays in quick succession (assuming the input was **290.35** and **245.12**): *AV. DIRECTION* changes to *DEGREES= 268* changes to *AV. VELOCITY* and then *KNOTS= 24.*



```
000  76 LBL    045  67  EQ    089  91 R/S
001  11  A     046  16 A'     090  76 LBL
002  47 CMS                   091  42 STO
003  42 STO    047  71 SBR    092  53  (
004  00  00    048  44 SUM    093  43 RCL
005  03  3     049  22 INV    094  11  11
006  06  6     050  77  GE    095  85  +
007  01  1     051  14  D     096  43 RCL
008  32 X:T    052  01  1     097  12  12
009  43 RCL    053  32 X:T    098  54  )
010  00  00    054  22 INV    099  55  ÷
011  77  GE    055  44 SUM    100  02  2
012  91 R/S    056  31  31    101  85  +
013  42 STO    057  71 SBR    102  93  .
014  21  21    058  42 STO    103  05  5
015  59 INT    059  44 SUM    104  95  =
016  42 STO    060  31  31    105  59 INT
017  11  11    061  71 SBR    106  92 RTN
018  22 INV    062  43 RCL    107  76 LBL
019  44 SUM    063  95  =     108  43 RCL
020  21  21    064  77  GE    109  43 RCL
021  43 RCL    065  15  E     110  21  21
022  00  00    066  85  +     111  85  +
023  91 R/S    067  03  3     112  43 RCL
024  76 LBL    068  06  6     113  22  22
025  12  B     069  00  0     114  95  =
026  77  GE    070  95  =     115  55  ÷
027  91 R/S    071  76 LBL    116  02  2
028  42 STO    072  15  E     117  85  +
029  22  22    073  91 R/S    118  43 RCL
030  59 INT    074  76 LBL    119  31  31
031  42 STO    075  14  D     120  95  =
032  12  12    076  71 SBR    121  92 RTN
033  22 INV    077  42 STO    122  76 LBL
034  44 SUM    078  42 STO    123  44 SUM
035  22  22    079  31  31    124  43 RCL
036  58 FIX    080  71 SBR    125  11  11
037  02  02    081  43 RCL    126  75  -
038  01  1     082  95  =     127  43 RCL
039  08  8     083  91 R/S    128  12  12
040  00  0     084  76 LBL    129  95  =
041  32 X:T    085  16 A'     130  92 RTN
042  71 SBR    086  09  9     131  00  0
043  44 SUM    087  09  9
044  50 I×I    088  09  9
```

Fig. 9-18. The Average Wind program for the TI-59.

```
   01♦LBL "WIND"
02 "AVERAGE WIND"
      03 AVIEW
      04 PSE
      05 CF 00
      06 CF 01
      07 FIX 2
08 "DIR/VEL 1. ?"
      09 PROMPT
      10 STO 00
      11 INT
      12 STO 01
      13 RCL 00
      14 FRC
      15 STO 02
      16 RCL 00
17 "DIR/VEL 2. ?"
      18 PROMPT
      19 STO 03
      20 INT
      21 STO 04
      22 RCL 03
      23 FRC
      24 STO 05
      25 RCL 01
      26 ENTER↑
      27 RCL 04
      28 X>Y?
      29 GTO J

      30♦LBL "K"
      31 180
      32 ENTER↑
      33 RCL 01
      34 RCL 04
      35 -
      36 STO 06
      37 X=Y?
      38 GTO A
      39 -180
      40 ENTER↑
      41 RCL 06
      42 X=Y?
      43 GTO A
      44 181
      45 ENTER↑
      46 RCL 06
      47 X<=Y?
      48 GTO B
      49 RCL 01
      50 RCL 04
      51 +
      52 2
      53 /
      54 180
      55 -
      56 STO 07
      57 RCL 03
      58 RCL 05
      59 +
      60 2
      61 /
      62 100
      63 *
      64 FIX 0
      65 STO 08
      66 RCL 07
      67 ENTER↑
      68 0
      69 X<=Y?
      70 GTO C
      71 SF 00
      72 360
      73 RCL 07
      74 +
      75 STO 10
      76 GTO D
      77 CF 00
      78 SF 01
      79♦LBL B
      80 RCL 01
      81 RCL 04
      82 +
      83 2
      84 /
      85 STO 11
      86 RCL 02
      87 RCL 05
      88 +
      89 2
      90 /
      91 STO 12
      92 100
      93 *
      94 STO 12
      95 GTO H
      96♦LBL C
      97 GTO F
      98♦LBL I
      99 FIX 0
     100 "AV.  ECTION"
         101 AVIEW
         102 PSE
     103 "DEGREE="
         104 ARCL 07
         105 AVIEW
         106 PSE
     107 "AV.VELOCITY"
         108 AVIEW
         109 PSE
         110 "KNOTS="
         111 ARCL 08
         112 AVIEW
         113 STOP
         114♦LBL D
         115 GTO F
         116 RCL 10
         117 STOP
         118♦LBL F
         119 FS? 00
         120 GTO G
         121 FS? 01
         122 GTO H
         123 GTO I

         124♦LBL H
         125 RCL 11
         126 STO 07
         127 RCL 12
         128 STO 08
         129 GTO I

         130♦LBL G
         131 RCL 10
         132 STO 07
         133 GTO I

         134♦LBL J
         135 X<>Y
         136 STO 04
         137 X<>Y
         138 STO 01
         139 GTO "K"

         140♦LBL A
     141 "NOT POSSIBLE"
         142 AVIEW
         143 STOP
         144 .END.
```

Fig. 9-19. An HP-41C program designed to extrapolate the average wind, based on two reports.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 2
Passes: 1	Passes: 3
Partition: 479 59	Size: 012
1). Enter first wind data in DDD.VV format (215.25 represents wind from 215 degrees at 25 knots)	Display: *DIR/VEL 1?*
	1). Enter first wind data; **R/S**
	Display: *DIR/VEL 2?*
2). Press **A**	2). Enter second wind data; **R/S**
3). Enter second wind data	Displays: First *AV.DIRECTION,* then *DEGREE* = and average degree, then *AV. VELOCITY,* then *KNOTS* = and average velocity
4). Press **B**	
Display: The average in terms of direction and velocity	

DAY OF THE WEEK

Here is a program which could be especially useful to history buffs, though it is a lot of fun for just anyone. The program is a long one. The point is to determine the day of the week for any date in history or in the future as well as the time lapse in terms of years, days, hours, minutes and seconds. It involves 368 steps in the TI and 334 in the HP (see Figs. 9-20 and 9-21), and it consists of many individual sections which makes it an interesting subject for a fairly detailed analysis.

To examine exactly how it works and what each section does, let's walk through it from beginning to end, one computer at a time.

TI-59. After turning the computer on and feeding both sides of the magnetic card through the card-reader slot (since the program has more than 240 steps, both sides are needed), key in the month, **2** and press **A.** (We're dealing with Washington's birthday, 2/22/1732.) The *2* will be in the display. Now enter the day, **22** and press **R/S.** The *22* is in the display. Enter the year, **1732** and press **R/S** again. It will take the computer about five seconds to come up with the appropriate answer, *0* in this case, standing for Saturday.

What has happened inside the computer is this: the month was stored in R_{01} (steps 004 and 005), the day was stored in R_{02} (steps 007 and 008) and the year was stored in R_{03} (steps 010 and 011). With those data at hand, the computer went on to the main portion of the program, which consists of the mathematical formula used to have the days of the week displayed in the form of 1 through 6 for Sunday through Friday and 0 for Saturday. This involves the entire sequence of steps from 012 through 312, in other words, 300 steps, which is the reason it takes the computer a few seconds to come up with the answer.

If we enter the current year, **1982,** and press **B** the display will tell us that on February 22, 1982 George Washington would have been *250* years old.

000	25	CLR	048	67	EQ	096	61	GTO
001	47	CMS	049	48	EXC	097	52	EE
002	76	LBL	050	01	1	098	76	LBL
003	11	A	051	02	2	099	39	COS
004	42	STO	052	67	EQ	100	43	RCL
005	01	01	053	49	PRD	101	02	02
006	91	R/S	054	76	LBL	102	85	+
007	42	STO	055	23	LNX	103	01	1
008	02	02	056	43	RCL	104	02	2
009	91	R/S	057	02	02	105	00	0
010	42	STO	058	42	STO	106	95	=
011	03	03	059	04	04	107	42	STO
012	43	RCL	060	61	GTO	108	04	04
013	01	01	061	52	EE	109	61	GTO
014	32	X≷T	062	76	LBL	110	52	EE
015	01	1	063	24	CE	111	76	LBL
016	67	EQ	064	43	RCL	112	30	TAN
017	23	LNX	065	02	02	113	43	RCL
018	02	2	066	85	+	114	02	02
019	67	EQ	067	03	3	115	85	+
020	24	CE	068	01	1	116	01	1
021	03	3	069	95	=	117	05	5
022	67	EQ	070	42	STO	118	01	1
023	37	P/R	071	04	04	119	95	=
024	04	4	072	61	GTO	120	42	STO
025	67	EQ	073	52	EE	121	04	04
026	38	SIN	074	76	LBL	122	61	GTO
027	05	5	075	37	P/R	123	52	EE
028	67	EQ	076	43	RCL	124	76	LBL
029	39	COS	077	02	02	125	28	LOG
030	06	6	078	85	+	126	43	RCL
031	67	EQ	079	05	5	127	02	02
032	30	TAN	080	09	9	128	85	+
033	07	7	081	95	=	129	01	1
034	67	EQ	082	42	STO	130	08	8
035	28	LOG	083	04	04	131	01	1
036	08	8	084	61	GTO	132	95	=
037	67	EQ	085	52	EE	133	42	STO
038	33	X^2	086	76	LBL	134	04	04
039	09	9	087	38	SIN	135	61	GTO
040	67	EQ	088	43	RCL	136	52	EE
041	34	$\sqrt{X}$	089	02	02	137	76	LBL
042	01	1	090	85	+	138	33	X^2
043	00	0	091	09	9	139	43	RCL
044	67	EQ	092	00	0	140	02	02
045	35	1/X	093	95	=	141	85	+
046	01	1	094	42	STO	142	02	2
047	01	1	095	04	04	143	01	1

Fig. 9-20. The TI version of the Day of the Week program.

144	02	2	192	02	02	240	55	÷
145	95	=	193	85	+	241	04	4
146	42	STO	194	03	3	242	95	=
147	04	04	195	03	3	243	42	STO
148	61	GTO	196	03	3	244	05	05
149	52	EE	197	95	=	245	59	INT
150	76	LBL	198	42	STO	246	42	STO
151	34	$\sqrt{X}$	199	04	04	247	05	05
152	43	RCL	200	61	GTO	248	43	RCL
153	02	02	201	52	EE	249	03	03
154	85	+	202	76	LBL	250	75	−
155	02	2	203	52	EE	251	01	1
156	04	4	204	43	RCL	252	95	=
157	03	3	205	03	03	253	55	÷
158	95	=	206	75	−	254	01	1
159	42	STO	207	01	1	255	00	0
160	04	04	208	05	5	256	00	0
161	61	GTO	209	08	8	257	95	=
162	52	EE	210	04	4	258	42	STO
163	76	LBL	211	95	=	259	06	06
164	35	1/X	212	55	÷	260	59	INT
165	43	RCL	213	04	4	261	42	STO
166	02	02	214	95	=	262	06	06
167	85	+	215	22	INV	263	43	RCL
168	02	2	216	59	INT	264	03	03
169	07	7	217	32	X≷T	265	75	−
170	03	3	218	00	0	266	01	1
171	95	=	219	67	EQ	267	95	=
172	42	STO	220	69	OP	268	55	÷
173	04	04	221	68	NOP	269	04	4
174	61	GTO	222	76	LBL	270	00	0
175	52	EE	223	69	OP	271	00	0
176	76	LBL	224	43	RCL	272	95	=
177	48	EXC	225	04	04	273	42	STO
178	43	RCL	226	85	+	274	07	07
179	02	02	227	01	1	275	59	INT
180	85	+	228	95	=	276	42	STO
181	03	3	229	42	STO	227	07	07
182	00	0	230	04	04	278	43	RCL
183	03	3	231	21	2ND	279	04	04
184	95	=	232	68	NOP	280	85	+
185	42	STO	233	76	LBL	281	43	RCL
186	04	04	234	68	NOP	282	03	03
187	61	GTO	235	43	RCL	283	85	+
188	52	EE	236	03	03	284	43	RCL
189	76	LBL	237	75	−	285	05	05
190	49	PRD	238	01	1	286	75	−
191	43	RCL	239	95	=	287	43	RCL

Fig. 9-20. Continued from page 153.

288	06	06	316	11	11	344	43	RCL
289	85	+	317	75	−	345	13	13
290	43	RCL	318	43	RCL	346	95	=
291	07	07	319	03	03	347	42	STO
292	95	=	320	95	=	348	12	12
293	42	STO	321	42	STO	349	91	R/S
294	08	08	322	12	12	350	76	LBL
295	55	÷	323	91	R/S	351	14	D
296	07	7	324	76	LBL	352	43	RCL
297	95	=	325	13	C	353	12	12
298	42	STO	326	43	RCL	354	65	×
299	09	09	327	12	12	355	02	2
300	22	INV	328	55	÷	356	04	4
301	59	INT	329	04	4	357	95	=
302	65	×	330	95	=	358	91	R/S
303	07	7	331	59	INT	359	65	×
304	95	=	332	42	STO	360	06	6
305	42	STO	333	13	13	361	00	0
306	10	10	334	43	RCL	262	95	=
307	75	−	335	12	12	363	91	R/S
308	01	1	336	65	×	364	65	×
309	95	=	337	03	3	365	06	6
310	42	STO	338	06	6	366	00	0
311	10	10	339	05	5	367	95	=
312	91	R/S	340	95	=	368	91	R/S
313	76	LBL	341	42	STO	369	00	0
314	12	B	342	12	12			
315	42	STO	343	85	+			

We can now press **C** and read the number of days, *91,312;* press **D** and read the number of hours, *2,191,488,* press **R/S** for the number of minutes, *131,489,280,* and press **R/S** once more for the seconds, *7,889,356,800.*

All these last calculations which turned the time lapse in years into days, hours, minutes and seconds by the appropriate multiplication were performed in steps 313 through 368.

HP-41C. Press **ON** and start the program by using the user-defined key for this program (I used **shift LOG,** but it could be any other). The display first identifies the program by showing *DAY/WEEK* and then changes to the first request for information: *MONTH?* (steps 06 and 07 in Fig. 6-20). Enter February by punching in **2** and **R/S.** This information is now stored in R_{01} (step 08). The display has changed to *DAY?* Assuming we're trying to find out the day of the week on which George Washington was born, let's key in *22* and **R/S.** This is now stored in R_{02} (step 11). The display asks *YEAR?* We key in the year of Washington's birth, **1732** and **R/S.** This is stored in R_{03} (step 14) but the computer doesn't stop there. It

```
01♦LBL "DW"
02 CLRG
03 "DAY/WEEK"
04 AVIEW
05 PSE
06 "MONTH ?"
07 PROMPT
08 STO 01
09 "DAY ?"
10 PROMPT
11 STO 02
12 "YEAR ?"
13 PROMPT
14 STO 03
15 RCL 01
16 ENTER↑
17 1
18 X=Y?
19 GTO "1"
20 RCL 01
21 ENTER↑
22 2
23 X=Y?
24 GTO "2"
25 RCL 01
26 ENTER↑
27 3
28 X=Y?
29 GTO "3"
30 RCL 01
31 ENTER↑
32 4
33 X=Y?
34 GTO "4"
35 RCL 01
36 ENTER↑
37 5
38 X=Y?
39 GTO "5"
40 RCL 01
41 ENTER↑
42 6
43 X=Y?
44 GTO "6"
45 RCL 01
46 ENTER↑
47 7
48 X=Y?
49 GTO "7"
50 RCL 01
51 ENTER↑
52 8
53 X=Y?
54 GTO "8"
55 RCL 01
56 ENTER↑
57 9
58 X=Y?
59 GTO "9"
60 RCL 01
61 ENTER↑
62 10
63 X=Y?
64 GTO "10"
65 RCL 01
66 ENTER↑
67 11
68 X=Y?
69 GTO "11"
70 RCL 01
71 ENTER↑
72 12
73 X=Y?
74 GTO "12"

75♦LBL "1"
76 RCL 02
77 STO 04
78 GTO A

79♦LBL "2"
80 RCL 02
81 31
82 +
83 STO 04
84 GTO A

85♦LBL "3"
86 RCL 02
87 59
88 +
89 STO 04
90 GTO A

91♦LBL "4"
92 RCL 02
93 90
94 +
95 STO 04
96 GTO A

97♦LBL "5"
98 RCL 02
99 120
100 +
101 STO 04
102 GTO A

103♦LBL "6"
104 RCL 02
105 151
106 +
107 STO 04
108 GTO A

109♦LBL "7"
110 RCL 02
111 181
112 +
113 STO 04
114 GTO A

115♦LBL "8"
116 RCL 02
117 212
118 +
119 STO 04
120 GTO A

121♦LBL "9"
122 RCL 02
123 243
124 +
125 STO 04
126 GTO A

127♦LBL "10"
128 RCL 02
129 273
130 +
131 STO 04
132 GTO A

133♦LBL "11"
134 RCL 02
135 303
136 +
137 STO 04
138 GTO A

139♦LBL "12"
140 RCL 02
141 333
142 +
143 STO 04
144 GTO A

145♦LBL A
146 RCL 03
147 1584
148 -
149 4
150 /
151 FRC
152 ENTER↑
153 0
154 X=Y?
155 GTO B
156 GTO C

157♦LBL B
158 RCL 04
159 1
160 +
161 STO 04
162 GTO C

163♦LBL C
164 RCL 03
165 1
166 -
167 4
168 /
169 STO 05
170 INT
171 STO 05
172 RCL 03
173 1
174 -
175 100
176 /
177 STO 06
178 INT
179 STO 06
180 RCL 03
181 1
182 -
183 400
184 /
185 STO 07
186 INT
187 STO 07
188 RCL 04
189 RCL 03
190 +
191 RCL 05
192 +
193 RCL 06
194 -
195 RCL 07
196 +
197 STO 08
```

Fig. 9-21. The Day of the Week program for the HP-41C.

```
198 7
199 /
200 STO 09
201 FRC
202 7
203 *
204 FIX 0
205 STO 10
206 CLST
207 RCL 10
208 ENTER↑
209 0.9999997
210 X=Y?
211 GTO "Z"
212 RCL 10
213 ENTER↑
214 2.0000001
215 X=Y?
216 GTO "Y"
217 RCL 10
218 ENTER↑
219 2.9999998
220 X=Y?
221 GTO "X"
222 RCL 10
223 ENTER↑
224 4.0000002
225 X=Y?
226 GTO "W"
227 RCL 10
228 ENTER↑
229 4.9999999
230 X=Y?
231 GTO "V"
232 RCL 10
233 ENTER↑
234 6.0000003
235 X=Y?
236 GTO "U"
237 RCL 10
238 ENTER↑
239 0.0000000
240 X=Y?
241 GTO "T"

242↑LBL F
243 "TIME TO YR. ?"
244 PROMPT
245 STO 11
246 RCL 03
247 -
248 FIX 0
249 STO 12
250 "YEARS="
251 ARCL 12
252 AVIEW
253 STOP
254 RCL 12
255 4
256 /
257 INT
258 STO 13
259 RCL 12
260 365
261 *
262 STO 12
263 RCL 13
264 +
265 STO 12
266 "DAYS="
267 ARCL 12
268 AVIEW
269 STOP
270 RCL 12
271 24
272 *
273 STO 12
274 "HRS="
275 ARCL 12
276 AVIEW
277 STOP
278 RCL 12
279 60
280 *
281 STO 12
282 "MINS="
283 ARCL 12
284 AVIEW
285 STOP

286 RCL 12
287 60
288 *
289 STO 12
290 "SECS="
291 ARCL 12
292 AVIEW
293 STOP
294 GTO "S"

295↑LBL "Z"
296 "SUNDAY"
297 AVIEW
298 STOP
299 GTO F

300↑LBL "Y"
301 "MONDAY"
302 AVIEW
303 STOP
304 GTO F
305↑LBL "X"
306 "TUESDAY"
307 AVIEW
308 STOP
309 GTO F

310↑LBL "W"
311 "WEDNESDAY"
312 AVIEW
313 STOP
314 GTO F

315↑LBL "V"
316 "THURSDAY"
317 AVIEW
318 STOP
319 GTO F

320↑LBL "U"
321 "FRIDAY"
322 AVIEW
323 STOP
324 GTO F

325↑LBL "T"
326 "SATURDAY"
327 AVIEW
328 STOP
329 GTO F

330↑LBL "S"
331 "END"
332 AVIEW
333 STOP
334 .END.
```

goes right on and after a second or two displays *SATURDAY.* In other words, February 22, 1732 was a Saturday.

So far it sounds quite simple. It becomes less so when we delve into what the computer had to go through to arrive at that conclusion.

The last step before the start of the actual calculation was step 14 in which the year was stored in R_{03}. From there the computer went to the consecutive numbered steps. It recalls the data in R_{01} (the month) and uses steps 16 through 19 to determine whether the month in question is January. Since it is not, it goes on and uses steps 20 through 24 to determine whether the month is February. It is, and it therefore follows the command in step 24 **(GTO 02)** and starts to search the entire program for a section labeled **LBL 02.** (If the month in question had been December, for instance, the computer would have continued to examine each month until getting to step 74.)

We find that **LBL 02** (the printer shows it as LBL "2") is in step 79 and in checking the section which consists of steps 79 through 84, we find that its purpose is to tell the computer that the month preceding February has 31 days and that those days will eventually have to be taken into consideration. The section ends with the command **GTO A:** go to the section labeled A.

Again the computer starts to search until it finds that **LBL A** is located at step 145. This section, which goes from step 145 to step 156 is used by the computer to determine whether 1732 was a leap year. Since Washington was born on February 22 it actually makes no difference, but if he had been born at any time between March 1 and December 31, the extra day in February would have been important in all subsequent calculations. (1732 was, in fact, a leap year.) Steps 154, 155 and 156 are used to force the computer to make a decision. The question posed is this: when we deduct 1584, the first leap year after the adoption of the Gregorian calendar, from 1732 can the result be divided by four without producing decimal fractions. By asking "do the fractions **(FRC)** equal 0 (steps 151, 152, 153 and 154)," the computer is given a choice between going to **LBL B** or **LBL C.** If the answer to the question is yes (**FRC** equals **0**) it must go to **LBL B.** If the answer is no, in other words, there are fractions, it must go to **LBL C.**

In our case it goes to **LBL B** which is the next section, consisting of steps 157 through 162. The purpose of this step is to add one day to all calculations involving dates after March 1 of that year. Having accomplished that the computer is told to go to **LBL C.**

We find **LBL C** to be the next section, starting with step 163 and finishing with step 241. This is the real guts of the program, the section in which the computer does the lion's share of the work. Steps 164 through 205 represent a known (and published, it is not my invention) mathematical formula which determines the day of the week for any data after the adoption of the Gregorian calendar in 1582. The result is expressed in the form of digits: 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday and 0=Saturday. But since the HP-41C can display words, we might just as well transform those numbers into a display of the actual name of the day. All that needs to be done, since the number determining the day is stored in R_{10} (step 205), is to compare the data in R_{10} with any number from 0 to 6 and then command the computer to display the appropriate name. Very simple.

If it's so simple, why did it take me days to accomplish this task? I will briefly explain what happened, because it is a valuable lesson to learn. The program was working perfectly, displaying *1* for Sunday and so on. The computer was in a **FIX 0** condition, meaning that only full numbers (integers) and no fractions were being displayed. I programmed the computer to compare the contents of R_{10} with numbers 1, 2, 3, 4, 5, 6, and 0 and the result simply made no sense at all. I'd inspect the contents of R_{10} by recalling them into the display and there, clear as anything, would be a number such as 1.

I'd then compare it with 1 (**RCL 10 ENTER ↑ 1 x=y?**) and the computer would stubbornly answer *NO.* I really don't remember how long I fooled around with this problem, but it was several days until I suddenly had the bright idea to change **FIX 0** to **FIX 7,** and the dawn came. That 1 stored in R_{10} wasn't a 1 at all, it was actually 0.9999997. True, that's awfully close to 1, but not close enough for the computer.

As you can see by looking at steps 209 through 239, I put in the exact amounts represented by the different numbers stored in R_{10} and all my troubles disappeared.

In our case, since February 22 was a Saturday, the computer had to go through all those steps, comparing the data stored in R_{10} with each of the numbers before finally being told to go to T **(GTO T)** in step 241. We find the section **LBL T** way down toward the end in steps 325 through 329 and it is the section which is responsible for the word *SATURDAY,* which is now in the display.

This concludes the first and most important portion of the program. We're now ready to go on to the second portion, dealing with elapsed time. The assumption is that we're always comparing two identical dates in different years. If the year with which the comparison is being made is later than the year used in the foregoing calculation, the resulting data will be positive. If the year used for comparison is earlier, the resulting data will be negative, preceded by a minus sign.

Press **R/S** and the display changes to *TIME TO YR.?* asking for the year to which we want to compare 1732. Let's punch in the present, **1982** and **R/S**. The display comes up with *YEARS= 250,* meaning that February 22, 1982 was George Washington's 250th birthday. Now press **R/S** again and we find that those 250 years consisted of 91,312 days. Press **R/S** again and the display shows *HRS.=2,191,488.* As long as we're at it, press **R/S** for *MINS=131,489,280* and once more for *SECS=7,889,356,800.*

All of these calculations are contained in the section labeled F, steps 242 through 294. Here the computer simply deducts one year from the other and then multiplies the result by the number of days in a year (365.25 or 365 plus 1 for each leap year) and then multiplies that result by 24 for hours, 60 for minutes and 60 again for seconds. Having accomplished all that, the computer is commanded to go to section **LBL S** which it finds at step 330 to put the word *END* into display.

You will have noticed that the different sections of the program are not necessarily placed in consecutive order. During the design and editing of a program, certain portions are added while others are eliminated. Considering the incredible speed with which the computer is able to locate what it is looking for, it really makes no difference. Still, when it comes to some extremely complicated programs which require that the computer perform lengthy series of consecutive calculations, the amount of time required to obtain the displayed result can be shortened somewhat by placing the various sections of the program into some sort of logical order.

Program Steps

TI-59	HP-41C
Cards: 1 Passes: 2 Partition: 479 59 1). Enter month 2). Press **A** 3). Enter day 4). Press **R/S** 5). Enter year 6). Press **R/S** Display: number representing day of the week (*1*=Sunday, *2*=Monday, *3*=Tuesday, *4*=Wednesday, *5*=Thursday, *6*=Friday, *0*=Saturday) 7). Enter year to which time difference is to be computed 8). Press **B** Display: Time difference in years 9). Press **C** Display: ditto, in days 10). Press **D** Display: ditto in hours; **R/S** in minutes; **R/S** in seconds	Cards: 4 Passes: 7 Size: 014 Display: *MONTH ?* 1. Enter the month; **R/S** Display: *DAY ?* 2. Enter the day; **R/S** Display: *YEAR ?* 3. Enter the year, **R/S** Display: *SUNDAY* (or any other day) 4. Press **R/S** Display: *TIME TO YR.?* 5. Enter the year to which time difference is to be figured; **R/S** Display: *YEARS*= time difference in years 6. Press **R/S** Display: *DAYS*= time difference in days 7. Press **R/S** Display: *HRS*= time difference in hours 8. Press **R/S** Display: *MINS*= time difference in minutes 9. Press **R/S** Display: *SECS*= time difference in seconds

HOW MUCH BEER IN A BARREL?

This is a program dealing with volumetric calculations. This one is complicated because of the peculiar shape of a barrel. The program can be used for any container with the same size top and bottom where the greatest or smallest circumference is half-way between top and bottom.

To perform the necessary computations the computer needs only three input data: the radius of the top (or bottom) and the radius at the widest point. Since it is fairly difficult to measure an exact radius figure for the top by placing a tape measure across the top, and it is even more difficult to arrive at a radius figure for the center area. To simplify this I have included steps to convert easy-to-obtain figures into the radius data needed by the computer. The third number to input is the height of the barrel.

The program will accept the diameter measure for the top (or bottom) and use the circumference measure for the center. Both these measures can be obtained with ease.

TI-59. The program for the TI (Fig. 9-22) assigns the three input data to keys **A, B** and **C,** and then uses keys **D** and **E** to obtain the computed results.

Steps 000 through 003 clear left-over data from the registers and set the computer to display no decimals. (If you want exact amounts, change that number in step 003 from 00 to 01 or 02 or whatever number of decimals you want to see.)

Steps 004 through 018 are used to accept the input data and to store it in R_{00}, R_{02} and R_{03}. Steps 019 through 078 are run by the computer when key **D** is pressed. These steps convert the input data from diameter and circumference to radius, and then perform the mathematical calculation necessary to arrive at the volume in cubic inches (or whatever measure was used for the input data).

Pressing key **E** then activates the rest of the program, steps 079 through 102, which converts cubic inches to gallons. (If another conversion is desired, such as cubic centimeters to liters or cubic inches to liters, the data used in this last calculation must be changed.)

HP-41C. After identifying the program (Fig. 9-23) with *BEER/BARREL* the display asks *END DIAMETER.* When that has been entered it requests *MID CIRCUMFER* and *HEIGHT?* The computer is now ready to go to work. In steps 14 through 17 it converts diameter into radius, and in steps 18 through 23 it converts circumference into radius.

Steps 24 through 46 represent the mathematical formula used to determine the volume of a barrel from the previously input measurements. Steps 47 through 50 display the result in the form of cubic inches. (If the original data is entered as centimeters, the result would be in cubic centimeters, but the next section of the program would become inaccurate.)

The last steps, 51 through 68, are used to convert cubic inches to US gallons and to display that result.

As an example: If we enter **12** inches for the diameter of the top, **45** inches for the circumference at the center, and **18** inches for the height of the barrel, the computer will respond with *CU.IN. = 2,612* followed by *GAL. = 11,* or if we have the computer set to **FIX 2,** with *GAL. = 11.31.*

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter end diameter
2. Press **A**
3. Enter mid circumference
4. Press **B**
5. Enter height
6. Press **C**
7. Press **D**
Display: Cubic inches of barrel content
8. Press **E**
Display: Volume in gallons

HP-41C

Cards: 1
Passes: 2
Size: 012
Display: *END DIAMETER*
1. Enter end diameter; **R/S**
Display: *MID. CIRCUMFER.*
2. Enter mid circumference; **R/S**
Display: *HEIGHT?*
3. Enter height; **R/S**
Display: *CU.IN.* = and volume in cubic inches
4. Press **R/S**
Display: *GAL.* = and volume in gallons

```
000  47 CMS     035  42 STO     070  42 STO
001  25 CLR     036  02  02     071  07  07
002  58 FIX     037  43 RCL     072  65  ×
003  00  00     038  00  00     073  43 RCL
004  76 LBL     039  33 X²      074  06  06
005  11  A      040  42 STO     075  95  =
006  42 STO     041  01  01     076  42 STO
007  00  00     042  43 RCL     077  08  08
008  91 R/S     043  02  02     078  91 R/S
009  76 LBL     044  33 X²      079  76 LBL
010  12  B      045  65  ×      080  15  E
011  42 STO     046  02  2      081  43 RCL
012  02  02     047  95  =      082  08  08
013  91 R/S     048  42 STO     083  65  ×
014  76 LBL     049  04  04     084  53  (
015  13  C      050  43 RCL     085  04  4
016  42 STO     051  03  03     086  93  .
017  03  03     052  65  ×      087  03  3
018  91 R/S     053  89  π      088  02  2
019  76 LBL     054  95  =      089  09  9
020  14  D      055  42 STO     090  65  ×
021  43 RCL     056  05  05     091  01  1
022  00  00     057  43 RCL     092  00  0
023  55  ÷      058  01  01     093  45 Y×
024  02  2      059  85  +      094  03  3
025  95  =      060  43 RCL     095  94 +/-
026  42 STO     061  04  04     096  54  )
027  00  00     062  95  =      097  95  =
028  43 RCL     063  42 STO     098  58 FIX
029  02  02     064  06  06     099  01  01
030  55  ÷      065  43 RCL     100  42 STO
031  89  π      066  05  05     101  05  05
032  55  ÷      067  55  ÷      102  91 R/S
033  02  2      068  03  3      103  00  0
034  95  =      069  95  =
```

Fig. 9-22. The volume of a beer barrel is figured by the TI-59.

```
01♦LBL "PPG 045"
02 "BEER/BARREL"
03 AVIEW
04 PSE
05 "END DIAMETER"
06 PROMPT
07 STO 00
08 "MID.CIRCUMFER."
09 PROMPT
10 STO 02
11 "HEIGHT ?"
12 PROMPT
13 STO 03
14 RCL 00
15 2
16 /
17 STO 00
18 RCL 02
19 PI
20 /
21 2
22 /
23 STO 02
24 RCL 00
25 X↑2
26 STO 01
27 RCL 02
28 X↑2
29 2
30 *
31 STO 04
32 RCL 03
33 PI
34 *
35 STO 05
36 RCL 01
37 RCL 04
38 +
39 STO 06
40 RCL 05
41 3
42 /
43 STO 07
44 RCL 06
45 *
46 STO 08
47 "CU.IN.="
48 ARCL 08
49 AVIEW
50 STOP
51 4.329
52 STO 09
53 10
54 ENTER↑
55 -3
56 Y↑X
57 STO 10
58 RCL 09
59 *
60 STO 11
61 RCL 08
62 *
63 STO 08
64 "GAL="
65 ARCL 08
66 AVIEW
67 STOP
68 END
```

Fig. 9-23. How Much Beer in a Barrel on the HP-41C.

STANDARD TEMPERATURE

It is well known that the air gets colder as we climb higher. For some reason, certain temperatures have been accepted internationally as standard, even though they may not be the temperatures which occur most frequently. (The reason for international accord is that temperatures on the ground as well as at altitudes are important in aviation which is an internationalized activity.)

The standard temperature at sea level is 59 degrees Fahrenheit or 15 degrees centigrade. The program in Figs. 9-24 and 9-25 will compute the standard temperature for any altitude or elevation. Key in the altitude or elevation you are interested in and the first result displayed will be the standard temperature in centigrade. Pressing **R/S** then converts that to degrees Fahrenheit.

For example, enter **20,320,** the altitude of Mount McKinley, the highest mountain in the United States. The first answer displayed is *–25.6,* representing degrees centigrade. Pressing **R/S** results in *–14.2,* the

```
000  76 LBL    011  75  -     022  08  8
001  11  A     012  01  1     023  35 1/X
002  42 STO    013  05  5     024  95  =
003  00  00    014  95  =     025  85  +
004  65  ×     015  94 +/-    026  03  3
005  00  0     016  58 FIX    027  02  2
006  93  .     017  01  01    028  95  =
007  00  0     018  91 R/S    029  42 STO
008  00  0     019  55  ÷     030  01  01
009  02  2     020  01  1     031  91 R/S
010  95  =     021  93  .     032  00  0
```

Fig. 9-24. The TI version of the Standard Temperature program.

standard temperature at that elevation in degrees Fahrenheit. We might enter **−282,** the elevation at the lowest point in the United States, Death Valley. The first answer displayed is *15.6* degrees centigrade. Pressing **R/S** results in *60.0* degrees Fahrenheit.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 002
1. Enter elevation or altitude	Display: *ELEVATION?*
2. Press **A**	1. Enter elevation or altitude; **R/S**
Display: Temperature, standard, for that altitude or elevation, in degrees C.	Display: *DEG:C.* = and standard temperature for that elevation
3. Press **R/S**	2. Press **R/S**
Display: The same in degrees F.	Display: *DEG:F.* = and temperature in degrees F.

```
        01♦LBL "STA"      12 -          23 /
02 "STAND. TEMP. C."      13 CHS        24 32
        03 "PSE"          14 STO 01     25 +
        04♦LBL 00         15 FIX 1      26 STO 01
    05 "ELEVATION ?"      16 "DEG:C.="  27 "DEG.F.="
        06 PROMPT         17 ARCL 01    28 ARCL 01
        07 STO 00         18 AVIEW      29 AVIEW
        08 0.002          19 STOP       30 STOP
        09 *              20 RCL 01     31 GTO 00
        10 STO 00         21 1.8        32 .END.
        11 15             22 1/X
```

Fig. 9-25. Standard Temperatures figured by the HP-41C.

DIRECT DISTANCE

This program determines the straight-line distance, "as the crow flies," between two points, the latitude/longitude coordinates of which are known.

Because I have kept this version as simple as possible, it can be used with reasonable accuracy only within the contiguous United States or in other parts of the earth which fall between the 30th and 50th parallel. It is most accurate around the 35th parallel, the accuracy diminishing the farther away we go. It is not equipped to deal with distances which involve crossing the international dateline.

The reason is that the distances between longitude vary from latitude to latitude. The program takes the distance between longitudes at the 35th latitude as the basis for its calculations.

TI-59. Enter the starting latitude (Fig. 9-26), **32.51,** and press **A.** Then enter the starting longitude, **96.52** and press **B.** Now enter the destination latitude, **37.37,** and press **C,** and finally enter the destination longitude, **122.22** and press **D.**

The computer is ready to go, so press **E;** after a moment's hesitation it puts *1404* in the display.

HP-41C. Initializing the program (Fig. 9-27) produces the display *DIRECT DIST.* which changes to *START LAT?*, asking for the latitude of the first location. Let's take Dallas and enter 32.509, which represents 32 degrees and 50.9 minutes. **32.51** saves one key stroke and is certainly accurate enough.

As we press **R/S,** the display asks for *START LONG.?* Enter **96.52,** representing 96 degrees, 52 minutes longitude.

The display next asks for *DEST. LAT.?* Let's pick San Francisco as the other location, and key in **37.37.** When the display asks *DEST. LONG.?* key in **122.22.**

The computer now has all the information it needs, and it can compute. Press **R/S** and after a moment the display reads *MILES=1,404,* which is pretty close to being on the button.

Pressing **R/S** again will put *AGAIN? 0=YES* into the display. If we want to determine the direct-line distance between two other points, all we do now is press **0** and **R/S.** Otherwise, pressing **R/S** results in *END.*

Steps 01 through 23 are used to identify the program and to accept and store the latitude/longitude data for the two locations. You will notice that after each entry there is a step **HR** (steps 09, 14, 19 and 24), the command to transform the entered data from degrees and minutes to degrees and decimal fractions of degrees, the format needed for the subsequent calculations. These calculations, steps 26 through 61, determine the number of longitudes and latitudes, including fractions, which lie between the two locations. They multiply that result by the distance between latitudes to determine the amount of north-south distance, and by the distance between longitudes (at the 35th parallel) to determine the east-west distance. The

computer has theoretically two lines at right angles to one another, one representing the distance north-south, the other representing the distance

```
000  47 CMS   036  50 I×I   071  03  3
001  25 CLR   037  42 STO   072  08  8
002  76 LBL   038  05  05   073  95  =
003  11  A    039  43 RCL   074  75  -
004  58 FIX   040  01  01   075  06  6
005  02  02   041  75  -     076  07  7
006  88 DMS   042  43 RCL   077  95  =
007  42 STO   043  03  03   078  50 I×I
008  01  01   044  95  =     079  42 STO
009  91 R/S   045  50 I×I   080  10  10
010  76 LBL   046  42 STO   081  43 RCL
011  12  B    047  06  06   082  05  05
012  88 DMS   048  43 RCL   083  65  ×
013  42 STO   049  01  01   084  43 RCL
014  02  02   050  85  +     085  10  10
015  91 R/S   051  43 RCL   086  95  =
016  76 LBL   052  03  03   087  42 STO
017  13  C    053  95  =     088  11  11
018  88 DMS   054  55  ÷     089  43 RCL
019  42 STO   055  02  2     090  11  11
020  03  03   056  95  =     091  32 X↕T
021  91 R/S   057  42 STO   092  43 RCL
022  76 LBL   058  07  07   093  08  08
023  14  D    059  43 RCL   094  22 INV
024  88 DMS   060  06  06   095  37 P/R
025  42 STO   061  65  ×     096  42 STO
026  04  04   062  06  6     097  12  12
027  91 R/S   063  07  7     098  58 FIX
028  76 LBL   064  95  =     099  00  00
029  15  E    065  42 STO   100  32 X↕T
030  43 RCL   066  08  08   101  42 STO
031  02  02   067  43 RCL   102  12  12
032  75  -    068  07  07   103  91 R/S
033  43 RCL   069  65  ×     104  00  0
034  04  04   070  93  .
035  95  =
```

Fig. 9-26. The TI-59 version of the Direct Distance.

```
      01*LBL "RL"      26 RCL 02          51 ABS
 02 "DIRECT DIST."     27 RCL 04          52 STO 10
      03 AVIEW         28 -               53 RCL 05
      04 PSE           29 ABS             54 RCL 10
      05*LBL 00        30 STO 05          55 *
 06 "START LAT. ?"     31 RCL 01          56 STO 11
      07 PROMPT        32 RCL 03          57 RCL 08
      08 STO 01        33 -               58 ENTER↑
      09 HR            34 ABS             59 RCL 11
      10 STO 01        35 STO 06          60 R-P
 11 "START LONG. ?"    36 RCL 01          61 STO 12
      12 PROMPT        37 RCL 03          62 FIX 0
      13 STO 02        38 +               63 "MILES="
      14 HR            39 2               64 ARCL 12
      15 STO 02        40 /               65 AVIEW
 16 "DEST. LAT. ?"     41 STO 07          66 STOP
      17 PROMPT        42 RCL 06     67 "AGAIN? 0=YES"
      18 STO 03        43 67              68 PROMPT
      19 HR            44 *               69 X=0?
      20 STO 03        45 STO 08          70 GTO 00
 21 "DEST. LONG. ?"    46 RCL 07          71 "END"
      22 PROMPT        47 0.38            72 AVIEW
      23 STO 04        48 *               73 STOP
      24 HR            49 67              74 END
      25 STO 04        50 -
```

Fig. 9-27. Direct Distance for the HP-41C

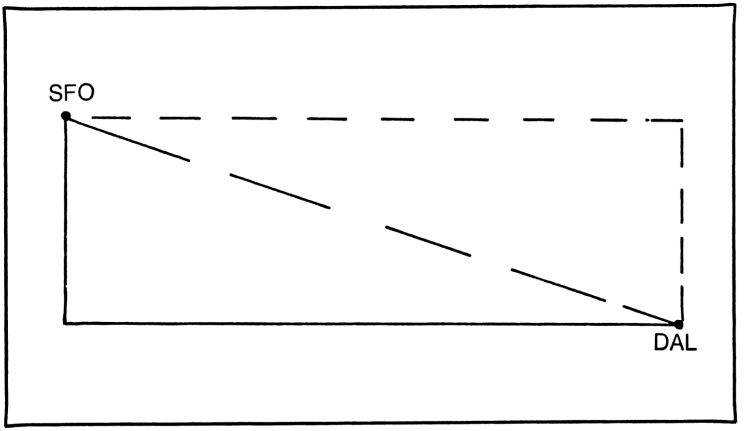


Fig. 9-28. The polar/rectangular conversion capability of the two computers.

east-west. These lines, or the distances they represent, are stored in R_{08} for north-south, and in R_{11} for east-west. In our Dallas to San Francisco example, the north-south distance is 319 miles, and the east-west distance is 1,367 miles.

The computer uses the polar conversion capability (steps 57 through 59) to determine the length of a diagonal line, which we might imagine to be drawn from the right-angle corner of the two lines to the opposite corner of what would be a rectangle if the two opposite lines were also drawn (Fig. 9-28). Performing that calculation results in 1,404, the figure now stored in R_{12} and displayed as the result representing direct-distance miles between those two points.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 013
1. Enter starting latitude,	Display: *START LAT.?*
2. Press **A**	1. Enter start latitude; **R/S**
3. Enter starting longitude	Display *START LONG.?*
4. Press **B**	2. Enter start longitude; **R/S**
5. Enter destination latitude	Display: *DEST. LAT?*
6. Press **C**	3. Enter destination latitude; **R/S**
7. Enter destination longitude	Display: *DEST. LONG.?*
8. Press **D**	4. Enter destination longitude;
9. Press **E**	**R/S**
Display: Distance	Display: *MILES*= and distance

DENSITY ALTITUDE, SCIENTIFIC OR SIMPLE

Density altitude is a term peculiar to aviation and probably unknown to or little understood by those who are not pilots. When temperature is above what is considered normal, the air is less dense than it would be at lower temperatures, and the effect is the same as if we were at a higher altitude at normal temperatures. In other words, when the temperature is 90 degrees F. at sea level, the density of the air is the same as it would be at an elevation of 1,970 feet at normal temperature.

The density of the air, in other words, the density altitude, has an effect on the way the reciprocating engines in our cars and airplanes perform and on the fuel mixture they need to be able to perform. It also has an effect on the amount of exercise we can do without becoming breathless and exhausted.

The primary reason for including the programs on determining density altitude is that they are ideally suited to demonstrate a number of subjects which tend to become important as you write more and more personal programs.

Let's first look at the scientific approach to determining density altitude. The mathematical formula looks like this:

Density altitude = (145426 × (1- (((288.15 - the actual elevation × .001981) / 288.15)$^{5.2563}$ / ((273.15 + temperature in degrees C.) / 288.15)) $^{.235}$))

```
000  76 LBL    038  01  01    076  54  )
001  11  A      039  65  ×     077  45 Y×
002  58 FIX    040  93  .     078  93  .
003  00  00     041  00  0     079  02  2
004  42 STO    042  00  0     080  03  3
005  01  01     043  01  1     081  05  5
006  91 R/S    044  09  9     082  54  )
007  76 LBL    045  08  8     083  54  )
008  12  B      046  01  1     084  59 INT
009  42 STO    047  54  )     085  91 R/S
010  02  02     048  55  ÷     086  76 LBL
011  91 R/S    049  43 RCL    087  16 A'
012  76 LBL    050  03  03    088  42 STO
013  13  C      051  54  )     089  02  02
014  53  (      052  45 Y×    090  75  -
015  01  1      053  05  5     091  03  3
016  04  4      054  93  .     092  02  2
017  05  5      055  02  2     093  95  =
018  04  4      056  05  5     094  55  ÷
019  02  2      057  06  6     095  01  1
020  06  6      058  03  3     096  93  .
021  65  ×      059  55  ÷     097  08  8
022  53  (      060  53  (     098  95  =
023  01  1      061  53  (     099  42 STO
024  75  -      062  02  2     100  02  02
025  53  (      063  07  7     101  91 R/S
026  53  (      064  03  3     102  00  0
027  53  (      065  93  .
028  02  2      066  01  1
029  08  8      067  05  5
030  08  8      068  85  +
031  93  .      069  43 RCL
032  01  1      070  02  02
033  05  5      071  54  )
034  42 STO    072  55  ÷
035  03  03     073  43 RCL
036  75  -      074  03  03
037  43 RCL    075  54  )
```

Fig. 9-29. The formula with its seven sets of parentheses can easily be handled by the TI-59.

Written like that it is too cumbersome to get a clear picture of what is involved. So let me try once more. This time, density altitude is expressed as DA, the actual elevation is expressed as PA (pressure altitude) and the temperature in degrees C. is expressed as TC.

$$DA=(145426\times(1-(((288.15-PA\times.001981)/288.15)^{5.2563}$$
$$/((273.15+TC)/288.15))^{.235}))$$

a b cde f g

hi j kl mn

Here the formula is more compact. The parentheses are identified by letters beneath each. The reason for this will become apparent when we try to convert this formula into a sequence of key strokes for the HP-41C.

TI-59. Entering this formula in the form of a computer program into the TI-59 is not particularly complicated. As shown in Fig. 9-29, it simply takes it one step at a time from left to right. Steps 000 through 011 enter the actual elevation (or pressure altitude) into computer memory (R_{01}) and the temperature in degrees C. into R_{02}. Steps 086 through 101 were an afterthought; here the operator can place the temperature in degrees F. into the computer and it will automatically convert them into degrees C. and then store the converted value in the appropriate memory register (R_{02}). The rest of the steps represent the actual formula. Steps 028 through 035 place the figure 288.15 into R_{03}. That figure is used repeatedly in the program and it is more convenient to recall the figure from memory than key it in each time. The y^x in steps 052 and 077 is the means by which the preceding content of the parentheses is raised to the appropriate power. INT in step 084 stands for integer. It gets rid of all the unwanted decimals and places the full number, the integer, into the display, representing the density altitude under the conditions initially entered into the computer.

Here is what happens when we run the program: The altitude in Santa Fe is 7,000 feet. Key in **7000** and **A.** The average temperature in the shade on a nice summer day is about 78 degrees F. Key in **78** and **2nd A.** The display shows *26,* the temperature in degrees C. Now press **C** and after a bit of calculation the display shows that the density altitude under those conditions is *9,747* feet.

HP-41C. Now we get to the tricky part. The HP-41C does not have any provision for parentheses. With a total of 14 parentheses in this formula, we'll have to break it down somehow into components which can be handled by the computer. The simplest way to do this is to calculate the contents of each pair of parentheses separately, store those contents in a memory register, and then perform the final calculation using the contents of those registers.

To start with take the pairs of parentheses which have no other parentheses between them. (The lower-case letters identify the parentheses in question.)

1. Between E and F; placing everything in the order in which the computer can use it: **288.15 PA -- .001981 ×** (no = sign) **STO 05**

```
    01♦LBL "DP"
    02 CLRG
03 "DENSITY ALT."
    04 AVIEW
    05 PSE
 06 "PRESS. ALT."
    07 PROMPT
    08 STO 01
    09 0
 10 "TEMP. F. ?"
    11 PROMPT
    12 STO 02
    13 ENTER↑
    14 0
    15 X=Y?
    16 GTO 01
    17 RCL 02
    18 32
    19 -
    20 1.8
    21 /
    22 STO 02
    23 GTO 02
    24♦LBL 01
 25 "TEMP. C. ?"
    26 PROMPT
    27 STO 02
    28♦LBL 02
    29 288.15
    30 STO 03
    31 RCL 01
    32 .001981
    33 *
    34 RCL 03
    35 -
    36 CHS
    37 STO 05
    38 RCL 03
    39 /
    40 STO 06
    41 273.15
    42 RCL 02
    43 +
    44 STO 07
    45 RCL 03
    46 /
    47 STO 08
    48 RCL 06
    49 ENTER↑
    50 5.2563
    51 Y↑X
    52 RCL 08
    53 /
    54 STO 09
    55 ENTER↑
    56 .235
    57 Y↑X
    58 STO 10
    59 1
    60 RCL 10
    61 -
    62 STO 11
    63 145426
    64 *
    65 STO 12
 66 "D.ALT.="
    67 ARCL 12
    68 AVIEW
    69 STOP
    70 "END"
    71 AVIEW
    72 STOP
    73 FIX 0
    74 VIEW 12
    75 STOP
    76 END
```

Fig. 9-30. With the HP-41C, parenthetic equations must be determined individually before the final calculation can be done.

2. Between I and J: **273.15 TC + STO 07**
3. Between D and G: **RCL 05 288.15 / STO 06**
4. Between H and K: **RCL 07 288.15 / STO 08**
5. Between C and L: **RCL 06$^{5.2563}$ RCL 08 / STO 09**
6. Between B and M: **RCL 09$^{.235}$ STO 10: 1 RCL 10 - STO 11**
7. Between A and N: **RCL 11 145426 × STO 12**

The contents of R_{12} are the final result. As you can see by the above, it requires a considerable amount of surgery and rearranging to make such a mathematical formula palatable to a computer which is designed to function by using RPN (reverse Polish notation).

Here is what takes place (see Fig. 9-30).

The display identifies the program with *DENSITY ALT.* then changes to *PRESS. ALT.*, asking for the pressure altitude or elevation. Key in

```
000  58 FIX   016  55  ÷    032  02  02
001  00  00   017  05  5    033  95  =
002  76 LBL   018  00  0    034  65  ×
003  11  A    019  00  0    035  01  1
004  42 STO   020  95  =    036  01  1
005  00  00   021  75  -    037  05  5
006  91 R/S   022  01  1    038  95  =
007  76 LBL   023  05  5    039  85  +
008  12  B    024  95  =    040  43 RCL
009  42 STO   025  94 +/-   041  00  00
010  01  01   026  42 STO   042  95  =
011  91 R/S   027  02  02   043  42 STO
012  76 LBL   028  43 RCL   044  04  04
013  13  C    029  01  01   045  91 R/S
014  43 RCL   030  75  -    046  00  0
015  00  00   031  43 RCL
```

Fig. 9-31. The short version, for the TI-59.

7,000 followed by **R/S.** The display now asks *TEMP. F. ?:* key in **78, R/S.** After a moment the display reads *9,747.*

Here is what happened in the computer: Steps 01 through 27 place the input data into the appropriate registers and convert degrees F. into degrees C. (The program can also accept an input of degrees C., in which case it ignores steps 17 through 22.) The actual calculation begins at step 28, LBL 02. In step 29, 288.15 is stored in R_{03} for convenient use later on. The elevation (R_{01}) is recalled and multiplied by .001981 and from the result the contents of R_{03} are deducted. Since this has produced a negative result, step 36, CHS, changes it to a positive figure which is then stored in R_{05}.

The contents of R_{05} are divided by the contents of R_{03} and the result stored in R_{06}. By now we are at step 41, where 273.15 is added to the temperature figure in R_{02} and the result is stored in R_{07}. The contents of R_{07} are then divided by the contents of R_{03} and the result is stored in R_{08}. Steps 48 through 51 raise the contents of R_{06} to the power of 5.2563, after which the result is divided by the contents of R_{08} and that result is stored in R_{09}. Steps 54 through 58 raise the contents of R_{09} to the power of .235 and store the result in R_{10}. The contents of R_{10} are then deducted from 1 and the result is stored in R_{11}. Those contents are now multiplied by 145426 and what results is the density altitude we're looking for.

This program is very scientific and will work for all elevations and altitudes up to 40,000 or 50,000 feet. It is also quite long. Once one time, when I was working on a program to determine the takeoff distance required by an aircraft under given weather conditions, I needed to include a density-altitude program but simply didn't have room for all those steps. I

```
01*LBL "DA2"      12 RCL 00      23 *
02 "DENSITY ALT"  13 500         24 RCL 00
03 AVIEW          14 /           25 +
04 PSE            15 15          26 STO 04
05 FIX 0          16 -           27 "FEET="
06 "ELEVATION"    17 CHS         28 ARCL 04
07 PROMPT         18 STO 02      29 AVIEW
08 STO 00         19 RCL 01      30 STOP
09 "TEMP. C."     20 RCL 02      31 .END.
10 PROMPT         21 -
11 STO 01         22 115
```

Fig. 9-32. The same short version of the Density Altitude program for the HP-41C.

decided there must be a simpler way to arrive at an answer close enough to the correct one to be used for the purpose of determining takeoff distances, where 30 or 40 feet of density altitude one way or the other won't make any difference.

I wrote a formula which is amazingly simple and at altitudes or elevations below 15,000 feet it has never erred more than 50 feet; where it did err it always did so on the safe side. Figures 9-31 and 9-32 show how it works. The formula I came up with is: Divide the elevation by 500, deduct 15 and change the sign. Deduct the result from the temperature, multiply by 115 and add the temperature. I don't really know why it works; it simply does. For instance, using an elevation of 2,000 feet and a temperature of 20 degrees C., the scientific program comes up with a density altitude of 3,043 feet, while the short version comes up with 3,035 feet, a difference which is meaningless under most conditions. But caution—the higher the elevation or pressure altitude, the greater the error becomes.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter elevation (pressure altitude)
2. Press **A**
3. Enter temperature; in centigrade or Fahrenheit
4. If °C., press **B** if °F, press **2nd A'**
5. Press **C**

Display: The density altitude

HP-41C

Cards: 1
Passes: 1 short; 2 long
Size: 013
Display: *PRESS ALT*
1. Enter elevation or pressure altitude; **R/S**

Display: *TEMP.F.?*
2. Enter temperature in °F. or ignore request and press **R/S**
Display: If previous step was ignored, *TEMP. C.?* Otherwise it goes directly to the next display
3. Press **R/S**
Display: *D.ALT*= and density altitude

PROPELLER TIP SPEED

Sometimes it is important to know the speed with which some type of extension, attached to a rotating shaft moves through the air at its farthest point, or at any intermediate point. A perfect example is the propeller of an aircraft which must turn at a certain speed to provide the necessary thrust, but it must not rotate so fast that the tips move through the air at supersonic speed.

In the program, the computer must be told the diameter of the propeller or, for that matter, the distance from the central rotating shaft at any point for which speed figures are desired. If this data is keyed in the form of inches, the computer will respond with miles per hour (statute miles, to be exact). It can easily be adjusted to use centimeters and kilometers, centimeters and statute or nautical miles, inches and nautical miles or any other combination by simply changing the sequence of steps which represent the relationship between the input measure and output measure.

Steps 16 through 23 in the HP program and steps 14 through 42 in the TI program store the number of inches which the propeller tip travels in one minute. That is divided by 12 to convert that figure into feet. The result is multiplied by 60 to calculate feet per hour, and that result is divided by 5280, the number of feet in a mile, calculating the figure in terms of miles per hour.

If you want the tip speed in terms of knots (nautical miles) rather than statute miles, change 5280 to 6076, the number of feet in a nautical mile. If, on the other hand, you'd want to use the metric system, the initial figure would represent the number of centimeters which the propeller tip travels in one minute. This should then be multiplied by 60 to arrive at the number of centimeters per hour, after which it should be divided by 10,000 the number of centimeters in a kilometer, and the resulting figure would be the speed in kilometers per mile.

You will notice that in the final steps (49 through 67 in the TI program and 40, 41, and 42 in the HP program) the result is compared to 600 which is the approximate speed of sound (depending on altitude and atmospheric conditions).

In the TI (Fig. 9-33), if the speed is greater than 600 mph, the result will be displayed in a flickering manner for a second, after which the computer shows a violently flashing string of 9s across the display. (one (or any number) divided by 0 = will result in a flashing 9.9999999 99 in the display.) Pressing **R/S** will return the miles-per-hour result into the display for further study.

In the HP (Fig. 9-34), if the result exceeds 600 mph, the computer will first display the words *TOO HIGH RPM.* Pressing **R/S** will produce the result in miles per hour so the operator can see exactly how far beyond mph the speed has gone.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter propeller diameter
2. Press **A**
3. Enter rpm
4. Press **B**
5. Press **C**

Display: Speed of propeller tips at mph. If in excess of 600 mph, speed is displayed in a flicker, followed by a flashing *99.999999999*

HP-41C

Cards: 1
Passes: 2

```
000  47 CMS   024  43 RCL  048  44 SUM
001  25 CLR   025  01  01  049  43 RCL
002  76 LBL   026  95  =   050  02  02
003  11  A    027  55  ÷   051  66 PAU
004  58 FIX   028  01  1   052  66 PAU
005  00  00   029  02  2   053  66 PAU
006  42 STO   030  95  =   054  66 PAU
007  00  00   031  65  ×   055  66 PAU
008  91 R/S   032  06  6   056  66 PAU
009  76 LBL   033  00  0   057  58 FIX
010  12  B    034  95  =   058  09  09
011  42 STO   035  55  ÷   059  01  1
012  01  01   036  05  5   060  55  ÷
013  91 R/S   037  02  2   061  00  0
014  76 LBL   038  08  8   062  95  =
015  13  C    039  00  0   063  91 R/S
016  58 FIX   040  95  =   064  76 LBL
017  01  01   041  42 STO  065  44 SUM
018  43 RCL   042  02  02  066  43 RCL
019  00  00   043  32 X:T  067  02  02
020  65  ×    044  06  6   068  91 R/S
021  89  π    045  00  0   069  00  0
022  95  =    046  00  0
023  65  ×    047  77  GE
```

Fig. 9-33. The TI-59 version of Propeller Tip Speed.

```
 01↑LBL "PROP"      16 STO 02          31 GTO 00
 02 "PROPELLER"     17 12              32 GTO 01
    03 AVIEW        18 /               33↑LBL 00
    04 PSE          19 60              34 FIX 1
05 "DIAMETER, IN."  20 *               35 "MPH="
    06 PROMPT       21 5280            36 ARCL 03
    07 STO 00       22 /               37 AVIEW
    08 " RPM ?"     23 STO 03          38 STOP
    09 PROMPT    24 "TIP SPEED"        39↑LBL 01
    10 STO 01       25 AVIEW        40 "TOO HIGH RPM"
    11 RCL 00       26 PSE             41 AVIEW
    12 PI           27 RCL 03          42 STOP
    13 *            28 ENTER↑          43 GTO 00
    14 STO 01       29 600             44 END
    15 *            30 X>Y?
```

Fig. 9-34. Propeller Tip Speed on the HP-41C.

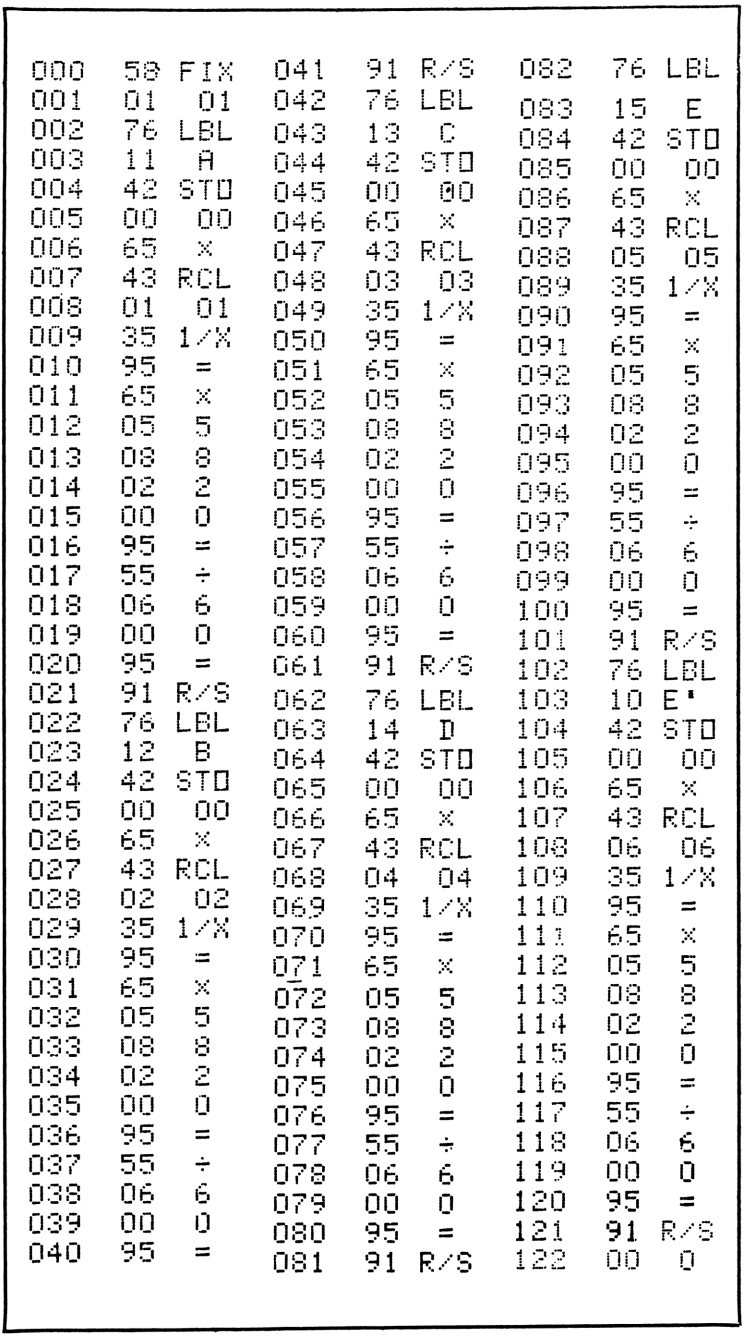
HP-41C

Size: 004
Display: *DIAMETER, IN?*
1). Enter diameter; **R/S**
Display: *RPM?*
2). Enter rpm; **R/S**
Display: *TIP SPEED,* then *MPH=* and speed
If speed is in excess of 600 mph:
Display: *TOO HIGH RPM*
3). Press **R/S**
Display: *MPH=* and speed

SCALE SPEED

This program is designed for model railroaders or others operating models of any type of vehicle other than airplanes. It converts prototype miles per hour into feet per minute, using the six most popular model-railroad scales (O gauge, 48:1; S gauge 64:1; 00 gauge 76.2:1; HO gauge, 87.1:1; TT gauge, 120:1; and N gauge, 160:1).

The programs, Figs. 9-35 and 9-36 actually consist of six subprograms, one for each of the scales involved, plus a section which contains the actual calculation. In addition, the program for the HP uses steps 02 through 13 to store the scales for the six gauges in the appropriate registers, while in the TI these data are recorded on a separate data card. Furthermore, the HP program includes a section labeled **LBL Q** that consists of steps 18 through 48, and is used by the operator to tell the computer which gauge he is interested in. This process of selection is accomplished in the TI by assigning each gauge to a separate key, 0 to key **A,** S to key **B** and so on.



```
000  50 FIX   041  91 R/S   082  76 LBL
001  01  01   042  76 LBL   083  15  E
002  76 LBL   043  13  C     084  42 STO
003  11  A    044  42 STO   085  00  00
004  42 STO   045  00  00    086  65  ×
005  00  00   046  65  ×     087  43 RCL
006  65  ×    047  43 RCL   088  05  05
007  43 RCL   048  03  03    089  35 1/X
008  01  01   049  35 1/X   090  95  =
009  35 1/X   050  95  =     091  65  ×
010  95  =    051  65  ×     092  05  5
011  65  ×    052  05  5     093  08  8
012  05  5    053  08  8     094  02  2
013  08  8    054  02  2     095  00  0
014  02  2    055  00  0     096  95  =
015  00  0    056  95  =     097  55  ÷
016  95  =    057  55  ÷     098  06  6
017  55  ÷    058  06  6     099  00  0
018  06  6    059  00  0     100  95  =
019  00  0    060  95  =     101  91 R/S
020  95  =    061  91 R/S   102  76 LBL
021  91 R/S   062  76 LBL   103  10 E'
022  76 LBL   063  14  D     104  42 STO
023  12  B    064  42 STO   105  00  00
024  42 STO   065  00  00    106  65  ×
025  00  00   066  65  ×     107  43 RCL
026  65  ×    067  43 RCL   108  06  06
027  43 RCL   068  04  04    109  35 1/X
028  02  02   069  35 1/X   110  95  =
029  35 1/X   070  95  =     111  65  ×
030  95  =    071  65  ×     112  05  5
031  65  ×    072  05  5     113  08  8
032  05  5    073  08  8     114  02  2
033  08  8    074  02  2     115  00  0
034  02  2    075  00  0     116  95  =
035  00  0    076  95  =     117  55  ÷
036  95  =    077  55  ÷     118  06  6
037  55  ÷    078  06  6     119  00  0
038  06  6    079  00  0     120  95  =
039  00  0    080  95  =     121  91 R/S
040  95  =    081  91 R/S   122  00  0
```

Fig. 9-35. The model railroad speed conversion program for the TI-59.

Anyone wanting to use this program with scales other than those included here would have to enter the new scale into one of the memory registers now used to store the scale data (R_{01} through R_{06}) because the difference in scales in no way affects the calculations.

TI-59. The magnetic card for this program (Fig. 9-35) involves one pass of side 1 which contains the program and a pass of the opposite side which contains the scale data. The first pass should leave a steady *1* in the display and the second pass should produce a steady *4*. Now enter the prototype speed, **45,** and if we now press **A** we find that 0-gauge train traveling at 45 scale miles will cover 90.9 feet in one minute. Pressing **B** produces 68.2 feet per minute for S gauge. Pressing **C** results in 57.3 fpm for 00 gauge. Key **D** shows 50.1 fpm for HO gauge. Key **E** shows 36.4 fpm for TT gauge and pressing **2nd E′** results in 27.3 fpm for N gauge.

HP-41C. The display identifies the program (Fig. 9-36) *SCALE SPEED* and then asks for *PROTO. MPH?* we enter the prototype speed which we are interested in, say **45** mph, and press **R/S**. The display responds with *GAUGE* and then changes to *0? 0=YES.* Being an N-gauger myself, I'll ignore this and press **R/S**. The display now asks the same question in succession for each gauge until it comes to *N? 0=YES.* I press **0** and **R/S** and the computer tells me *FT/MIN= 27.3,* meaning an N-gauge train, covering 27.3 feet of track in one minute, is traveling at a scale speed of 45 mph.

Pressing **R/S** again produces *AGAIN? 0= NO* because one might want to get the results for a whole series of prototype speeds.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 2
Passes: 1 program, 1 data	Passes: 2 program, 1 data
Partition: 479 59	Size: 007
1. Enter prototype miles per hour figure	Display: *PROTO.MPH?*
2. Press key for the scale desired:	1. Enter prototype speed in miles per hour; **R/S**
A=0 gauge	Display: *GAUGE?* then
B=S gauge	*0? 0=YES; S? 0=YES; 00? 0=YES;*
C=00 gauge	*HO? 0=YES; TT? 0=YES; N? 0=YES.*
D=HO gauge	2. Press **O** and **R/S** when the desired gauge is in the display.
E=TT gauge	Display: *FT/MIN=* number of feet per minute
E′=N gauge	3. Press **R/S**
Display: Feet per minute of travel to represent the prototype speed	Display: *AGAIN? 0= NO*
3. After the above, enter speed in feet per minute	
4. Press **R/S**	
Display: Equivalent prototype speed in mph	

```
01♦LBL "MRR"
 02 48
 03 STO 01
 04 64
 05 STO 02
 06 76.2
 07 STO 03
 08 87.1
 09 STO 04
 10 120
 11 STO 05
 12 160
 13 STO 06
14 "SCALE SPEED"
 15 AVIEW
 16 PSE
 17 FIX 1
 18♦LBL "Q"
19 "PROTO. MPH ?"
 20 PROMPT
 21 STO 00
 22 "GAUGE:"
 23 AVIEW
 24 PSE
 25 "Q? 0=YES'
 26 PROMPT
 27 X=0?
 28 GTO "Q"
 29 "S? 0=YES"
 30 PROMPT
 31 X=0?
 32 GTO "S"
 33 "OO? 0=YES"
 34 PROMPT
 35 X=0?
 36 GTO "OO"
37 "HO? 0=YES"
 38 PROMPT
 39 X=0?
 40 GTO "HO"
41 "TT? 0=YES"
 42 PROMPT
 43 X=0?
 44 GTO "TT"
 45 "N? 0=YES"
 46 PROMPT
 47 X=0?
 48 GTO "N"

49♦LBL "Q"
 50 RCL 00
 51 RCL 01
 52♦LBL "U"
 53 1/X
 54 *
 55 5820
 56 *
 57 60
 58 /
 59 STO 00
 60 "FT/MIN="
 61 ARCL 00
 62 AVIEW
 63 STOP
 64 GTO "T"

65♦LBL "S"
 66 RCL 00
 67 RCL 02
 68 GTO "U"

69♦LBL "OO"
 70 RCL 00
 71 RCL 03
 72 GTO "U"

73♦LBL "HO"
 74 RCL 00
 75 RCL 04
 76 GTO "U"

77♦LBL "TT"
 78 RCL 00
 79 RCL 05
 80 GTO "U"

81♦LBL "N"
 82 RCL 00
 83 RCL 06
 84 GTO "U"

 85♦LBL "T"
 86 "AGAIN 0=NO"
 87 PROMPT
 88 X=0?
 89 GTO "R"
 90 GTO "Q"
 91♦LBL "R"
 92 "END"
 93 AVIEW
 94 STOP
 95 .END.
```

Fig. 9-36. This program converts prototype miles per hour to scale-model feet per minute for the HP-41C.

SPEED, DISTANCE, AND THE VALUE OF TIME

The cost of fuel consumption increases with an increase in speed, no matter whether we're traveling in an automobile or airplane. But increases in speed reduce the time spent en route, and if we consider that our time is

worth a given amount per hour, that saving in time, multiplied by what we consider ourselves worth, will usually prove to be worth more than the increase in the cost of fuel.

To function properly, the program needs quite a bit of input data. Let's simply run it and explain what happens along the way.

TI-59. With the TI it is important to slide the program card (Fig. 9-37), after it has been recorded, into the upper slot to remind us which data should be entered in conjunction with which of the 10 user-defined keys (A through E and A′ through E′).

Start by entering the distance in question, **500,** followed by pressing **A.** Then enter the first speed to be examined, **55,** and press **B.** Now press **R/S** and the display reads *9.0527.* Enter the low and high speeds for which fuel-consumption figures have been established, **30 C, 65 R/S** and then key in the miles per gallon figures for those speeds, **38 R/S, 26 R/S.** By pressing **D** the display responds with the fuel consumption figure at 55 mph, *29.7.* Now enter our own worth per hour, **25** and press **E,** followed by the price of gas, **1.369** and press **2nd E′.** Now press **2nd D′** and the display shows *227.08* as the cost of time at $25 per hour. Pressing **2nd C′** displays *23.04* as the price of gas, and pressing **2nd B′** results in *250.12* as the total cost of the trip.

To compare figures, enter **45** and press **B** and then **R/S** to learn that at that speed the trip will take *11.0640* (11 hours, six minutes and 40 seconds). Pressing **2nd D′** changes the cost of time to *277.78:* **2nd C′** reduces the gas cost to *$20.48* and **2nd B′** shows the total cost of the trip as *$298.26.*

Using the 80 mph figure, press **80** and **B** and **R/S** displays trip time as *6.1500;* the cost of time as *$156.25;* the cost of fuel as *$32.04* and the total cost for the trip as *$188.29.*

HP-41C: The computer identifies the program (Fig. 9-38) by displaying *SPEED/TIME* and then asks for *DISTANCE?* Key in **500** miles and see what happens. **R/S** asks for *MILES/HOUR.* Being basically law-abiding, we'll enter **55,** then press **R/S.** The display responds with *TIME=* and changes to *H.MMSS 9.0527,* representing nine hours, five minutes and 27 seconds. So far this has been straightforward. We've simply divided the total distance by the mph figure.

Now press **R/S** and the display asks *GAS MILEAGE AT LOW SPEED.* What the computer wants to know here is the lowest speed for which a legitimate fuel-consumption figure is available. Let's enter **30.** Press **R/S** and the display asks *AT HI SPEED,* meaning the highest speed for which such figures are known. Let's enter **65. R/S** now results in *MPG AT LOW S.,* the known fuel-consumption figure at 30 mph. Let's enter **38** and press **R/S.** Now it asks *MPG AT HI S.* The comparable figure at 65, and we'll assume that to be **26. R/S** results in *MPG = 29.4,* the miles per gallon which can be expected at 55 mph.

Pressing **R/S** causes the display to ask *TIME/HR. $?,* meaning the computer wants to know what you believe yourself to be worth in terms of dollars per hour. Let's put in **25;** pressing **R/S** will tell you that the time

000	58	FIX	053	14	D	106	91	R/S
001	00	00	054	43	RCL	107	76	LBL
002	47	CMS	055	02	02	108	10	E'
003	25	CLR	056	75	-	109	42	STO
004	76	LBL	057	43	RCL	110	07	07
005	11	A	058	03	03	111	91	R/S
006	42	STO	059	95	=	112	76	LBL
007	00	00	060	42	STO	113	19	D'
008	91	R/S	061	09	09	114	43	RCL
009	76	LBL	062	43	RCL	115	06	06
010	12	B	063	04	04	116	65	×
011	42	STO	064	75	-	117	43	RCL
012	01	01	065	43	RCL	118	10	10
013	91	R/S	066	05	05	119	88	DMS
014	58	FIX	067	95	=	120	95	=
015	04	04	068	42	STO	121	42	STO
016	43	RCL	069	20	20	122	15	15
017	01	01	070	55	÷	123	91	R/S
018	35	1/X	071	43	RCL	124	76	LBL
019	65	×	072	09	09	125	18	C'
020	43	RCL	073	95	=	126	58	FIX
021	00	00	074	42	STO	127	02	02
022	95	=	075	11	11	128	43	RCL
023	42	STO	076	43	RCL	129	00	00
024	11	11	077	01	01	130	55	÷
025	22	INV	078	75	-	131	43	RCL
026	88	DMS	079	43	RCL	132	14	14
027	42	STO	080	03	03	133	95	=
028	10	10	081	95	=	134	65	×
029	91	R/S	082	42	STO	135	43	RCL
030	76	LBL	083	13	13	136	07	07
031	13	C	084	65	×	137	95	=
032	58	FIX	085	43	RCL	138	42	STO
033	00	00	086	11	11	139	17	17
034	42	STO	087	95	=	140	91	R/S
035	02	02	088	42	STO	141	76	LBL
036	91	R/S	089	14	14	142	17	B'
037	76	LBL	090	75	-	143	43	RCL
038	13	C	091	43	RCL	144	17	17
039	42	STO	092	04	04	145	85	+
040	03	03	093	95	=	146	43	RCL
041	91	R/S	094	94	+/-	147	15	15
042	76	LBL	095	42	STO	148	95	=
043	13	C	096	14	14	149	42	STO
044	42	STO	097	58	FIX	150	19	19
045	04	04	098	01	01	151	91	R/S
046	91	R/S	099	91	R/S	152	00	0
047	76	LBL	100	76	LBL	153	00	0
048	13	C	101	15	E	154	00	0
049	42	STO	102	58	FIX	155	00	0
050	05	05	103	02	02	156	00	0
051	91	R/S	104	42	STO			
052	76	LBL	105	06	06			

Fig. 9-37. The Speed, Distance and the Value of Time program for the TI-59.

```
01♦LBL "ST"
02 CF 00
03 CF 01
04 CF 02
05 "SPEED/TIME"
06 AVIEW
07 PSE
08 FIX 4
09 "DISTANCE ?"
10 PROMPT
11 STO 15

12♦LBL 00
13 "MILES/HOUR ?"
14 PROMPT
15 STO 16
16 1/X
17 RCL 15
18 *
19 HMS
20 STO 01
21 "TIME="
22 AVIEW
23 PSE
24 FIX 4
25 "H.MMSS="
26 ARCL 01
27 AVIEW
28 STOP
29 FIX 2
30 FS? 00
31 GTO 01
32 "GAS MILAGE"
33 AVIEW
34 PSE
35 "AT LO SPEED"
36 PROMPT
37 STO 05
38 "AT HI SPEED"
39 PROMPT
40 STO 06
41 "MPG AT LOW S."
42 PROMPT
43 STO 07
44 "MPG AT HI S."
45 PROMPT
46 STO 08
47 SF 00
48♦LBL 01
49 RCL 06
50 RCL 05
51 -
52 STO 09
53 RCL 07
54 RCL 08
55 -
56 STO 10
57 RCL 09
58 /
59 STO 11
60 RCL 16
61 STO 12
62 RCL 05
63 -
64 STO 13
65 RCL 11
66 *
67 STO 14
68 RCL 07
69 -
70 CHS
71 STO 04
72 FIX 1
73 "MPG="
74 ARCL 04
75 AVIEW
76 STOP
77 FS? 01
78 GTO 02
79 "TIME/HR. $ ?"
80 PROMPT
81 STO 18
82 SF 01
83♦LBL 02
84 RCL 18
85 RCL 01
86 HR
87 *
88 STO 03
89 FIX 2
90 "TIME $="
91 ARCL 03
92 AVIEW
93 STOP
94 FS? 02
95 GTO 03
96 "GAS $ GAL ?"
97 PROMPT
98 STO 02
99 SF 02
100♦LBL 03
101 RCL 02
102 RCL 15
103 RCL 04
104 /
105 RCL 02
106 *
107 STO 17
108 "GAS $="
109 ARCL 17
110 AVIEW
111 STOP
112 RCL 03
113 +
114 STO 03
115 "= $"
116 ARCL 03
117 AVIEW
118 STOP
119 GTO 00
120 .END.
```

Fig. 9-38. The Speed, Distance and the Value of Time program on the HP-41C.

spent covering those 500 miles amounts to $227.27. The display shows *TIME $=227.27.* **R/S** now produces the request *GAS $ GAL?* or the cost of gas, on the average, per gallon. Let's take an average, **$1.349,** and **R/S** tells us the trip will cost *GAS $ = 22.92*. Pressing **R/S** adds those two figures and tells us the trip will actually cost us = *$250.19.*

Since the primary purpose of the program is to compare trip costs at different speeds, pressing **R/S** now results once more in *MILES/HOUR?*. This time we plan to save on the gas so we key in **45.** The result is that the time-en-route display shows 11.0640. The miles-per-gallon display shows 32.9 and the cost at $25 per hour for the trip has risen to $277.78. By contrast, the cost of gas has dropped to $20.53 but the overall cost of the trip has risen to $298.31.

Since that is obviously poor financing, let's go to the other extreme and when the computer once more asks for *MILES/HOUR?* let's enter 80 (maybe this is Europe where you can legally drive that fast). The resulting time display now reads *H.MMSS 6.1500.* The miles per gallon have dropped to *MPG= 20.9*; the cost-of-time total has dropped to *TIME $=156.25*. The gas costs have risen to *GAS $= 32.34*, but the total cost for the trip has come down to *$188.59.*

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

1. Enter distance to be traveled
2. Press **A**
3. Enter miles per hour
4. Press **B**
5. Press **R/S**

Display: Time to cover that distance in hours, minutes and seconds

6. Enter lowest speed with known fuel consumption
7. Press **C**
8. Enter highest speed with known fuel consumption
9. Press **R/S**
10. Enter mpg for low speed
11. Press **R/S**
12. Enter mpg for hi speed
13. Press **R/S**
14. Press **D**

Display: mpg figure for speed entered in step 3

15. Enter what you consider your time to be worth per hour
16. Press **E**
17. Enter the price per gallon of gas
18. Press **2nd E′**
19. Press **2nd D′**

Display: The cost of our time for the length of the trip

20. Press **2nd C′**

Display: Price of gas used for the trip

21. Press **2nd B′**

Display: Total cost of the trip including gas and our value for the time spent en route

22. Enter other speeds to be used for the trip
23. Press **B**, then **R/S**
24. Press **2nd D′**, **C′** and **B′**

Displays: Cost of time; cost of gas; total cost at new speed

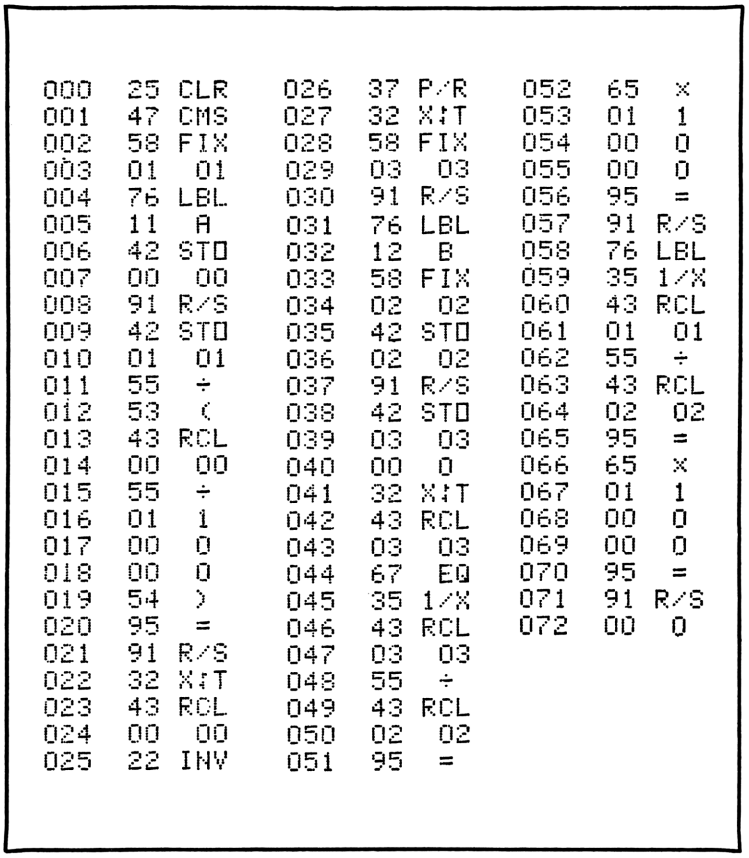
Cards: 2
Passes: 3
Size: 019
Display: *DISTANCE?*
1. Enter distance to be covered; **R/S**
Display: *MILES/HOUR?*
2. Enter speed to be used; **R/S**
Display: then *TIME*=.
H.MMSS = and the time en route in hours, minutes and seconds
3. Press **R/S**
Display: *GAS MILEAGE,* then *AT LO SPEED?*
4. Enter low speed for which mpg is known; **R/S**
Display: *AT HI SPEED?*
5. Enter high speed for which mpg is known; **R/S**
Display: *MPG AT LOW S.?*
6. Enter known mpg for low speed; **R/S**
Display: *MPG AT HI S.?*
7. Enter mpg for high speed; **R/S**
Display: *MPG*= and gas mileage for speed entered in step 2
8. Press **R/S**
Display: *TIME/HR, $?*
9. Enter what your time is worth per hour; **R/S**
Display: *TIME* = *$* and the worth of your time during the trip
10. Press **R/S**
Display: *GAS $ GAL ?*
11. Enter price of gas per gal.; **R/S**
Display: *GAS*= *$* and the cost of gas for the trip
12. Press **R/S**
Display: = *$* and the combined total
13. Press **R/S**
Display: *MILES/HOUR?*
14. Enter different speed to be used; **R/S**
Display: *TIME*=, then *H.MMSS* = and new time en route
15. Press **R/S**
Display: *MPG*= new mpg figure
16. Press **R/S**
Display: *TIME* = *$* new time-cost
17. Press **R/S**
Display: *GAS*= *$* new gas cost
18. Press **R/S**
Display: = *$* new total cost

PERCENT GRADE

This program was originally meant to help me construct a model-railroad layout, where it is always important to know how much track is needed to accomplish the increase in elevation required to have one track pass over or under another. It has a great many other possible applications.

The program consists of two parts. In part one we feed in the percent grade which we are interested in or willing to accept, and we then feed in the amount of altitude change which must be accomplished. The computer responds by providing first the required horizontal distance, followed by the slant distance. Whatever measure is used in the altitude-change data will also be used displaying the lengths. Thus, if the change in elevation is given in inches, the length will be in inches and so on.

The second half of the program accomplishes the reverse. We feed in the length and the required change in elevation and the computer will tell us



```
000  25 CLR    026  37 P/R    052  65  ×
001  47 CMS    027  32 X:T    053  01  1
002  58 FIX    028  58 FIX    054  00  0
003  01  01    029  03  03    055  00  0
004  76 LBL    030  91 R/S    056  95  =
005  11  A     031  76 LBL    057  91 R/S
006  42 STO    032  12  B     058  76 LBL
007  00  00    033  58 FIX    059  35 1/X
008  91 R/S    034  02  02    060  43 RCL
009  42 STO    035  42 STO    061  01  01
010  01  01    036  02  02    062  55  ÷
011  55  ÷     037  91 R/S    063  43 RCL
012  53  (     038  42 STO    064  02  02
013  43 RCL    039  03  03    065  95  =
014  00  00    040  00  0     066  65  ×
015  55  ÷     041  32 X:T    067  01  1
016  01  1     042  43 RCL    068  00  0
017  00  0     043  03  03    069  00  0
018  00  0     044  67  EQ     070  95  =
019  54  )     045  35 1/X    071  91 R/S
020  95  =     046  43 RCL    072  00  0
021  91 R/S    047  03  03
022  32 X:T    048  55  ÷
023  43 RCL    049  43 RCL
024  00  00    050  02  02
025  22 INV    051  95  =
```

Fig. 9-39. The TI-59 version.

what percent grade will result. Figure 9-39 and 9-40 are the programs. Let's quickly walk through it to see what happens.

TI-59. Here the program, (Fig. 9-39) is split between two of the user-defined keys. Key **A** controls the first part of the program while key **B** controls the second part.

Feed in the desired percent grade, **2.3,** and press **A.** Then enter the change in elevation, **1.9** and press **R/S.** The display responds with *82.61.* Press **R/S** again and the display changes to *82.661,* representing the slant distance while the first figure was the horizontal distance.

To use the second half of the program, key in **82.6** and press **B.** Then key in the change in elevation, **1.9,** again, and press **R/S.** The display shows *2.30* as the required percent grade to accomplish that change in elevation in the stated distance.

HP-41C. After displaying *GRADE* the display asks *PERCENT?.* Feed in a percent figure we want to examine, say **2.3,** and press **R/S.** The

```
01↑LBL "GRADE"
   02 FIX 1
   03 "GRADE"
   04 AVIEW
   05 PSE

   06↑LBL G
07 "PERCENT ?"
   08 PROMPT
   09 STO 00
   10 X=0?
   11 GTO "G3"
   12 100
   13 /
   14 STO 00
15 "ALT. CHANGE?"
   16 PROMPT
   17 STO 01
   18 RCL 00
   19 /
   20 STO 02
  21 "LENGTH="
   22 ARCL 02
   23 AVIEW
   24 STOP
   25 RCL 01
   26 ENTER↑
   27 RCL 02
   28 R-P
   29 STO 03
30 "LENGTH↑="
   31 ARCL 03
   32 AVIEW
   33 STOP
34 "AGAIN? 0=NO"
   35 PROMPT
   36 X=0?
   37 GTO "G1"
   38 GTO G

   39↑LBL "G3"
40 "LENGTH ?"
   41 PROMPT
   42 STO 02
43 "ALT. CHANGE?"
   44 PROMPT
   45 STO 01
   46 RCL 02
   47 /
   48 100
   49 *
   50 FIX 1
   51 STO 04
52 "PERCENT="
   53 ARCL 04
   54 AVIEW
   55 STOP
56 "AGAIN? 0=NO"
   57 PROMPT
   58 X=0?
   59 GTO "G1"
   60 GTO "G3"

   61↑LBL "G1"
   62 "END"
   63 AVIEW
   64 STOP
   65 .END.
```

Fig. 9-40. This program for the HP-41C determines the required grade to change elevations in a given distance, or vice versa.

second part). Pressing that key produces a display reading *LENGTH?* Put in the horizontal distance to accomplish the required change in elevation. display now requests the *ALT. CHANGE.* Put in **1.9** (the distance in inches is required to have one track pass over another in N-gauge model railroading) and press **R/S**. The display now tells us it will require 82.6 inches of horizontal distance to accomplish that change in elevation at the 2.3 percent grade. The display reads *LENGTH = 82.6.* Now press **R/S** and the display changes to *LENGTH = 82.7,* meaning there is a tiny difference in length between the horizontal distance and the slant distance in this example.

Pressing **R/S** again produces *AGAIN? 0=NO*, meaning that if we simply press **R/S** the computer will go back to the beginning and again ask for percent data. That way it is easy to run a number of different combinations to find the one that works best under given circumstances. If, instead, we press **0** and **R/S** the computer will simply display *END.*

To get to the second part of the program, I assigned G3 to a key (the R↓ key starts the first part of the program and the shift R↓ key starts the

Key in **82.6** and **R/S**. The display asks for the *ALT CHANGE*. Key in **1.9** as before. **R/S** now results in *PERCENT= 2.3*. After that the *AGAIN? 0=NO* routine is repeated, only this time, if you do want to repeat, it returns to the beginning of the second part of the program.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 005
1. Enter the desired percent grade	Display: *PERCENT?*
2. Press **A**	1. Enter desired % grade, **R/S**
3. Enter the desired change in elevation or altitude	Display: *ALT.CHANGE?*
4. Press **R/S**	2. Enter the change in elevation or altitude desired, **R/S**
Display: The distance required measured horizontally	Display: *LENGTH*=horizontal distance
5. Press **R/S**	3. Press **R/S**
Display: The needed distance measured on the slant	Display: *LENGTH* = slant distance; **R/S**
	Display: *AGAIN? 0=NO*
	4. Press either **O** and **R/S** or simply **R/S**

Chapter 10
Programs with Practical Uses

This section continues programs you should find a use for in every day life.

CHECKBOOK

This program accepts the operator's word for his last checkbook balance, totals all current monthly deposits and adds them to the balance, totals all withdrawals and deducts them from the balance, and displays the final balance. It will then accept an input of the balance shown on the bank statement and display the total amount represented by checks which have not, as yet, cleared through the bank.

The arithmetic involved is straight-forward, but a problem arises involving the need to be able to enter a large number of data representing deposits and withdrawals, in succession, and to have the computer accept them and add them to one another. This problem shows up from time to time when we want to write a personal program, and it is worth while to look at the way it was solved here. There are several ways in which this problem can be handled. The one chosen here seems to be the easiest to understand.

TI-59. When the program is written for the TI-59 (Fig. 10-1) it uses the A through E keys to identify different parts of the program. The sections which permit entry of an unlimited number of deposits or withdrawals consist of steps 009 through 021 and 022 through 34. The computer accepts the entry and places it into R_{01} or R_{03}. It then adds that amount to whatever is stored in R_{02} or R_{04}. (It is important here that each time the program is run, it is preceded by keying in **RST R/S**, which ensures that steps 02 and 03 are executed and all data previously stored in any of the memory registers is cleared out. Failing that, we're likely to find new entries of deposits or withdrawals added to left-out data.) The display will show the

gradually increasing total as each new entry is made. To be sure that the two computers agree, we'll use the same figures when running the program.

To run the program first enter the last balance, **3247.67** and press **A**. Then enter the first deposit, **435.11** and press **B**. The display shows that figure. Enter the next deposit, **76.31** and press **B** again. The display now shows the total of the two deposits as *511.42*. Enter the withdrawals in the same manner (listed in the HP section), and after each entry, press **C**. The display will successively show *13.77, 90.11, 189.12* and *231.90*. That concludes the entries of the activity during the period in question.

Now press **D** and the display shows *3527.19* as the new balance. Key in the balance shown on the last statement, **3226.14** and press **E**. That figure will show up in the display. Press **R/S** and the display shows *301.05*, the amount which has not yet cleared through the bank.

HP-41C. In Fig. 10-2 we show the program for the HP. Steps 12 through 19 and 23 through 30 control the input of successive data. The input is first stored in R_{01} (R_{03} in the withdrawal sequence). The computer then

```
000  58 FIX    023  13  C     046  43 RCL
001  02  02    024  42 STO    047  04  04
002  47 CMS    025  03  03    048  95  =
003  25 CLR    026  85  +     049  42 STO
004  76 LBL    027  43 RCL    050  05  05
005  11  A     028  04  04    051  91 R/S
006  42 STO    029  95  =     052  76 LBL
007  00  00    030  42 STO    053  15  E
008  91 R/S    031  04  04    054  42 STO
009  76 LBL    032  91 R/S    055  06  06
010  12  B     033  61 GTO    056  91 R/S
011  42 STO    034  13  C     057  43 RCL
012  01  01    035  76 LBL    058  05  05
013  85  +     036  14  D     059  75  -
014  43 RCL    037  43 RCL    060  43 RCL
015  02  02    038  00  00    061  06  06
016  95  =     039  85  +     062  95  =
017  42 STO    040  43 RCL    063  42 STO
018  02  02    041  02  02    064  06  06
019  91 R/S    042  95  =     065  91 R/S
020  61 GTO    043  42 STO    066  00  0
021  12  B     044  05  05
022  76 LBL    045  75  -
```

Fig. 10-1. The Checkbook program for the TI-59.

```
 01♦LBL "CB"        21 0                41 "$"
 02 CLRG        22 "WITHDRAWELS"        42 ARCL 05
03 "CHECKBOOK"      23♦LBL 04           43 AVIEW
 04 AVIEW           24 PROMPT           44 STOP
 05 PSE             25 STO 03     45 "STATEMENT:BAL."
 06 FIX 2           26 X=0?             46 PROMPT
 07 0               27 GTO 01           47 STO 06
08 "LAST BALANCE"   28 RCL 03           48 RCL 05
 09 PROMPT          29 ST+ 04           49 -
 10 STO 00          30 GTO 04           50 CHS
 11 "DEPOSITS"      31♦LBL 01           51 STO 06
 12♦LBL 03          32 RCL 00         52 "NOT CL.="
 13 PROMPT          33 RCL 02           53 ARCL 06
 14 STO 01          34 +                54 AVIEW
 15 X=0?            35 RCL 04           55 STOP
 16 GTO 00          36 -                56 "END"
 17 RCL 01          37 STO 05           57 AVIEW
 18 ST+ 02      38 "NEW BALANCE"        58 STOP
 19 GTO 03          39 AVIEW            59 END
 20♦LBL 00          40 PSE
```

Fig. 10-2. Checkbook as designed for the HP-41C.

asks if that last entry was a zero, which would indicate no further entries are to be made. If it is zero, the computer will execute the next step and go to the section **LBL 00** (or **LBL 01**). If it is not zero, if an entry of a deposit (or withdrawal) was made, the computer will ignore that step and go to the one beyond. Here it recalls the previously stored data and adds it to data already stored in R_{02}. The fact that there is nothing in R_{02} when the first entry is made does not matter; the first entry is simply added to zero. The computer is then told to return to section **LBL 03** (or **LBL 04**) which is, in fact, the section it has just executed. In this manner it is possible to enter either one or an unlimited number of deposits or withdrawal data.

When all deposits or withdrawals have been entered, press **0** and **R/S** and the computer will go on to the next portion of the program. Here is how a run of the program turns out:

The display identifies the program with *CHECKBOOK* and then asks for *LAST BALANCE*. We key in **3247.67** (or whatever) and the input appears in the display. Press **R/S** and the display asks *DEPOSITS*. Enter **435.11**, and when we press **R/S**, the display again asks *DEPOSITS*. Enter **76.31** and press **R/S** again. Assuming that these are the only deposits made during the period in question, now respond to the request *DEPOSITS* with **0** and **R/S**.

The display now asks for *WITHDRAWALS*; key in one after the other in the same manner. Let's simply say that the only checks written were **13.77, 76.34, 99.01** and **42.78**, after which press **0** and **R/S** and the display first reads *NEW BALANCE* and then changes to *$3,527.19*, which is the balance after all deposits were added and all withdrawals deducted. Now press **R/S** and the display asks *STATEMENT:BAL*. Here key in the balance shown on the last bank statement, say, **3226.14** and **R/S**, and the display responds with *NOT CL.=301.05*, the total amount for which checks have been written which have not yet cleared through the bank. Pressing **R/S** once more produces *END*.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 007
1. Key in last balance	Display: *LAST BALANCE*
2. Press **A**	1. Key in previous balance; **R/S**
3. Key in deposits and after each deposit, press **B.**	Display: *DEPOSITS*
Display: the increasing totals of deposits.	2. Key in deposits. The display will repeat DEPOSITS after each entry. Enter 0 to indicate no more deposits; **R/S**
4. Key in withdrawals, press **C** after each withdrawal	Display: *WITHDRAWALS*
Display: the increasing total of withdrawals	3. Key in withdrawals in the same manner as deposits. **0** for finish; **R/S**
5. Press **D.**	Display: *NEW BALANCE*, then *$=*
Display: new balance	4. Press **R/S**
6. Enter balance according to bank statement	Display: *STATEMENT:BAL.*
7. Press **E**	5. Key in balance on last statement; *R/S*
8. Press **R/S**	Display: *NOT CL.=$* and the total of check that have not yet cleared.
Display: total of checks that have not yet cleared.	

THE COST OF USING ELECTRICITY

Most of us have a foggy idea of what using any given electrical device amounts to in terms of the monthly electricity bill. This program is designed to answer those questions.

The computer must be told the charge per kilowatt hour which is made by the utility company providing electricity in your area. In my case, that charge is 0.0808, as can be seen in the sample run illustrated in Figs. 10-3 and 10-4.

Next it must be told the amount of watt rating of each appliance. We then must enter the approximate amount of use in hours, minutes, and seconds. Here we can simply enter one hour (1) and the result will be the cost of running the appliance for one hour. Or, we can enter the number of hours during which the appliance is turned on during the average day, and enter the number of days during which the appliance is used for that number of hours in one month. The computer will produce two figures, the number of hours during a month during which the appliance is in use, and the cost of that use in terms of dollars and cents.

In the sample run (Figs 10-5 and 10-6) we have assumed a kilowatt-hour cost of 0.0808, an appliance that uses 250 watts and which is used for an average of 5.30 hours a day during 18 days of the month, resulting in a monthly use of 95.40 hours. Converting 95 hours and 40 minutes into the decimal equivalent results in 95.67, multiplied by the watts used by the appliance (250) for a result of 23,917, representing the total number of watts used. That figure is then multiplied by 0.0808 and the result divided by 1000, to produce the cost for the month which, in this instance, comes to $1.93.

We can run the program again with a different appliance (in the case of the HP by simply pressing R/S, and in the case of the TI by starting all over again).

This time enter 100 watts, because we want to know the cost of using a 100-watt bulb for a given time period. Enter a use figure of 10 hours per day

000	58	FIX	018	91	R/S	036	01	01
001	04	04	019	42	STO	037	95	=
002	76	LBL	020	03	03	038	55	÷
003	11	A	021	65	×	039	01	1
004	42	STO	022	43	RCL	040	00	0
005	00	00	023	02	02	041	00	0
006	91	R/S	024	95	=	042	00	0
007	76	LBL	025	42	STO	043	95	=
008	12	B	026	03	03	044	58	FIX
009	58	FIX	027	91	R/S	045	02	02
010	00	00	028	88	DMS	046	42	STO
011	42	STO	029	42	STO	047	03	03
012	01	01	030	03	03	048	91	R/S
013	91	R/S	031	65	×	049	00	0
014	76	LBL	032	43	RCL			
015	13	C	033	00	00			
016	42	STO	034	65	×			
017	02	02	035	43	RCL			

Fig. 10-3. The Cost of Using Electricity for the TI-59.

```
01♦LBL "EL"
02 "ELECTRICITY"
03 AVIEW
04 PSE
05 "COST KWH ?"
06 PROMPT
07 STO 00
08♦LBL 01
09 "APPL. WATT ?"
10 PROMPT
11 STO 01
12 "H.MMSS IN USE"
13 AVIEW
14 PSE
15 "PER DAY ?"
16 PROMPT
17 STO 02
18 "DAYS/MONTHS?"
19 PROMPT
20 STO 03
21 RCL 02
22 *
23 STO 03
24 FIX 0
25 "HRS./MO.="
26 ARCL 03
27 AVIEW
28 PSE
29 HR
30 STO 03
31 RCL 01
32 *
33 RCL 00
34 *
35 1000
36 /
37 STO 03
38 FIX 2
39 "COST=$"
40 ARCL 03
41 AVIEW
42 STOP
43 GTO 01
44 END
```

Fig. 10-4. The same program revised for the HP-41C.

for 25 days of the month. After the computer goes through the same steps as before, it tells us that we are looking at a total use of 250 hours and the cost for the month will be $2.02.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 004
1. Enter cost of electricity per kilowatt	Display: *COST KWH ?*
2. Press **A**	1. Enter cost of electricity per kilowatt hour; **R/S**
3. Enter watt rating of appliance	Display: **APPL. WATT?**
4. Press **B**	2. Enter the watt rating of the appliance: **R/S**
5. Enter hours of use per day	3. Enter daily use in hours, minutes and seconds: **R/S**
6. Press **C**	Display: *DAYS/MONTH?*
7. Enter number of days in use per month	4. Enter the days per month when the appliance is in use; **R/S**
8. Press **R/S**	Display: *HRS./MO.=* and total hours of use
Display: Total number of hours in use, per month	5. Press **R/S**
9. Press **R/S**	Display: *COST=$* and total cost
Display: Cost of use	

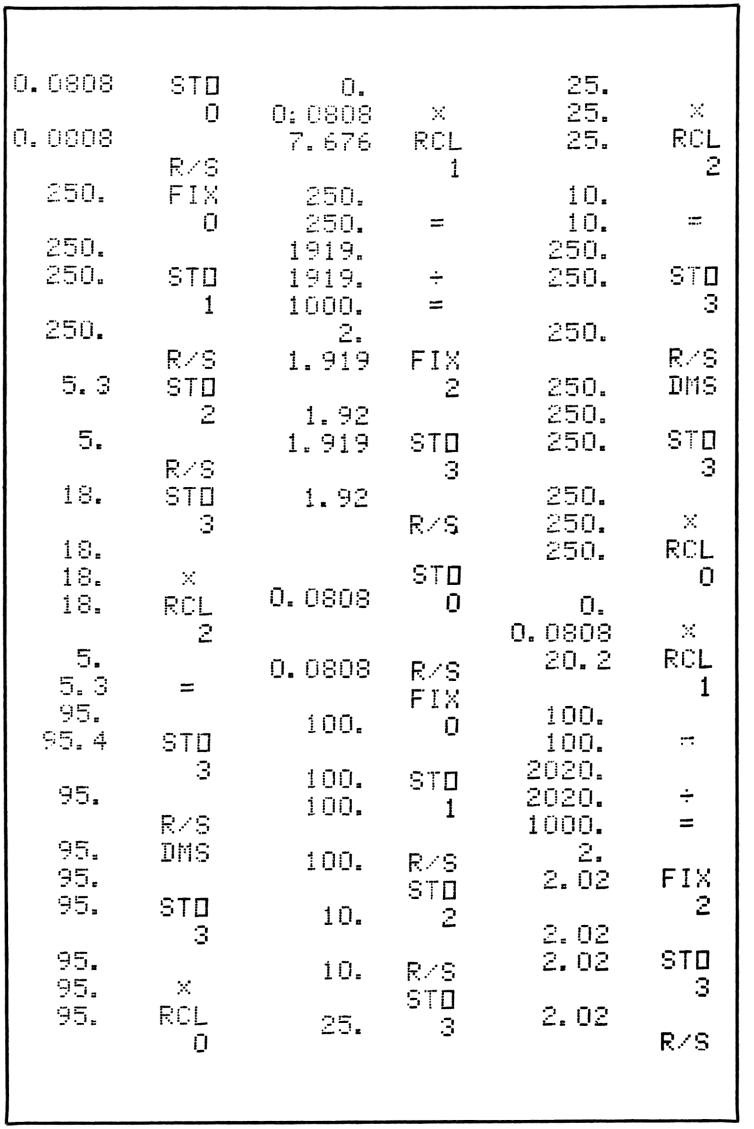


Fig. 10-5. The TI-59 optional printer provides a record of what happens when the Electricity program is run.

COMPOUND INTEREST

In these days of constantly fluctuating interest rates, trying to figure out what a given amount of interest will actually cost us (or earn us) over a period of time is becoming increasingly important. This program is designed to figure compound as well as straight interest at any rate for any

time period, allowing for daily, monthly or yearly compounding periods (the yearly period being equivalent to straight interest). The program consists of 138 steps in the HP and 234 steps in the TI. The basic mathematical formula for figuring compound interest is quite simple and not particularly long, using only 15 steps in the HP and 28 steps in the TI. The rest of the program steps and sections are used to process input data in such a way that, when they are finally used by the computer in running the compound-interest formula, the result is correct.

TI-59. Steps 000 through 003 are used to clear extraneous data from the computer and to restrict the number of displayed decimals to two (Fig. 10-7). Steps 004 through 023 are used to store the information about principal, interest rates, compounding periods and time to be calculated in the same registers in which it is stored in the HP: R_{00}, R_{01}, R_{03}, R_{04}.

LBL E, consisting of steps 024 through 042, tells the computer where to go next, depending on the information relative to the compounding period.

LBL A′, which consists of steps 043 through 089, is comparable to **LBL 01** in the HP. It converts the months and days into fractions of years. The result is added to the years and the compounding period is expressed in years and fractions of years.

LBL D′, steps 090 through 118, performs the actual calculation and, in steps 10. through 118, places the results into display. **LBL B′**, steps 120 through 172, transforms the years/months/days data into months and fractions of months. In addition, it divides the interest rate by 12 to arrive at a monthly figure for the interest.

LBL C′, steps 173 through 234, transforms the years/months/days data into days, and it divides the interest rate by 365.25, the number of days in a year, thus preparing the computer to use a daily interest rate when, as told in steps 233 and 234, it goes to **LBL D′ (GTO D′)** for the actual calculation and the display of the results.

HP-41C. When called up, the program (Fig. 10-8) identifies itself with *COMPOUND INT.* and then changes to *INVESTMENT?* while, in steps 05 and 06, clearing the computer of unwanted left-over data (**CLRG**) and limiting the displayed decimals to two (**FIX 2**). The amount of investment or the amount of the obligation being financed at a given interest rate is stored in R_{00}. The computer then asks *INT. RATE, % ?*; the answer to that request is stored in R_{01}. The next display reads *COMP.PERIOD?* and then changes to *Y:12/M:30/D:1*, which means that if the compounding period calls for daily compounding, press **1**, if the period is monthly press **30**, and if the period is yearly (in other words, straight interest) press **12**. The last information needed by the computer is *TIME: Y.MMDD*, meaning the time expressed in years-decimal point-months-days for which the interest is to be figured (1.0605 represents one year, six months and five days. Months and days must be entered as two digits each).

Let's enter data for a fictitious situation: Investment, **1000** ($1,000.00); **R/S**; Interest rate, **14.75** (14.75 percent); **R/S**; Compounding

```
01*LBL "PPG 0-9"
"ELECTRICITY"
AVIEW
ELECTRICITY
PSE
"COST KWH ?K"
PROMPT
COST KWH ?K
0.0800 RUN
STO 00

08*LBL 01
"APPL. WATT ?"
PROMPT
APPL. WATT ?
250.00 RUN
STO 01
"H.MMSS IN USE"
AVIEW
H.MMSS IN USE
PSE
"PER DAY ?"
PROMPT
PER DAY ?
5.30 RUN
STO 02
"DAYS/MONTH"
PROMPT
DAYS/MONTH
18.00 RUN
STO 03
RCL 02
5.30 ***
*
95.40 ***
STO 03
FIX 0
"HRS./MO.="
ARCL 03
AVIEW
HRS./MO.=95.
PSE
HR
96. ***
STO 03
RCL 01
250. ***
*
23,917. ***
RCL 00
8.-02 ***
*
1 932. ***
1000
/
2. ***
STO 03
FIX 2
"COST $"
ARCL 03
AVIEW
COST $1.93
STOP
RUN
GTO 01

08*LBL 01
"APPL. WATT ?"
PROMPT
APPL. WATT ?
100.00 RUN
STO 01
"H.MMSS IN USE"
AVIEW
H.MMSS IN USE
PSE
"PER DAY ?"
PROMPT
PER DAY ?
10.00 RUN
STO 02
"DAYS/MONTH"
PROMPT
DAYS/MONTH
25.00 RUN
STO 03
RCL 02
10.00 ***
*
250.00 ***
STO 03
FIX 0
"HRS./MO.="
ARCL 03
AVIEW
HRS./MO.=250.
```

Fig. 10-6. A trace of the Electricity program in the HP-41C.

```
          PSE                          2,020.    ***
           HR                            1000
      250.    ***                           /
      STO 03                             2.    ***
      RCL 01                           STO 03
      100.    ***                      FIX 2
           *                           "COST $"
   25,000.    ***                      ARCL 03
      RCL 00                           AVIEW
     8.-02    ***          COST $2.02
           *                           STOP
```

period, daily, press **1**; **R/S**; Time period, **3.0615** (three years, six months, 15 days); **R/S**. After a moment the computer displays: *TOTAL=$1.685.58*. Then press **R/S** and the display changes to *INT. =$685.58*.

To take a look inside the computer, steps 22 through 36 are used to guide the computer to the appropriate section of the program, depending on the compounding period. If it's yearly, the computer is told to **GTO 01**; if it's monthly the appropriate command is **GTO 02**; for daily compounding it's **GTO 03**.

Steps 37 through 57 are **LBL 01**, the section used for straight interest or yearly compounding periods. In fact, it transforms months and days into fraction of a year. In our example, those three years, six months and 15 days would have become 3.541068 years. The last step in that section, step 57, is **RCL 01**, recalling the interest rate and placing it into the X register.

Steps 58 through 73 are **LBL 04**, the section which performs the actual calculation. Remember that the interest rate (**RCL 01**) is still in the X register.

The computer now starts by dividing that rate by 100. It adds 1 and stores the result (which is now 1.1475 in our example) in R_{02}. The next three steps, 64 through 66, raise the figure stored in R_{02} to the power of the compound periods, the figure stored in R_{06}. (Written out, this would look like: $1.1457^{3.541068}$.) That rather complicated figure is multiplied by the amount of the investment stored in R_{00} and the result is stored in R_{07} for later display. Stored in R_{07} is the total amount of principal plus interest. In steps 70 through 72 the computer deducts the principal from that total and stores the interest alone in R_{08}. The computer is then told to **GTO 05**.

LBL 05 is the last section of the program, steps 126 through 138. Its sole purpose is to arrange for the display of the data stored in R_{07} and R_{08} and to place an *END* into display.

In **LBL 02** the computer transforms the years, months and days into months and fractions of months (in our example the result, stored in R_{06} would look like 42.493421 months). But it also divides the interest rate by

```
000 CMs   029 01   059 01   088 RCL   117 08   146 =     175 RCL   204 x     233 GTO
001 CLR   030 02   060 02   089 01    118 R/S   147 STO   176 01    205 01    234 D'
002 FIX   031 A'   061 =    090 LBL   119 LBL   148 05    177 /     206 00       0
003 02    032 RCL  062 STO  091 D'    120 B'    149 INT   178 03    207 00       0
004 LBL   033 03   063 06   092 /     121 RCL   150 SUM   179 06    208 =
005 A     034 x≷t  064 RCL  093 01    122 01    151 06    180 05    209 STO
006 STO   035 03   065 05   094 00    123 /     152 RCL   181 .     210 05
007 00    036 00   066 INV  095 00    124 01    153 05    182 02    211 INT
008 R/S   037 B'   067 INT  096 =     125 02    154 INV   183 05    212 x
009 LBL   038 RCL  068 x    097 +     126 =     155 INT   184 =     213 03
010 B     039 03   069 01   098 01    127 STO   156 x     185 STO   214 00
011 STO   040 x≷t  070 00   099 =     128 02    157 01    186 02    215 .
012 01    041 01   071 00   100 STO   129 RCL   158 00    187 RCL   216 04
013 R/S   042 C'   072 =    101 02    130 04    159 00    188 04    217 =
014 LBL   043 LBL  073 /    102 y^x   131 INT   160 =     189 INT   218 SUM
015 C     044 A'   074 03   103 RCL   132 x     161 /     190 x     219 06
016 STO   045 RCL  075 06   104 06    133 01    162 03    191 03    220 RCL
017 03    046 04   076 05   105 x     134 02    163 00    192 06    221 05
018 R/S   047 INV  077 .    106 RCL   135 =     164 .     193 05    222 INV
019 LBL   048 INT  078 02   107 00    136 STO   165 04    194 .     223 INT
020 D     049 x    079 05   108 =     137 06    166 =     195 02    224 x
021 STO   050 01   080 =    109 STO   138 RCL   167 SUM   196 05    225 01
022 04    052 00   081 SUM  110 07    139 04    168 06    197 =     226 00
023 R/S   053 00   082 06   111 R/S   140 INV   169 RCL   198 STO   227 00
024 LBL   054 =    083 TCL  112 -     141 INT   170 02    199 06    228 =
025 E     055 STO  084 04   113 RCL   142 x     171 GTO   200 RCL   229 SUM
026 RCL   056 05   085 INT  114 00    143 01    172 D'    201 04    230 06
027 03    057 INT  086 SUM  115 =     144 00    173 LBL   202 INV   231 RCL
028 x≷t   058 /    087 06   116 STO   145 00    174 C'    203 INT   232 02
```

Fig. 10-7. Compound Interest on the TI-59

12 to arrive at the monthly interest (in steps 75 through 78) which is then stored in R_{02} (in our example that would be 1.229167 percent). The very end of the section leaves the contents of R_{02} in the X register. The computer is told to **GTO 04** where the calculation, this time using the contents of R_{02} rather than R_{01}, takes place.

LBL 03 performs the same tasks for the daily compounding period. In steps 101 through 104 it divides the interest rate by 365.25 (the number of days in a year) and stores the result, 0.040383, in R_{02}. It then transforms the years, months and days into days by multiplying the years by 365.25, and the months by 30.4. By adding these results to the days originally entered, the computer arrives at the number of days which represent the total compounding periods (1,293.15 days), again storing that result in R_{06} to be used when **LBL 04** is being executed.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 2
Passes: 2	Passes: 4
Partition: 479 59	Size: 017
1. Enter investment	Display; *INVESTMENT*
2. Press **A**	1. Enter investment; **R/S**
3. Enter interest rate	Display: *INT.RATE %*
4. Press **B**	2. Enter interest rate; **R/S**
5. Enter **12** if interest if compounded annually, **30** if it is compounded monthly, or **1** if it is compounded daily.	Display: *COMP.PERIOD*, then *Y:12, M:30, D:1*
6. Press **C**	3. Enter **12, 30** or **1**, depending on the compounding period; **R/S**
7. Enter time period in Y.MMDD format (3.0406 represents 3 years, 4 months, 6 days)	Display: *TIME PERIOD*
8. Press **D**	4. Enter the time period in Y.MMDD format (3.0406 equals 3 years, 4 months, 6 days); **R/S**
Display: The initial investment amount plus the interest	Display: *$* and total of investment plus interest
9. Press **R/S**	5. Press **R/S**
Display: The interest alone	Display: The interest alone

GAS MILEAGE

In reciprocating engines, the ones driving our cars or the ones powering our airplanes, the faster we go the more gas we use per mile. What we rarely know is the exact relationship between different speeds and different rates of fuel consumption. This program is designed to answer those questions.

```
       01♦LBL "CI"
 02 "COMPOUND INT."
       03 AVIEW
       04 PSE
       05 CLRG
       06 FIX 2
  07 "INVESTMENT ?"
       08 PROMPT
       09 STO 00
 10 "INT. RATE,% ?"
       11 PROMPT
       12 STO 01
 13 "COMP. PERIOD?"
       14 AVIEW
       15 PSE
 16 "Y:12/M:30/D:1"
       17 PROMPT
       18 STO 03
19 "TIME: Y.MMDD ?"
       20 PROMPT
       21 STO 04
       22 RCL 03
       23 ENTER↑
       24 12
       25 X=Y?
       26 GTO 01
       27 RCL 03
       28 ENTER↑
       29 30
       30 X=Y?
       31 GTO 02
       32 RCL 03
       33 ENTER↑
       34 1
       35 X=Y?
       36 GTO 03

       37♦LBL 01
       38 RCL 04
       39 FRC
       40 100
       41 *
       42 STO 05
       43 INT
       44 12
       45 /
       46 STO 06
       47 RCL 05
       48 FRC
       49 100
       50 *
       51 365.25
       52 /
       53 ST+ 06
       54 RCL 04
       55 INT
       56 ST+ 06
       57 RCL 01
       58♦LBL 04
       59 100
       60 /
       61 1
       62 +
       63 STO 02
       64 ENTER↑
       65 RCL 06
       66 Y↑X
       67 RCL 00
       68 *
       69 STO 07
       70 RCL 00
       71 -
       72 STO 08
       73 GTO 05

       74♦LBL 02
       75 RCL 01
       76 12
       77 /
       78 STO 02
       79 RCL 04
       80 INT
       81 12
       82 *
       83 STO 06
       84 RCL 04
       85 FRC
       86 100
       87 *
       88 STO 05
       89 INT
       90 ST+ 06
       91 RCL 05
       92 FRC
       93 100
       94 *
       95 30.4
       96 /
       97 ST+ 06
       98 RCL 02
       99 GTO 04

      100♦LBL 03
      101 RCL 01
      102 365.25
      103 /
      104 STO 02
      105 RCL 04
      106 INT
      107 365.25
      108 *
      109 STO 06
      110 RCL 04
      111 FRC
      112 100
      113 *
      114 STO 05
```

Fig. 10-8. Compound Interest for the HP-41C.

```
115 INT          123 ST+ 06       131 "INT.=$"
116 30.4         124 RCL 02       132 ARCL 08
117 *            125 GTO 04       133 AVIEW
118 ST+ 06       126*LBL 05       134 STOP
119 RCL 05      127 "TOTAL=$"     135 "END"
120 FRC          128 ARCL 07      136 AVIEW
121 100          129 AVIEW        137 STOP
122 *            130 STOP         138 END
```

The data needed to perform the necessary computations are a low and a high speed for which the rate of fuel consumption has been pretty well established. We need those rates, and we need to key in the speed for which we want an answer. Here is how it works:

TI-59. With the program for the TI, use the **A** key for the low and high speeds, the **B** key for the rates of fuel consumption, the **C** key for the speed for which we want the answer, and for the answer itself.

First enter **35** and press **A**. Then key in **75** and press **R/S**. Now enter **38** and press **B** followed by **23** and **R/S**. That takes care of the data for the speeds for which fuel consumption is known, and the rates of consumption for those two speeds (Fig. 10-9). Key in the speed for which we want the fuel consumption rate to be computed, **55**, and press **C**. The entry shows in the display. Press **R/S** and the result appears as *29.4*. (The two computers do not totally agree here.) Key in **65, C** and **R/S** and the displayed result is *27.2*. Key in **45, C** and **R/S** and the new result is *31.7*.

It might be a good idea to speed up this program a bit by eliminating step 022 (R/S), which would mean that as soon as the speed is entered and **C** is pressed, the computer would go to work on the computation.

HP-41C. The display identifies the program (Fig. 10-10) with *GAS MILEAGE* and then changes to *LOW SPEED?* Let's say we enter **35** as a low speed for which we have established the amount of fuel being used. Press **R/S** and the display asks *HIGH SPEED?* Here enter **75** and press **R/S**. The display now asks *MPG AT LO S?* The computer wants to know the miles per gallon applicable to the low speed previously entered. Enter **38** and press **R/S**. It now asks *MPG AT HI S.?* Here enter **23**. Pressing **R/S** displays the request *SPEED MPH?* Let's say we want to know what happens at 55 mph, so key in **55** and **R/S**. The displayed result, *MPG=30.5* gives us the gas mileage we can expect at 55 mph. Pressing **R/S** again repeats *SPEED MPH?* and we might now enter **65**, which provides *MPG=26.8* or, if we had called for 45 mph, the result would have been *MPG= 34.3*.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter a low speed for which gas mileage is known
2. Press **A**
3. Enter a high speed for which gas mileage is known
4. Press **R/S**
5. Enter miles-per-gallon for low speed
6. Press **B**
7. Enter mpg for high speed
8. Press **R/S**
9. Enter speed for which mpg is to be calculated
10. Press **E**
11. Press **R/S**
Display: mpg figure for speed entered in step 9

HP-41C

Cards: 1
Passes: 2
Size: 011
Display: *LOW SPEED?*
1. Enter low speed for which mpg is known; **R/S**
Display: *HIGH SPEED?*
2. Enter high speed with known mpg; **R/S**
Display: *MPG AT LO S.?*

000	58	FIX	024	01	01	048	43	RCL
001	01	01	025	75	-	049	00	00
002	76	LBL	026	43	RCL	050	95	=
003	11	A	027	00	00	051	42	STO
004	42	STO	028	95	=	052	06	06
005	00	00	029	42	STO	053	43	RCL
006	91	R/S	030	04	04	054	09	09
007	42	STO	031	43	RCL	055	65	×
008	01	01	032	02	02	056	43	RCL
009	91	R/S	033	75	-	057	06	06
010	76	LBL	034	43	RCL	058	95	=
011	12	B	035	03	03	059	42	STO
012	42	STO	036	95	=	060	07	07
013	02	02	037	42	STO	061	75	-
014	91	R/S	038	05	05	062	43	RCL
015	42	STO	039	55	÷	063	02	02
016	03	03	040	43	RCL	064	95	=
017	91	R/S	041	04	04	065	94	+/-
018	76	LBL	042	95	=	066	42	STO
019	13	C	043	42	STO	067	08	08
020	42	STO	044	09	09	068	91	R/S
021	10	10	045	43	RCL	069	00	0
022	91	R/S	046	10	10			
023	43	RCL	047	75	-			

Fig. 10-9. The Gas Mileage program for the TI-59.

3. Enter mpg for low speed; **R/S**
Display: *MPG AT HI S.?*
4. Enter mpg for high speed; **R/S**
Display: *SPEED MPH?*
5. Enter speed for which mpg is to be calculated; **R/S**
Display: *MPG=* and gas mileage for speed entered in step 5.

```
01♦LBL "0"          17 RCL 01          33 -
02 "GAS MILAGE"     18 RCL 00          34 STO 06
03 AVIEW            19 -               35 RCL 09
04 PSE              20 STO 04          36 *
05 "LOW SPEED ?"    21 RCL 02          37 STO 07
06 PROMPT           22 RCL 03          38 RCL 02
07 STO 00           23 -               39 -
08 "HIGH SPEED ?"   24 STO 05          40 CHS
09 PROMPT           25 RCL 04          41 STO 08
10 STO 01           26 /               42 FIX 1
11 "MPG AT LO S.?"  27 STO 09          43 "MPG="
12 PROMPT           28♦LBL 00          44 ARCL 08
13 STO 02       29 "SPEED MPH ?"       45 AVIEW
14 "MPG AT HI S.?"  30 PROMPT          46 STOP
15 PROMPT           31 STO 10          47 GTO 00
16 STO 03           32 RCL 00          48 END
```

Fig. 10-10. The Gas Mileage program for the HP-41C.

HOW MANY TILES TO COVER THE FLOOR?

Tiles for kitchen, den or other floors come in many different materials and in many different sizes. Their prices can vary greatly. To complicate matters, rooms aren't of the same size, and frequently we're not even interested in covering an entire room.

One way or the other, we need to know how many tiles we need to cover a given area, and it would be nice to know in advance what that luxury is going to cost us. This program accepts information about the size of the tiles in inches, data about the size of the area to be covered in feet, and the cost per tile in dollars and cents. It then digests all that information and displays the number of tiles required and the cost (excluding labor). (This program can be used with square tiles only. In the unlikely even the tiles you want to use are not square but rectangular, use the program entitled HOW MANY BRICKS TO BUILD A WALL?)

Let's quickly walk through the program:

TI-59. After the magnetic card with the program has been passed through the card-reader slot, the display must respond with a steady **1** (Fig.

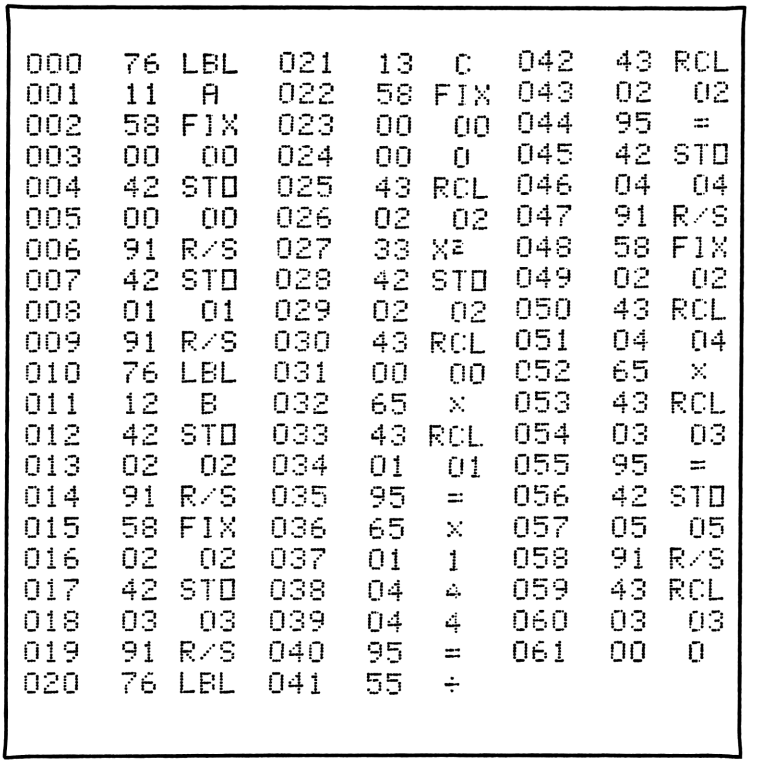


Fig. 10-11. The Floor Tiles program for the TI-59.

10-11). If not, repeat. Key in **12** for one side of the floor area and press **A**. Then key in **16.5** for the other side of the floor area and press **R/S**. Key in **9** for the tile size and press **B**.

After that, key in **0.83** for the price per tile and press **R/S**. (In other words, the floor-area data are under key **A**, the tile data under key **B**.) Now press **C** and the display will respond with *352*, the number of tiles required. Press **R/S** and the display changes to *292.16*, the cost of the tiles.

HP-41C. Turn the computer on and it will display *FLOOR TILES* once the program has been called up (Fig. 10-12). That display changes to *TILE SIZE?* Let's put in **9**, as nine-inch tiles are particularly common.

R/S results in the request *FLOOR SIDE 1*. Let's put in **12** for 12 feet. It then asks for *FLOOR SIDE 2*.

Say that the room is 16 feet 6 inches long. Key in **16.5**. Next question: *$PER TILE?* Say they cost 83 cents each. Be sure the amount is entered in the dollar format, meaning **0.83**, as otherwise the computer will produce some astronomical figure. The display now shows *NO.TILES=352*. Pressing **R/S** changes that to *TOTAL=$292.16*.

```
01↑LBL "Y"        15 PROMPT        29 ARCL 04
    02 FIX 0          16 STO 01        30 AVIEW
03 "FLOOR TILES"  17 "$ PER TILE ?"  31 STOP
    04 AVIEW          18 PROMPT        32 FIX 2
    05 PSE            19 STO 03        33 RCL 04
06 "TILE SIZE ?"      20 RCL 00        34 RCL 03
    07 PROMPT         21 RCL 01        35 *
    08 STO 02         22 *             36 STO 05
    09 X↑2            23 144          37 "TOTAL=$"
    10 STO 02         24 *             38 ARCL 05
11 "FLOOR SIDE 1"     25 RCL 02        39 AVIEW
    12 PROMPT         26 /             40 STOP
    13 STO 00         27 STO 04        41 END
14 "FLOOR SIDE 2"   28 "NO. TILES="
```

Fig. 10-12. The Floor Tiles program for the HP-41C.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 006
1. Key in floor, side 1	Display: *TILE SIZE?*
2. Press **A**	1. Enter length of one side: **R/S**
3. Key in floor, side 2	Display: *FLOOR SIDE 1?*
4. Press **R/S**	2. Enter length, side 1; **R/S**
5. Key in tile size (one side)	Display: *FLOOR SIDE 2?*
6. Press **B**	3. Enter length, side 2; **R/S**
7. Key in price per tile	Display: *$ PER TILE?*
8. Press **R/S**	4. Enter cost per tile; **R/S**
9. Press **C**	Display: *NO. TILES=* and total
Display: Number of tiles	5. Press **R/S**
10. Press **R/S**	Display: *TOTAL = $* and total cost
Display: Total cost	

BUILDING A FENCE

This program is an example of accumulating a series of data and then relating them to one another to arrive at one or several results. Such a program may not occur too often by itself, but it will frequently turn up possibly as a subroutine as part of a longer and more complicated program.

I have assumed we want to build a fence of a given length, with fence posts spaced a certain distance apart. We know the cost of the fence,

```
000   76 LBL   028   76 LBL   056   95  =
001   11  A    029   14  D    057   42 STO
002   58 FIX   030   42 STO   058   07  07
003   00  00   031   04  04   059   58 FIX
004   42 STO   032   91 R/S   060   02  02
005   00  00   033   76 LBL   061   43 RCL
006   91 R/S   034   15  E    062   05  05
007   76 LBL   035   43 RCL   063   91 R/S
008   13  B    036   00  00   064   58 FIX
009   42 STO   037   65  ×    065   00  00
010   01  01   038   43 RCL   066   43 RCL
011   35 1/X   039   03  03   067   02  02
012   65  ×    040   95  =    068   91 R/S
013   43 RCL   041   42 STO   069   58 FIX
014   00  00   042   05  05   070   02  02
015   95  =    043   43 RCL   071   43 RCL
016   42 STO   044   02  02   072   06  06
017   02  02   045   65  ×    073   91 R/S
018   43 RCL   046   43 RCL   074   43 RCL
019   01  01   047   04  04   075   07  07
020   91 R/S   048   95  =    076   91 R/S
021   76 LBL   049   42 STO   077   00  0
022   13  C    050   06  06
023   58 FIX   051   43 RCL
024   02  02   052   05  05
025   42 STO   053   85  +
026   03  03   054   43 RCL
027   91 R/S   055   06  06
```

Fig. 10-13. The TI-59 version of the Building a Fence program.

including labor, per finished foot, and we know the cost of each fencepost. We need to know the number of fence posts required, because they have to be ordered along with the material for the fencing itself, and, of course, what all that is going to cost.

Let's run it and see what happens.

TI-59. The letter keys **A** through **D** are used to enter the information which the computer needs (Fig. 10-13), and pressing **E** followed by **R/S** several times produces the following results:

Key in **1200** and press **A**.

Key in **7.5** (distance between posts) and press **B**.

Key in **3.50** (cost of fence per foot) and press **C**.

Key in **4.15** (cost per post) and press **D**.

Now press **E** and up comes *4200.00*. Now press **R/S** and the display shows *160*, the number of fence posts needed. Press **R/S** again and the

display shows *664.00* the cost of the posts. One last **R/S** changes the display to *4864.00* the total cost of the fence.

HP-41C. After identifying the program (Fig. 10-14) as *FENCE*, it asks *LENGTH?* Enter the total length, say **1,200** feet (assuming a 200 by 400 foot property). Press **R/S** and receive the question *FT.BETW. POSTS?* We have decided we can get away with seven feet, six inches between posts, so enter **7.5**. The display now asks *$ PER FOOT*. The fencing itself has been quoted at $3.25 per foot, and the fellow who is going to install it gave us a flat bid of $150 for the whole deal, 0.13 cents per foot. Knowing from experience that things always end up costing more, add 25 cents per foot and key in *3.50*. Now the display asks *$ per post?*, and based on the quote from the supplier key in **4.15**.

Now the computer has all the data it needs, and the first display simply recalls the length of the fence to make sure we've got in right in the first place: *FENCE FT=1.200*. Then it reads *POST=160*, the number that will be needed. Pressing R/S again tells us the fence is going to cost $4,200.00 not including the posts: *FENCE= $4,200.00*. The next display is *POSTS= $664.00*, and pressing **R/S** once more puts the total bad news at *TOTAL=$4,864.00* (at that price, maybe we better forget about the whole thing).

```
01*LBL "W"          20 PROMPT          39 "FENCE=$"
02 FIX 0            21 STO 04          40 ARCL 05
03 "FENCE"          22 RCL 00          41 AVIEW
04 AVIEW            23 RCL 03          42 STOP
05 PSE              24 *               43 "POSTS=$"
06 "LENGTH ?"       25 STO 05          44 ARCL 06
07 PROMPT           26 RCL 02          45 AVIEW
08 STO 00           27 RCL 04          46 STOP
09 "FT.BETW.POSTS?" 28 *               47 RCL 05
10 PROMPT           29 STO 06          48 RCL 06
11 STO 01           30 "FENCE FT.="    49 +
12 1/X              31 ARCL 00         50 STO 07
13 RCL 00           32 AVIEW           51 "TOTAL=$"
14 *                33 STOP            52 ARCL 07
15 STO 02           34 "POSTS="        53 AVIEW
16 "$ PER FOOT?"    35 ARCL 02         54 STOP
17 PROMPT           36 AVIEW           55 " "
18 STO 03           37 STOP            56 END
19 "$ PER POST?"    38 FIX 2
```

Fig. 10-14. Building a Fence, here in the HP-41C version.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 008
1. Key in length of fence	Display: *LENGTH?*
2. Press **A**	1. Enter length of fence; **R/S**
3. Key in distance between fence posts	Display: *FT.BETW. POSTS?*
4. Press **B**	2. Enter feet between posts; **R/S**
5. Key in cost per fence feet	Display: *$ PER FOOT?*
6. Press **C**	3. Enter cost of fence per foot; **R/S**
7. Key in cost per post	Display: *$ PER POST?*
8. Press **D**	4. Enter cost per post; **R/S**
9. Press **E**	Display; *FENCE FT* = recall length of fence
Display: Cost of fence	5. Press **R/S**
10. Press **R/S**	Display: *POSTS* = and number
Display: Cost of posts	6. Press **R/S**
11. Press **R/S**	Display; *FENCE* = *$* and costs
Display: Total cost	7. Press **R/S**
	Display: *POSTS* = *$* and costs
	8. Press **R/S**
	Display: *TOTAL* = *$* and total cost

TRAVEL EXPENSES

It is often important, especially for persons on an expense account, to keep track of their travel costs. This program accepts all trip-related data and provides the user with individual subtotals as well as overall totals. Let's take a theoretical trip and see how that works.

TI-59. This is one instance where the TI must be used differently than the HP (Fig. 10-15). Whenever we turn the TI off, all data and program information are lost and have to be reentered for the next use. For that reason, it would appear to be a lot simpler to keep track of the individual data throughout the day with pencil and paper, and to use the computer only in the evenings to get a clear picture of what each day cost and what the different subtotals were spent for.

The TI program is assigned to different keys which handle the individual segments of the program: Key **A** collects data about the fuel quantity.

```
000  25 CLR   034  44 SUM   068  43 RCL
001  47 CMS   035  06  06   069  04  04
002  76 LBL   036  43 RCL   070  91 R/S
003  11  A    037  06  06   071  58 FIX
004  58 FIX   038  91 R/S   072  00  00
005  01  01   039  76 LBL   073  43 RCL
006  42 STO   040  14  D    074  06  06
007  01  01   041  58 FIX   075  91 R/S
008  44 SUM   042  02  02   076  58 FIX
009  02  02   043  42 STO   077  02  02
010  43 RCL   044  07  07   078  43 RCL
011  02  02   045  44 SUM   079  08  08
012  91 R/S   046  08  08   080  91 R/S
013  76 LBL   047  43 RCL   081  43 RCL
014  12  B    048  08  08   082  10  10
015  58 FIX   049  91 R/S   083  91 R/S
016  02  02   050  76 LBL   084  43 RCL
017  42 STO   051  15  E    085  04  04
018  03  03   052  42 STO   086  85  +
019  65  ×    053  09  09   087  43 RCL
020  43 RCL   054  44 SUM   088  08  08
021  01  01   055  10  10   089  85  +
022  95  =    056  43 RCL   090  43 RCL
023  44 SUM   057  10  10   091  10  10
024  04  04   058  91 R/S   092  95  =
025  43 RCL   059  76 LBL   093  91 R/S
026  04  04   060  10 E'    094  00  0
027  91 R/S   061  58 FIX
028  76 LBL   062  01  01
029  13  C    063  43 RCL
030  58 FIX   064  02  02
031  00  00   065  91 R/S
032  42 STO   066  58 FIX
033  05  05   067  02  02
```

Fig. 10-15. The Travel Expenses program in the TI-59.

Every time a new amount of gas is added to the previously entered amount, the figure is keyed in, followed by pressing **A** (do not press **R/S**). Key **B** handles the cost of the gas in the same manner. Key **C** is used to add up the miles, key **D** the cost of the food, key **E** the cost of the lodging, and pressing **2nd E'** and then **R/S** will display, one after another, the totals for each of the categories. The first display after pressing **2nd E'** is the quantity of gas purchased. Pressing **R/S** produces the cost of the gas. Pressing **R/S** again displays the total miles traveled, and pressing **R/S** once more produces the total spent for food. The next **R/S** results in the total for lodging and the last

R/S combines all those subtotals to come up with the complete total for the period covered by the entries.

HP-41C. After identifying the program (Fig. 10-16) with *TRIP DATA*, the display asks *FUEL ON BD*. It wants to know the amount of fuel with which the trip started.

This program works just as well for aircraft as it does for automobiles. We'll assume this to be an automobile trip. Enter **16** as the number of gallons on board at the start of the trip. **R/S** results in **$ PER GAL?** What did we pay for it? Let's say, **$1.349** per gallon. Enter it and the computer tells us the initial gas cost *$21.58*.

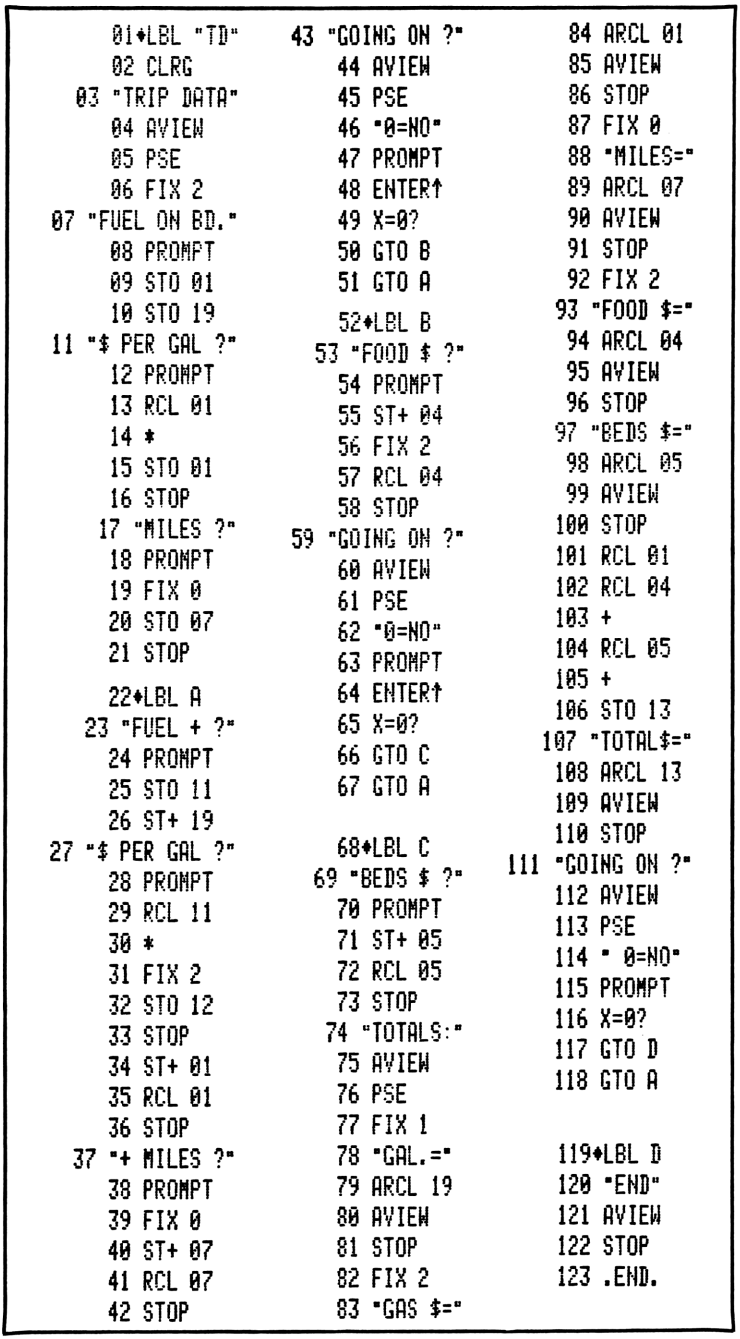
Press **R/S** and the computer asks *MILES?* This step would, in reality, not be taken until we stopped for gas somewhere along the way. Say we've been driving **208** miles. Enter that and the computer replies that the total mileage so far is, in fact, *208*. It now asks *FUEL +?* This time we've taken **9.8** gallons and enter that figure. Next question: *$ PER GAL?* Away from a major city it cost **1.429**. Enter that and we find the current fuel bill is *$14.00* and the total fuel cost so far is *$35.59*.

When we stop again, this time for gas as well as something to eat, the computer when **R/S** is pressed, asks again *+MILES?* Enter **164**; the total today is *372*. When we press **R/S** again the display wants to know *GOING ON?* It then changes to *0=NO*. Let's assume that we're not going on. Press **0** and then **R/S** and the display now asks *FOOD $?* We've had a good meal, so enter **8.75** which includes the top and the computer responds by telling us *$8.75* is all that has been spend on food so far. Now it asks again, *GOING ON?* followed by *0=NO*.

We don't like the looks of the motels around here, so we decide to go on to the next town. Press **R/S**: *FUEL +?* Are we taking on any fuel? Might as well. **7.1** gallons fill the tank, and the price here is *$1.367*, resulting in a fuel bill of *9.71*. The next display says the total fuel so far has cost *$45.29*.

Next time we stop, and this time definitely to spend the night, the display asks again *+MILES?* and we enter the last distance traveled, **84** miles. That brings the total mileage for the day, according to the next figure in the display, to *456*. Next question: *GOING ON?* and *0=NO*. Press **0**, then **R/S**, and the computer asks *FOOD $?* After dinner we enter **12.95** and press **R/S** to be shown that today's food, so far, has cost *$21.70*. *GOING ON? 0=NO*. Again press **0** and **R/S** and now the display asks *BEDS $?* The motel costs **$48** for the night, so enter that amount the next morning, and the display tells us the total spend for lodging so far is *$48.00*. Pressing **R/S** results in *TOTALS:* which changes to *GAL.= 32.9*. Press **R/S**: *GAS $=45.29*. Press **R/S**: *MILES= 456*. Press **R/S**: *FOOD $= 21.70*. Press **R/S**: *BEDS $= 48.00*. Press **R/S**: *TOTAL $=114.99*.

These data can be left in the computer and as the trip is continued the next day, the entire routine can be repeated and the new figures will automatically be added to the previous totals, giving us a complete accounting of how much was spent for what, and how much the total came to at the end of the trip.



```
01♦LBL "TD"
02 CLRG
03 "TRIP DATA"
04 AVIEW
05 PSE
06 FIX 2
07 "FUEL ON BD."
08 PROMPT
09 STO 01
10 STO 19
11 "$ PER GAL ?"
12 PROMPT
13 RCL 01
14 *
15 STO 01
16 STOP
17 "MILES ?"
18 PROMPT
19 FIX 0
20 STO 07
21 STOP

22♦LBL A
23 "FUEL + ?"
24 PROMPT
25 STO 11
26 ST+ 19
27 "$ PER GAL ?"
28 PROMPT
29 RCL 11
30 *
31 FIX 2
32 STO 12
33 STOP
34 ST+ 01
35 RCL 01
36 STOP
37 "+ MILES ?"
38 PROMPT
39 FIX 0
40 ST+ 07
41 RCL 07
42 STOP
43 "GOING ON ?"
44 AVIEW
45 PSE
46 "0=NO"
47 PROMPT
48 ENTER↑
49 X=0?
50 GTO B
51 GTO A

52♦LBL B
53 "FOOD $ ?"
54 PROMPT
55 ST+ 04
56 FIX 2
57 RCL 04
58 STOP
59 "GOING ON ?"
60 AVIEW
61 PSE
62 "0=NO"
63 PROMPT
64 ENTER↑
65 X=0?
66 GTO C
67 GTO A

68♦LBL C
69 "BEDS $ ?"
70 PROMPT
71 ST+ 05
72 RCL 05
73 STOP
74 "TOTALS:"
75 AVIEW
76 PSE
77 FIX 1
78 "GAL.="
79 ARCL 19
80 AVIEW
81 STOP
82 FIX 2
83 "GAS $="
84 ARCL 01
85 AVIEW
86 STOP
87 FIX 0
88 "MILES="
89 ARCL 07
90 AVIEW
91 STOP
92 FIX 2
93 "FOOD $="
94 ARCL 04
95 AVIEW
96 STOP
97 "BEDS $="
98 ARCL 05
99 AVIEW
100 STOP
101 RCL 01
102 RCL 04
103 +
104 RCL 05
105 +
106 STO 13
107 "TOTAL$="
108 ARCL 13
109 AVIEW
110 STOP
111 "GOING ON ?"
112 AVIEW
113 PSE
114 " 0=NO"
115 PROMPT
116 X=0?
117 GTO D
118 GTO A

119♦LBL D
120 "END"
121 AVIEW
122 STOP
123 .END.
```

Fig. 10-16. The Travel Expenses program for the HP-41C.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter gallons of fuel on board
2. Press **A**
3. Enter cost of gas per gallon
4. Press **B**
Display: Total cost, gas
Repeat the above steps whenever gas is taken
5. Enter miles driven
6. Press **C**
Repeat 5 and 6 for additional miles
Display: Total miles
7. Enter cost of food
8. Press **D**
Repeat 7 and 8 for more food
Display: Total food costs
9. Enter cost of lodging
10. Press **E**
Repeat 9 and 10 for additional nights lodgings
Display: Total lodging costs
11. Press **2nd E'** (any time)
Display: Total gallons, fuel
12. Press **R/S**
Display: Total cost of gas
13. Press **R/S**
Display: Total miles driven
14. Press **R/S**
Display: Total food costs
15: Press **R/S**
Display: Total lodging costs
16. Press **R/S**
Display: Grand total costs

HP-41C

Cards: 2
Passes: 4
Size: 020
Display: *FUEL ON BD.*
1. Enter fuel on board; **R/S**
Display: *$ PER GAL.?*
2. Enter cost per gallon; **R/S**
Display: Total fuel cost
3. Press **R/S**
Display: *MILES?*
4. Enter miles driven; **R/S**
Display: Total miles driven
5. Press **R/S**
Display: *FUEL + ?*
6. Enter fuel added; **R/S**
Display: *$ PER GAL.?*
7. Enter cost per gal.; **R/S**
Display: Cost of latest fuel purchase
8. Press **R/S**
Display: Cost of fuel, total so far
9. Press **R/S**
Display: *+ MILES?*
10. Enter additional miles driven; **R/S**
Display: Total miles so far
11. Press **R/S**
Display: *GOING ON?*, then *0=NO*
If you are going on without stopping to eat, the above is repeated.
In the event of a food stop:
12. Press **0**, then **R/S**
Display: *FOOD $?*
13. Enter food costs; **R/S**
Display: Total food costs so far
14. Press **R/S**
Display: *GOING ON?*, then *0= NO*
If staying for the night:
15. Press **0**, then **R/S**
Display: *BEDS $?*
16. Enter cost of lodging; **R/S**
Display: Total for lodging, so far
17. Press **R/S**
Display: *TOTALS:*, then *GAL=* and total gallons purchased

18. Press **R/S**
Display: *GAS*= *$* and total cost; **R/S**
Display: *MILES*= and total; **R/S**
Display: *FOOD*= *$* and total; **R/S**
Display: *BEDS*= *$* and total; **R/S**
Display: *TOTAL*= *$* and grand total

EATING OUT

This program will remove the shock we often feel when the waiter presents us with the bill after an evening out.

The way the program is set up, it will accept a certain amount for the food, another for drinks. It also needs to be told the tax rate (if any) applicable, and the percentage you want to use calculating the tip. The cost of the food and drink is entered in the form of dollars and cents. The tax and tip rates are entered as percentages and fractions of percentages. In order for the computer to use these figures, they must be divided by 100, because percentage rate divided by 100 and the result multiplied by the number of which the percentage is to be figured will produce the desired result, which then has to be added to the principal to come up with the total of the check, the tip, and eventually the total cost of the visit to the restaurant.

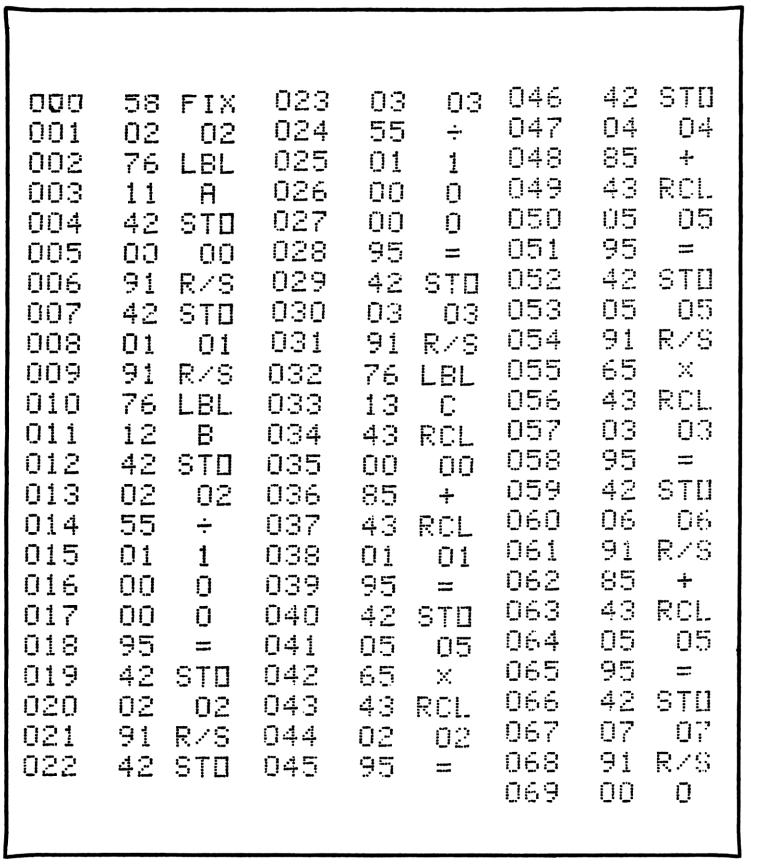
TI-59. We key in the cost of food (Fig. 10-17) and press **A**, then key in the cost of the drinks and press **R/S**. Key in the tax rate, **7.25** and press **B**. Next key in the percent for the tip and press **R/S**. Press **C** and the first amount to show in the display is the total check. The second amount, after pressing **R/S**, again, is the tip. Press **R/S** one more time and the total cost of the evening appears in the display.

HP-41C. The display first identifies the program (Fig. 10-18), *EATING OUT* and then changes to *FOOD COST?* Enter whatever our food cost, say **$23.50** for two. It now asks *DRINKS COST?* We had two each at $2 per, which comes to **$8.00**. The computer now asks *TAX RATE %?* Let's say it's **7.25** percent; key in that figure. The last question is *PERCENT TIP?* The service was adequate but not great, so we'll stick with the standard **15** percent. The display reads *CHECK= $33.78*. Pressing **R/S** produces *TIP= $5.07* and pressing **R/S** again changes the display to *TOTAL= $38.85*.

So much for a simple approach to the subject. Maybe we'd prefer to have a program which would permit us to progressively enter whatever we order, and one which would also keep several accounts separately in case we're, say, two couples, each expected to pay their share. Let's rework the program and see what's involved.

TI-59. With the TI, we simply assign the **A** and **A′** keys to the check for couple A, and assign keys **B** and **B′** to couple B. Key **C** accepts the rate of tax, and key **D** is used to store the percentage to be used for the tip (Fig. 10-19).

Steps 000 through 003 are used to clear stray data out of all registers and to limit the number of displayed decimals to two.



000	58	FIX	023	03	03	046	42	STO
001	02	02	024	55	÷	047	04	04
002	76	LBL	025	01	1	048	85	+
003	11	A	026	00	0	049	43	RCL
004	42	STO	027	00	0	050	05	05
005	00	00	028	95	=	051	95	=
006	91	R/S	029	42	STO	052	42	STO
007	42	STO	030	03	03	053	05	05
008	01	01	031	91	R/S	054	91	R/S
009	91	R/S	032	76	LBL	055	65	×
010	76	LBL	033	13	C	056	43	RCL
011	12	B	034	43	RCL	057	03	03
012	42	STO	035	00	00	058	95	=
013	02	02	036	85	+	059	42	STO
014	55	÷	037	43	RCL	060	06	06
015	01	1	038	01	01	061	91	R/S
016	00	0	039	95	=	062	85	+
017	00	0	040	42	STO	063	43	RCL
018	95	=	041	05	05	064	05	05
019	42	STO	042	65	×	065	95	=
020	02	02	043	43	RCL	066	42	STO
021	91	R/S	044	02	02	067	07	07
022	42	STO	045	95	=	068	91	R/S
						069	00	0

Fig. 10-17. The one-check version of the Eating Out program written for the TI-59.

Steps 004 through 008 are used to store successive food and drink order amount in R_{00}. The identification **SUM 00** (rather than the more often used **STO 00**) in steps 006 and 007 simply commands the computer to add entries to whatever was in R_{00}. Thus, the first entry is added to zero, all following entries are added to the previous sum. Entries are made by keying in the amount and pressing **A** (not **R/S**).

Steps 009 through 013 are a repeat of the above, except that they accumulate the amounts applicable to the second check. (When there is not to be a second check, the B portion can be ignored.)

Steps 014 through 025 accept input of the tax rate and then divide that figure by 100 to prepare it for subsequent calculations.

Steps 026 through 037 do the same with the percentage that will be used later to figure the tip. These entries, tax and tip, do not have to be made until it is time to figure the final amount of the bill.

When it's time to determine the damage, press **2nd A′** and the computer will display the total cost of food and drinks without tax. Then press **R/S** and the total of the check including tax is displayed. Press **R/S** again and the amount of the tip appears in the display. Press **R/S** once more and the display shows the total cost of the evening for couple A. All this is taken care of in steps 038 through 063.

Steps 064 through 089 are an exact duplication of the steps described above, except that the computer uses the amounts charged to couple B by pressing the **B** key after each entry. Therefore, to obtain all the applicable information for couple B, press **2nd B′** and then **R/S** after each display.

While this program might be good exercise, it is not terribly practical on the TI, because you'd be forced to keep the computer turned on throughout the entire evening to avoid listing previously entered data. If the batteries are fully charged and you don't plan to spend much over two hours over dinner, that may be possible, but I'm not convinced it's very practical.

HP-41C. Figure 10-20 shows the changes and additions that had to be made for the expanded program to work properly:

```
01♦LBL "PPG 044"
 02 "EATING OUT"
     03 AVIEW
     04 PSE
 05 "FOOD COST ?"
     06 PROMPT
     07 STO 00
08 "DRINKS COST?"
     09 PROMPT
     10 STO 01
     11 RCL 00
     12 +
     13 STO 01
14 "TAX RATE % ?"
     15 PROMPT
     16 STO 02
     17 100
     18 /
     19 STO 02
     20 RCL 01
     21 *
     22 RCL 01
     23 +
     24 STO 01
25 "PERCENT TIP?"
     26 PROMPT
     27 STO 03
     28 100
     29 /
     30 STO 03
     31 FIX 2
    32 "CHECK=$"
     33 ARCL 01
     34 AVIEW
     35 STOP
     36 RCL 01
     37 RCL 03
     38 *
     39 STO 03
     40 "TIP=$"
     41 ARCL 03
     42 AVIEW
     43 STOP
     44 RCL 01
     45 RCL 03
     46 +
     47 STO 03
   48 "TOTAL=$"
     49 ARCL 03
     50 AVIEW
     51 STOP
     52 END
```

Fig. 10-18. The Eating Out program version designed for a single check; written for the HP-41C.

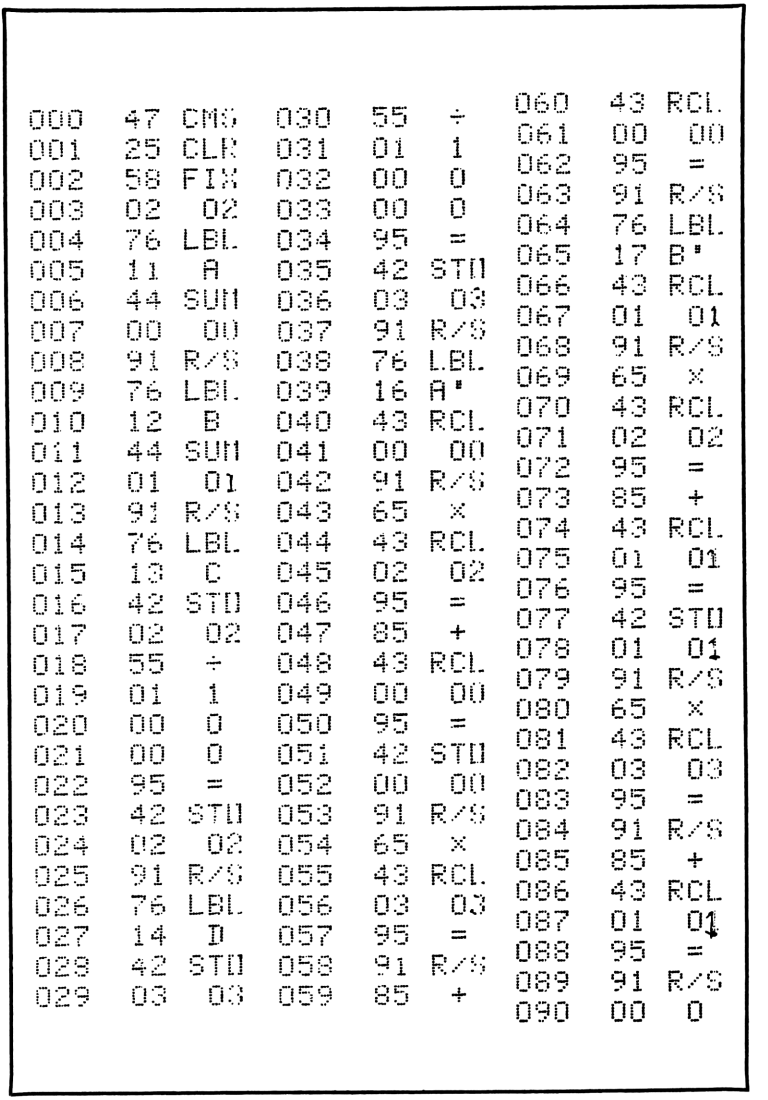


Fig. 10-19. The separate checks version of the Eating Out program for the TI-59.

Step 06: **CF 00** clears a flag which is set later during the running of the program (step 73).

Step 07: A **1** is placed into the X register as assurance that something other than **0** is in the register when we get to the decision-making step (step 13).

Step 08 through 12: Are these to be separate checks? If so, press **0** and then **R/S**. If not simply press **R/S**.

```
01♦LBL "PPG 062"
    02 CLRG
 03 "EATING OUT"
    04 AVIEW
    05 PSE
    06 CF 00
    07 1
08 "SEP. CHECKS ?"
    09 AVIEW
    10 PSE
    11 " 0=YES"
    12 PROMPT
    13 X=0?
    14 GTO 01
    15♦LBL 00
 16 "ORDER $ ?"
    17 PROMPT
    18 STO 01
    19 ST+ 00
    20 RCL 01
    21 X=0?
    22 GTO 02
    23 GTO 00
    24♦LBL 02
25 "TAX RATE % ?"
    26 PROMPT
    27 STO 02
    28 .00
    29 /
    30 STO 02
31 "PERCENT TIP?"
    32 PROMPT
    33 STO 03
    34 100
    35 /
    36 STO 03
    37 FIX 2
    38 FS? 00
    39 GTO 03
    40 RCL 02
    41 RCL 00
    42 *
    43 STO 02
    44 RCL 00
    45 +
    46 STO 01
 47 "CHECK=$"
    48 ARCL 01
    49 AVIEW
    50 STOP
    51 RCL 01
    52 RCL 03
    53 *
    54 STO 03
    55 "TIP=$"
    56 ARCL 03
    57 AVIEW
    58 STOP
    59 RCL 01
    60 RCL 03
    61 +
    62 STO 03
 63 "TOTAL=$"
    64 ARCL 03
    65 AVIEW
    66 STOP
    67 GTO 05
    68♦LBL 01
69 "ORDER A $ ?"
    70 PROMPT
    71 ST+ 04

    72♦LBL 04
    73 SF 00
74 "ORDER B $ ?"
    75 PROMPT
    76 STO 06
    77 ST+ 05
    78 RCL 06
    79 X=0?
    80 XEQ 02
    81 GTO 01
    82♦LBL 03
    83 RCL 04
    84 RCL 02
    85 *
    86 RCL 04
    87 +
    88 STO 04
 89 "CHECK:A=$"
    90 ARCL 04
    91 AVIEW
    92 STOP
    93 RCL 04
    94 RCL 03
    95 *
    96 STO 06
    97 "TIP $"
    98 ARCL 06
    99 AVIEW
   100 STOP
   101 RCL 06
   102 RCL 04
   103 +
   104 STO 04
 105 "TOTAL=$"
   106 ARCL 04
   107 AVIEW
   108 STOP
   109 RCL 05
   110 RCL 02
   111 *
   112 RCL 05
   113 +
   114 STO 05
115 "CHECK:B=$"
   116 ARCL 05
   117 AVIEW
   118 STOP
   119 RCL 05
   120 RCL 03
   121 *
   122 STO 07
   123 "TIP $"
   124 ARCL 07
   125 AVIEW
   126 STOP
   127 RCL 07
   128 RCL 05
   129 +
   130 STO 05
 131 "TOTAL=$"
   132 ARCL 05
   133 AVIEW
   134 STOP

   135♦LBL 05
   136 "END"
   137 AVIEW
   138 STOP
   139 END
```

Fig. 10-20. The separate checks version of the Eating Out program for the HP-41C.

Steps 13 and 14: If **0** has been pressed, the answer to *X = 0?* in step 13 is yes and the command is **GTO 01.** Otherwise, the computer goes on to **LBL 00.**

Steps 15 through 23: **LBL 00** is the section used if there is only one check. It accepts the amounts of each successive order, whether drink or food, and will continue to accept each item until **0** is pressed. When that happens the *X= 0?* in step 21 directs the computer to go on **LBL 02.**

If, after asking whether these are to be separate checks, the answer was in the affirmative, meaning that **0** and **R/S** were pressed, the computer ignored **LBL 00** and went directly to **LBL 01** which is to be found in the step 68 location.

Steps 68 through 71: This section is designed to successively accept amounts for drink and food orders for couple A. It then automatically goes on to the next section, **LBL 04.**

Steps 72 through 81: Here, in step 73, flag 00 is set for reasons which will become apparent when we discuss step 38 in the section **LBL 02.** The section then goes on to accept the costs of each successive drink and food order for couple B, and when it gets to step 79, as long as an order has been entered, it is directed by step 81 to go back to **LBL 01** and again accepts additional orders for each of the couples. Only when couple B keys in a **0,** signifying that no additional orders will be given, does the computer accept the command to go on (or rather backwards) and **XEQ 02.**

Steps 24 through 37: The tax rate and the percentage of tip are established and keyed into the computer. With this information on hand, the computer must be told whether to write one check or two.

Step 38: **FS? 00** asks if flag 00 has been set. If this is a separate-checks situation, flag 00 has been set (in step 73) and the computer will go to the section labeled **LBL 03** in accordance with the command in step 39 (**GTO 03**). If there is to be one check only, the computer will skip that command and go on through the balance of section **LBL 02** all the way to step 67 which tells it to go to **LBL 05,** placing the word *END* into the display.

Steps 82 through 134: This is section **LBL 03** which adds the different orders for the two couples and their separate checks, providing separate totals for the checks, the tip and the overall total cost for each couple. When everything has been displayed, it automatically continues to **LBL 05** and the *END* display.

Program Steps—One Check

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Key in cost of food
2. Press **A**
3. Key in cost of drinks
4. Press **R/S**
5. Key in tax rate
6. Press **B**
7. Key in percent of tip
8. Press **R/S**
9. Press **C**

Display: Total check
10. Press **R/S**
Display: Amount of tip
11. Press **R/S**
Display: Total incl. tip

HP-41C

Cards: 1
Passes: 2
Size: 004
Display: *FOOD COST?*
1. Enter cost of food; **R/S**
Display: *DRINKS COST?*
2. Enter cost of drinks; **R/S**
Display: *TAX RATE %?*
3. Enter tax rate; **R/S**
Display: *PERCENT TIP?*
4. Enter percent of tip; **R/S**
Display: *CHECK*= *$* and amount
5. Press **R/S**
Display: *TIP*= *$* and amount
6. Press **R/S**
Display: *TOTAL* = *$* and amount

Programs Steps—Separate Checks

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Key in cost of food or drink order for party A.
2. Press **A**
3. Key in food or drink cost of party B
4. Press **B**
5. Repeat the above whenever additional drinks or food are ordered. (Do not press **R/S.**)
6. Enter tax rate
7. Press **C**
8. Enter percent for tip
9. Press **D**
10. For party A, press **2nd A′**
Display: Food, drink total without tax
11. Press **R/S**
Display: Total check including tax
12. Press **R/S**
Display: Amount of tip
Press **R/S**
Display: Total for the evening for party A
13. For party B, press **2nd B′**
Display: Food and drink total excluding tax
14. Press **R/S**
Display: Food and drink total including tax
15. Press **R/S**
Display: Amount of tip
16. Press **R/S**
Display: Total for the evening for party B

HP-41C

Cards: 2
Passes: 3
Size: 008
Display: *SEP. CHECKS?*, then *0=YES*
1. Press **0** and **R/S** for separate checks, simply **R/S** for one check
Display: *ORDER A $?*
2. Enter order for party A; **R/S**
Display: *ORDER B $?*
3. Enter order for party B; **R/S**
Display: *ORDER A $?*
4. Enter additional order for party A. If none, enter **0**; **R/S**

Display: *ORDER B $?*

5. Enter additional order for party B. If none, press **0**; **R/S**
Display: *TAX RATE %?*

6. Enter tax rate; **R/S**
Display: *PERCENT TIP?*

7. Enter percent for tip; **R/S**
Display: *CHECK:A= $* and amount including tax

8. Press **R/S**
Display: *TIP= $* and amount

9. Press **R/S**
Display: *TOTAL= $* and amount

10. Press **R/S**
Display: *CHECK:B= $* and amount

11. Press **R/S**
Display: *TIP= $* and amount

12. Press **R/S**
Display: *TOTAL= $* and amount

RATES OF EXCHANGE

This multi-part program could prove useful on any trip to foreign countries. As anyone who has ever left the good old U.S. knows only too well that it can be rather frustrating to figure out what we are paying for that hotel room, the meals in a restaurant, or the gifts we want to bring back to friends at home.

Not only do the rates of exchange fluctuate quite frequently, even when they don't it is difficult to remember what the dollar equivalent is for, say, 4,328 lira in Italy or 1.78 pounds in England ($3.16 U.S.). This program provides the answers in both ways; it will display how much of a foreign currency we will get for a given dollar amount, or it will tell us how much in dollars is the equivalent of any given amount in foreign currency.

The exchange rates applicable to any of the countries included in the program can be changed at a moment's notice by a very simple and short sequence of keystrokes.

For the purpose of the example used here, I have limited the number of countries to 10 for each computer, though, as you will see, the number could easily be increased. The countries I chose are: Canada, Mexico, England, France, Germany, Italy, Switzerland, Japan, Spain and Sweden.

The program is not one program but a string of ten two-part programs combined for convenience.

The program design deals with two separate requirements. One is the actual program which takes care of the calculations, the other concerns the data which represents the exchange rates. In the program reproduced in Figs. 10-21 and 10-22, I have assigned R_{00} to the U.S., R_{01} to Canada, R_{02} to Mexico, R_{03} to England, R_{04} to France, R_{05} to Germany, R_{06} to Italy, R_{07} to Switzerland, R_{08} to Japan, R_{09} to Spain and R_{10} to Sweden. The only other register used is R_{11}, which acts as a temporary storage register during the result calculation.

The program could also be designed to work in one of two ways. We

could set it up so we initially put in an amount in U.S. dollars and the computer calculates the amount of foreign currency we'll get for that amount. When that is displayed, we could enter a different amount of foreign currency and the computer will figure out what that amount represents in U.S. dollars. Or, we could start with the foreign amount, convert that to U.S. dollars, and then convert U.S. dollars to the foreign currency. Either way, the result is the same and the design is a question of personal preference.

In our example I have chosen the first alternative. Let's look at what the programs look like.

TI-59. Figure 10-23 shows the magnetic card which on the side marked with 1, contains the program, while the side marked with an upside-down 4 (for this pass the card has to be turned around) contains the data representing the rates of exchange for the 10 countries. When this card is inserted into the upper slot in the computer, the display shows which key relates to which country, and it also displays the register numbers in which the exchange-rate data are stored.

Figure 10-21 is the printout of the complete program and Fig. 10-24 represents the results for each country after $10 has been entered for the U.S. currency.

Let's take Fig. 10-21 first. Steps 000 through 006 enter the amount of U.S. currency we want to convert. Steps 007 through 10 provide the result for Canada. Steps 11 through 16 reconvert the Canadian currency to U.S. dollars. After that, each labeled section (steps 018 through 32 are LBL B, steps 033 through 47 are LBL D and so on) performs the same calculations for the other countries. At the bottom right-hand corner in the box, we see the data which represents the exchange rates, as of September 1, 1981. They are longer than those shown for the HP-41C because the computer was set to FIX 9, while with the HP it was set to FIX 2, limiting the number of decimal positions to two. Both computers use the complete data with all decimals regardless of whether they are shown or not.

To produce the trace of the computer action shown in Fig. 10-24, start by pressing **1 0 . 0 0 A** which put *10.00* into the display. Press **R/S** which changes the display to *12.20*, the Canadian equivalent. Then press **R/S** again and the display changes back to the U.S. dollar figure. Now press **B** and the display shows the equivalent in Mexican pesos. Pressing **C** shows the British pounds and so on.

Whenever we want to find out what a given amount of foreign currency is in terms of U.S. dollars, simply press the key representing that country, say **2nd C** for Japan and, ignoring what is in display, enter the amount you are interested in, such as 50,000 yen, and then press **R/S** and the display shows that 50,000 yen are equal to $207.50.

If you want to use this while traveling, perhaps to check what the price of a certain dinner is in a restaurant, the TI-59 loses all program and data information whenever it is turned off, and the magnetic card will have to be run through the card-reader slot again before you can use the program.

000	76	LBL	046	95	=	092	91	R/S
001	11	A	047	91	R/S	093	76	LBL
002	58	FIX	048	76	LBL	094	17	B″
003	02	02	049	14	D	095	43	RCL
004	42	STO	050	43	RCL	096	00	00
005	00	00	051	00	00	097	65	×
006	91	R/S	052	65	×	098	43	RCL
007	65	×	053	43	RCL	099	07	07
008	43	RCL	054	04	04	100	95	=
009	01	01	055	95	=	101	91	R/S
010	95	=	056	91	R/S	102	65	×
011	91	R/S	057	65	×	103	43	RCL
012	65	×	058	43	RCL	104	07	07
013	43	RCL	059	04	04	105	35	1/X
014	01	01	060	35	1/X	106	95	=
015	35	1/X	061	95	=	107	91	R/S
016	95	=	062	91	R/S	108	76	LBL
017	91	R/S	063	76	LBL	109	18	C″
018	76	LBL	064	15	E	110	43	RCL
019	12	B	065	43	RCL	111	00	00
020	43	RCL	066	00	00	112	65	×
021	00	00	067	65	×	113	43	RCL
022	65	×	068	43	RCL	114	08	08
023	43	RCL	069	05	05	115	95	=
024	02	02	070	95	=	116	91	R/S
025	95	=	071	91	R/S	117	65	×
026	91	R/S	072	65	×	118	43	RCL
027	65	×	073	43	RCL	119	08	08
028	43	RCL	074	05	05	120	35	1/X
029	02	02	075	35	1/X	121	95	=
030	35	1/X	076	95	=	122	91	R/S
031	95	=	077	91	R/S	123	76	LBL
032	91	R/S	078	76	LBL	124	19	D′
033	76	LBL	079	16	A″	125	43	RCL
034	13	C	080	43	RCL	126	00	00
035	43	RCL	081	00	00	127	65	×
036	00	00	082	65	×	128	43	RCL
037	65	×	083	43	RCL	129	09	09
038	43	RCL	084	06	06	130	95	=
039	03	03	085	95	=	131	91	R/S
040	95	=	086	91	R/S	132	65	×
041	91	R/S	087	65	×	133	43	RCL
042	65	×	088	43	RCL	134	09	09
043	43	RCL	089	06	06	135	35	1/X
044	03	03	090	35	1/X	136	95	=
045	35	1/X	091	95	=	137	91	R/S

Fig. 10-21. Printout of the Exchange Rates program for the TI-59.

138	76	LBL		
139	10	E″	1.219512195	01
140	43	RCL	31.25	02
141	00	00	.5633802817	03
142	65	×	6.25	04
143	43	RCL	2.580645161	05
144	10	10	1369.863014	06
145	95	=	2.222222222	07
146	91	R/S	240.9638554	08
147	65	×	125.	09
148	43	RCL	5.555555556	10
149	10	10		
150	35	1/X		
151	95	=		
152	91	R/S		
153	00	0		

HP-41C. The printer tape of the program is shown in Fig. 10-22. As you can see, it is divided into 15 sections, each headed by an identifying label. The first section, involving steps 01 through 06, might be described as a housekeeping section, and it requires an explanation. Step 01 simply identifies the program to the computer, so that, whenever **XEQ ALPHA E X C H ALPHA** (or the appropriate user-defined key) is pressed, the computer knows which of the stored programs to activate.

Step 2 asks whether flag 00 has been set. If it is not set, the program will continue with step number 04, skipping step number 03. If it has been set, it goes to step number 03 which tells the computer to ignore the next steps and go to the section labeled AA. The flag routine does this: if the program has been recorded on magnetic cards, it requires two cards (four passes) to enter the program. The program is of no value without the prevailing exchange rates. Therefore, as we activate the program it will display: CARD, asking for the card containing the exchange rates. Step 04 (000.011) alerts the computer to the fact that the data card contains information to be stored in registers 00 through 10. The obscure display message in step 05 (*XROM 30.03*) appears because when I attached the printer to my HP-41C I disconnected the card reader. The command which is step 05 is actually **RDTAX**, which stands for Read Data according to X register. This command is one of the many prerecorded program commands stored in the PROM (permanent read-only memory) in the card reader where it occupies the position 30.03. By printing XROM 30.03 the printer is telling us that the card reader is not attached and the step cannot be executed. Once the card reader is installed, step 05 will execute RDTAX and cause the word *CARD* to be displayed to remind us the data has yet to be entered.

```
 01*LBL "EXCH"
   02 FS? 00
   03 GTO "AA"
   04 000.011
 05 XROM 30,07
   06 SF 00
   07*LBL R
   08 ARCL 11
   09 AVIEW
   10 STOP
   11 RTN
   12 XEQ "AA"
   13*LBL C
   14 1/X
   15 *
   16 STO 11
  17 "U.S.$= "
   18 ARCL 11
   19 AVIEW
   20 STOP
   21 RTN
   22*LBL "AA"
   23 FS? 00
  24 GTO "AAA"
   25 000.011
 26 XROM 30,03
   27 SF 00
   28*LBL "AAA"
29 "EXCH. RATES"
   30 AVIEW
   31 PSE
   32 FIX 2
  33 "U.S. $ ?"
   34 PROMPT
   35 STO 00
   36*LBL "A1"
   37 RCL 01
   38 *
   39 STO 11
  40 "CAN.$= "
   41 XEQ B
   42 RCL 01
   43 XEQ C
   44*LBL "A2"
   45 RCL 00
   46 RCL 02
   47 *
   48 STO 11
  49 "MEX.P.="
   50 XEQ B
   51 RCL 02
   52 XEQ C
   53*LBL "A3"
   54 RCL 00
   55 RCL 03
   56 *
   57 STO 11
   58 "POUND="
   59 XEQ B
   60 RCL 03
   61 XEQ C
   62*LBL "A4"
   63 RCL 00
   64 RCL 04
   65 *
   66 STO 11
 67 "F.FRANC="
   68 XEQ B
   69 RCL 04
   70 XEQ C
   71*LBL "A5"
   72 RCL 00
   73 RCL 05
   74 *
   75 STO 11
   76 "MARK="
   77 XEQ B
   78 RCL 05
   79 XEQ C
   80*LBL "A6"
   81 RCL 00
   82 RCL 06
   83 *
   84 STO 11
   85 "LIRA="
   86 XEQ B
   87 RCL 06
   88 XEQ C
   89*LBL "A7"
   90 RCL 00
   91 RCL 07
   92 *
   93 STO 11
 94 "S.FRANC="
   95 XEQ B
   96 RCL 07
   97 XEQ C
   98*LBL "A8"
   99 RCL 00
  100 RCL 08
  101 *
  102 STO 11
  103 "YEN="
  104 XEQ B
  105 RCL 08
  106 XEQ C

  107*LBL "A9"
  108 RCL 00
  109 RCL 09
  110 *
  111 STO 11
  112 "PES.="
  113 XEQ B
  114 RCL 09
  115 XEQ C

 116*LBL "A10"
  117 RCL 00
  118 RCL 10
  119 *
  120 STO 11
  121 "KRON.="
  122 XEQ B
  123 RCL 10
  124 XEQ C
  125 "END"
  126 AVIEW
  127 STOP
  128 .END.
```

```
R00=  0.00
R01=  1.22
R02=  31.25
R03=  0.56
R04=  6.25
R05=  2.58
R06=  1,369.86
R07=  2.22
R08=  240.96
R09=  125.00
R10=  5.56
R11=  0.00
```

Fig. 10-22. The 10-part Rates of Exchange program for the HP-41C and a list of the registers containing the conversion data.

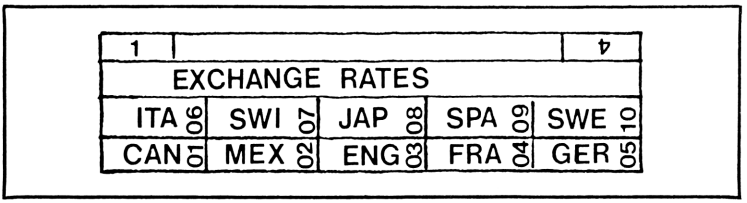


Fig. 10-23. Suggested legend used on the TI-59 program card for the Exchange Rates program.

Once the data representing the current exchange rates has been entered, it is a permanent part of the computer memory and the data card need never be used again (unless the entire program is purposely replaced or erased). Therefore, step 06 sets flag 00 which remains set from now on and tells the computer that steps 04 through 06 should be ignored.

The next section is labeled B and consists of steps 07 through 12. When flag 00 is set, the computer automatically skips this section and goes directly to label AA. When the flag is not set, it will race through the section labeled B, but since there is no particular command which can be executed at this time, the computer simply goes on to the end of the section, step 12, which tells the computer to skip the next section and to go directly to AA.

The next section is labeled C and it is used repeatedly throughout the program.

The section marked LBL AA (steps 22 through 27) is a really good example of what can happen when we write a program and don't keep complete notes on the steps we have taken. This section is simply a repeat, step by step, of the first section, the only difference being that in step 24 it commands the computer to go to the section labeled AAA (GTO AAA). This entire section is totally useless and should be edited out. I left it in to demonstrate how that sort of thing can happen. Leaving that section in the program does no harm, but it reduces the space available for other programs.

We now go to the section marked LBL AAA which consists of steps 28 through 35 and represents the actual start of the program. Step 29 (**EXCH. RATES**) puts the name of the program into the display. Step 32 (**FIX 2**) tells the computer we want only decimal spaces to be displayed at all times. Step 33 (**U.S. $?**) places the request for an amount in dollars into the display. The display will remain until whatever amount we are interested in has been entered, followed by pressing **R/S**. Step 35 tells the computer to store that amount in R_{00}.

The next section, identified by LBL A1, recalls the data stored in R_{01}. This data represents the exchange rate for Canadian dollars. The next step calls for multiplication (x), and since whatever we previously stored in R_{00} has remained in the display (and thus in the X register), the computer knows that it is being commanded to multiply the data in R_{00} by the data recalled from R_{01}. The next step (**STO 11**) commands the computer to

store the result in R_{11}. Step 40 puts the words CAN.$= into a position from which they can be displayed when the computer is ready to do so.

Step 41 tells the computer to execute the section labeled B (**XEQ B**). Now go to LBL B which, in step 08 (**ARCL 11**) tells the computer to recall the data stored in R_{11} and combine it with the previously prepared string of alpha characters (**CAN.$=**). The next step (**AVIEW**) tells the computer to display the combined words and figures, resulting (depending on the amount of U.S. dollars originally entered) in something like *CAN.$=12.20*—you get 12.20 Canadian dollars for a given amount of U.S. dollars. In step 11 the computer is told to go back where it came from, to step 42 in section LBL A1. This step asks that the data stored in R_{01} be recalled once more. The next step tells the computer to execute the section labeled C (XEQ C) which consists of steps 13 through 21.

```
      10.  FIX
           2
     10.00
      10.  STO
           0
     10.00
           R/S
      10.  ×
      10.  RCL
           1
      1.22
1.219512195  =
     12.20
           R/S
12.19512195  RCL
           0
     10.00
      10.  ×
      10.  RCL
           2
     31.25
     31.25  =
    312.50
           R/S
    312.5  RCL
           0
     10.00
      10.  ×
      10.  RCL
           3
      0.56
.5633802817  =
      5.63
           R/S
5.633802817  RCL
           0
     10.00
      10.  ×
      10.  RCL
           4
      6.25
      6.25  =
     62.50
           R/S
     62.5  RCL
           0
     10.00
      10.  ×
      10.  RCL
           5
      2.58
2.580645161  =
     25.81
           R/S
25.80645161  RCL
           0
     10.00
      10.  ×
      10.  RCL
           6
   1369.86
1369.863014  =
   13698.63
           R/S
13698.63014  RCL
           0
     10.00
      10.  ×
      10.  RCL
           7
      2.22
2.222222222  =
     22.22
           R/S
22.22222222  RCL
           0
     10.00
      10.  ×
      10.  RCL
           8
    240.96
240.9638554  =
    2409.64
           R/S
2409.638554  RCL
           0
     10.00
      10.  ×
      10.  RCL
           9
    125.00
     125.  =
   1250.00
           R/S
    1250.  RCL
           0
     10.00
      10.  ×
      10.  RCL
          10
      5.56
5.555555556  =
     55.56
           R/S
```

Fig. 10-24. Trace of the Exchange Rates program run in the TI-59.

LBL C is the section of the program which converts the amount of foreign currency in display into U.S. dollars. This may seem silly, since we just did that in reverse. Before telling the computer to go on when the amount of foreign currency is displayed, we can enter a new amount, and the computer will use that amount rather than the result of the previous computation to figure out what it would be in U.S. dollars. Thus, if with *CAN.$=12.20* in display we now press, say, **26.34 R/S** the display will show *26.34*. We now press **R/S** and the computer responds with a display of *U.S.$= 21.60*. In other words, if all we're interested in is the conversion of a certain amount of foreign currency to U.S. dollars, we can simply ignore the first time the computer requests *U.S. $?* by entering nothing and pressing **R/S.** Then, when some meaningless amount of foreign currency is displayed, we change it by entering the amount we are interested in, press **R/S** again and the dollar equivalent will be displayed instantly.

All the other sections simply follow the same routine for each of the other countries. The table in the upper right-hand corner of Fig. 10-21 shows the rates of exchange used in the example. Figure 10-25 is the printer trace of a few calculations, showing what the computer does to arrive at its conclusions. The one on the left first converts $10 U.S. into the Canadian equivalent. We have then entered 26.34 and the computer displays the equivalent amount in U.S. dollars ($21.60).

On the right, with the $10 U.S. still in the computer memory in R_{00}, we ask for the equivalent in Italian lira. It shows that $10 equals, 13,698.63 lira. Then, retaining this amount, press **R/S** and the computer reminds us that amount represents $10 U.S.

It would be inconvenient if we had to run through all countries to get information about Sweden or Japan or whatever country you need. This problem is solved by assigning each country to a special individual key. The key assignments are shown in the upper right-hand corner of Fig. 10-25. The numbers stand for the row in which the key is located (key 11 is the key on the far left in the top row), and the minus sign means that the shift key must be pressed before using the user-defined key. (Always remember, in order to use user-defined keys, the computer must be in the user mode. When not in that mode, the keys perform their normal function.)

With these key assignments, pressing **A** (I'm using the letters for clarity, even though the computer is not in alpha mode) starts the program at the beginning; pressing **shift A** brings up Canada; **shift B** Mexico; **shift C** England and so on.

To change the exchange rates being used by the computer to perform its calculations, here is what must be done. Say that the exchange rate between the U.S. and Canada changes from 1.22 to 1.43 (which would mean that you get 1.43 Canadian dollars for U.S. $1.00). You press **1 . 4 3 STO 01** and from now on until a new figure is entered and stored in R_{01}, the rate used by the computer for Canada will be 1.43. All that is necessary is to remember which memory register controls which country. To simplify that task as well as the task of remembering which key calls up which country, it

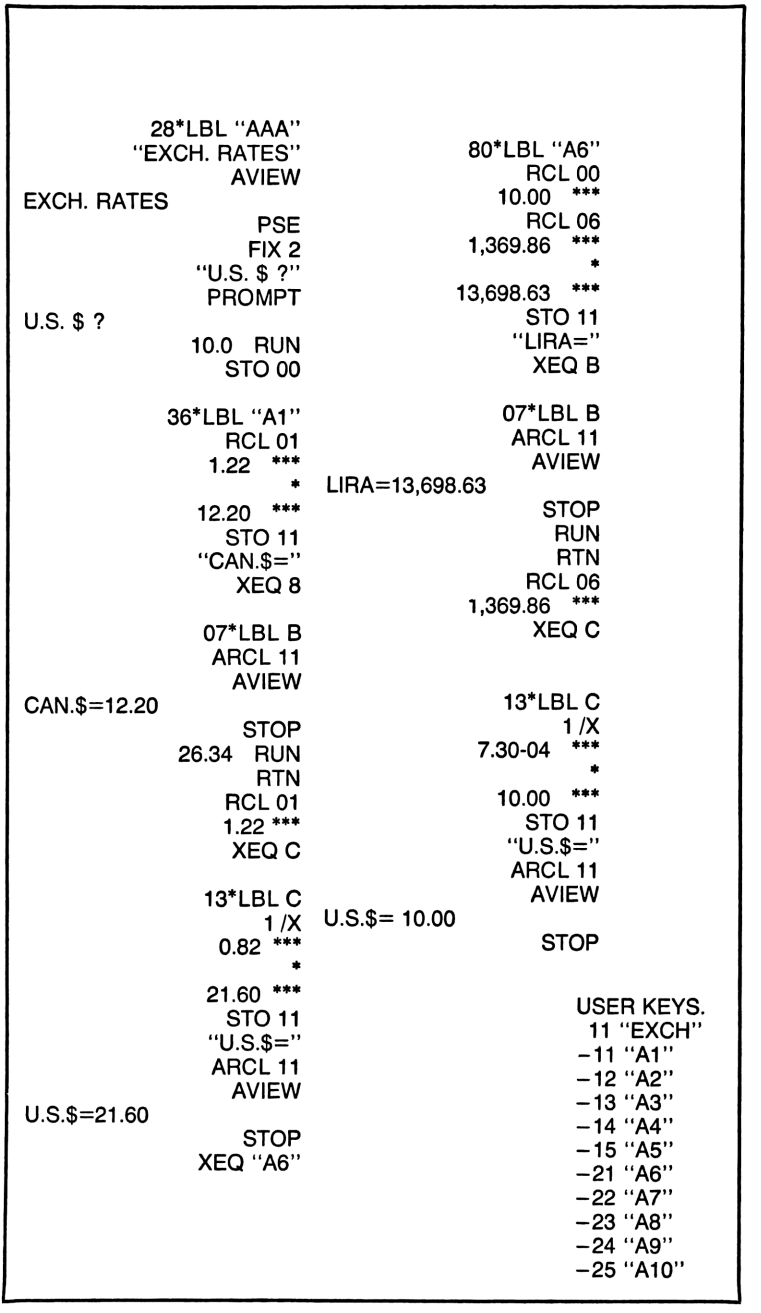


Fig. 10-25. Trace of the Exchange Rate program for Canada and Italy on the HP-41C plus a list of the user-defined keys.

might be a good idea to prepare one of the overlaps available from Hewlett-Packard as shown in Fig. 10-26.

Program Steps

TI-59	HP-41C
Cards: 1 program, 1 data Passes: 1 program, 1 data Partition: 479 59 1. Enter amount in U.S. dollars. 2. Press key for country desired. Display: The equivalent amount in foreign currency. For the reverse data: 3. Enter the amount in foreign currency 4. Press **R/S** Display: The equivalent in U.S. dollars.	Cards: 2 program, 1 data Passes: 4 program, 1 data Size: 012 1. Press **USER** before entering cards into card reader. 2. Press **LN** to start program. Display: *CARD* 3. Feed in data card, 1 pass. Display: *U.S.$?* 4. Key in amount; press key for country of interest (using shift and any of the top 10 keys will call up a given country) Display: Pound= or Yen =, etc. For reverse information: 5. Key in amount in foreign currency. Display: *U.S. $=* and the amount. (Note, after the program has once been run, steps 1 and 3 need not be repeated. Pressing LN will start program from the beginning.)

COLLISION COURSE

This program is a simple version of some very complicated and sophisticated programs designed to predict the courses of vehicles or celestial bodies, predicting the likelihood of collisions.

This program asks the direction of travel of two different vehicles or moving bodies. It then needs to know the speed at which each is traveling and the distance between the two vehicles. With this information at hand, the computer will determine whether the two vehicles will collide or miss each other, and it will display the time until the collision occurs if a collision is imminent. If a collision won't occur the program displays the time by which the two vehicles will miss one another.

Let's see what happens when we run the program in the two computers, using some imaginary data.

TI-59. In converting the program (Fig. 10-27) to the TI-59, I used six of the user-defined keys. Key **A** accepts and stores the direction in degrees for vehicle number one. Key **B** accepts and stores the speed of that vehicle. Key **C** accepts and stores the direction in degrees for vehicle number two.

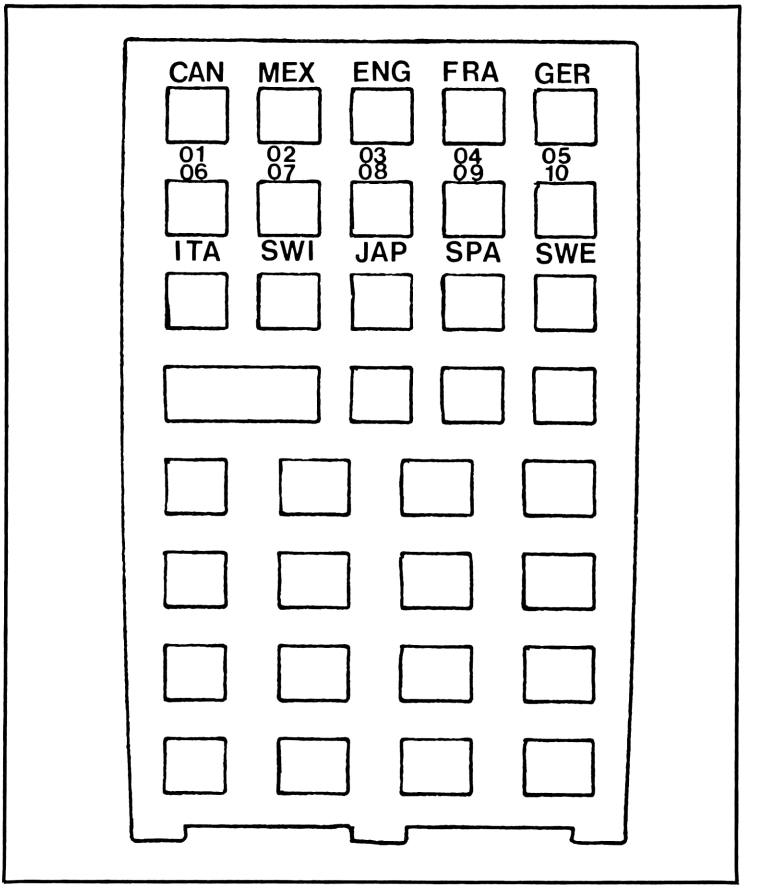


Fig. 10-26. Keyboard overlay for the Exchange Rates program.

Key **D** accepts and stores the speed for that vehicle. Key **E** accepts and stores the information about the distance between the two vehicles.

When all that information has been fed into the computer, pressing **2nd E′** will result in a display of *555555*, meaning the vehicles are not going to collide, or in a flashing *99.99999999*, meaning a collision is inevitable, unless at least one of the vehicles changes direction or speed. Press **CE** to stop the flashing, and then press **R/S**. After a moment the display will show the time in hours, minutes and seconds which is left before the collision will occur.

If you examine the printout (Fig. 10-27) you will see that the keys used for input data, key **A** through **E**, use steps 004 through 028 only. The rest of the program for the computation which is exactly the same as that used for the HP except that it is harder to figure out which steps do what because **STO** and **RCL** each take up two steps, and the occasionally obscure

user-defined labels: **LBL LNX**, step 040, 041; **LBL X^2**, step 048, 049; **LBL** $\sqrt{x}$, step 085, 086; **LBL 1/X**, step 107, 108; **LBL SUM**, step 128, 129; **LBL Y_x**, step 140, 141; **LBL)**, step 153, 154; **LBL SIN**, step 164, 165; **LBL TAN**; step 180, 181; **LBL LOG**, step 230, 231.

HP-41C. Upon initialization the display reads *COLLISION*, the name of the program (Fig. 10-28), then changes to *VEHICLE NO. 1* and changes again to *DEGREE ?* The computer needs to know the vehicle's direction of travel. Enter **250** degrees and press **R/S**. The display asks *SPEED?* Let's assume vehicle number one is traveling at 150 knots; enter **150** and press **R/S**. The display now needs the same information for the second vehicle, and it prompts us by displaying *VEHICLE NO.2* which changes to *DEGREE ?* This time enter **285** and press **R/S**. The question once more is *SPEED?* Let's say both vehicles travel at the same speed, so enter **150** again. After pressing **R/S** the display asks *DIST. APART?* Since we gave the speed in knots give the distance in nautical miles. Let's say they are 50 nm apart. Enter **50** and press **R/S**. The display shows *COLLISION*, then *CHANGE COURSE* and then *TIME = 0.3315*.

If both vehicles continue as they are going right now, they will collide, but they have 33 minutes and 15 seconds in which to change their course or alter their speed.

Let's do the same thing once more using different data. This time Vehicle One is traveling 040 degrees at 145 knots and Vehicle Two is traveling 360 degrees at 155 knots, and the two vehicles are 88 miles apart.

This time the computer displays *SAFE* followed by *TIME = 0.0326*, meaning the two vehicles will miss one another by three minutes and 26 seconds, assuming no change in direction or speed.

In Fig. 10-28 you will notice a reference to left object and right object, instead of vehicle no. 1 and vehicle no. 2. They are the same.

Steps 01 through 05 identify the program by placing *COLLISION* into the display, and step 05 clears away flag 00 which was set later in the program.

Steps 06, 07 and 08 place *LEFT OBJECT* (or *VEHICLE NO.1*) into the display.

Steps 09, 10 and 11 are used to request the information about the direction of travel, and to store that information in R_{00}.

Steps 12 through 15 are necessary because degrees conversion is always complicated because the two degree data which must be dealt with fall on either side of 360 degrees, and calculations won't work unless one or the other is converted to something else. Here the computer asks whether the degree which has been entered is smaller than 90. (Actually, the degree figure has been moved up into the Y register and 90 is placed into the X register and the **x>y?** asks whether the contents of the X register, 90, are greater than the contents of the Y register.) If the answer is yes, the degree figure entered is between 001 and 089, the computer is commanded to **GTO 05**. We find **LBL 05** in steps 35 through 40 and see that it adds 360 to the entered degree figure and then tells the computer to go on to **LBL 07**,

```
000  47 CMS    045  00  00
001  25 CLR    046  35 1/X
002  58 FIX    047  91 R/S
003  00  00    048  76 LBL
004  76 LBL    049  33 X²
005  11  A     050  43 RCL
006  42 STO    051  08  08
007  00  00    052  55  ÷
008  91 R/S    053  43 RCL
009  76 LBL    054  01  01
010  12  B     055  95  =
011  42 STO    056  22 INV
012  01  01    057  88 DMS
013  91 R/S    058  42 STO
014  76 LBL    059  09  09
015  13  C     060  43 RCL
016  42 STO    061  08  08
017  02  02    062  55  ÷
018  91 R/S    063  43 RCL
019  76 LBL    064  03  03
020  14  D     065  95  =
021  42 STO    066  22 INV
022  03  03    067  88 DMS
023  91 R/S    068  42 STO
024  76 LBL    069  10  10
025  15  E     070  32 X:T
026  42 STO    071  43 RCL
027  04  04    072  09  09
028  91 R/S    073  67  EQ
029  76 LBL    074  34 FX
030  10 E'     075  43 RCL
031  43 RCL    076  09  09
032  00  00    077  43 RCL
033  32 X:T    078  10  10
034  09  9     079  86 STF
035  00  0     080  00  00
036  77  GE    081  58 FIX
037  44 SUM    082  00  00
038  61 GTO    083  61 GTO
039  28 LOG    084  23 LNX
040  76 LBL    085  76 LBL
041  23 LNX    086  34 FX
042  43 RCL    087  58 FIX
043  12  12    088  09  09
044  87 IFF    089  05  5

090  55  ÷
091  00  0
092  95  =
093  91 R/S
094  58 FIX
095  04  04
096  43 RCL
097  08  08
098  55  ÷
099  43 RCL
100  01  01
101  95  =
102  22 INV
103  88 DMS
104  42 STO
105  11  11
106  91 R/S
107  76 LBL
108  35 1/X
109  43 RCL
110  09  09
111  88 DMS
112  75  -
113  43 RCL
114  10  10
115  88 DMS
116  95  =
117  50 I×I
118  22 INV
119  88 DMS
120  05  5
121  05  5
122  05  5
123  05  5
124  05  5
125  05  5
126  91 R/S
127  76 LBL
128  44 SUM
129  43 RCL
130  00  00
131  85  +
```

Fig. 10-27. Collision Course for the TI-59.

```
132  03   3     169  12   12    206  75   -
133  06   6     170  43  RCL    207  01   1
134  00   0     171  02   02    208  08   8
135  95   =     172  42  STO    209  00   0
136  42  STO    173  00   00    210  95   =
137  00   00    174  43  RCL    211  94  +/-
138  61  GTO    175  12   12    212  42  STO
139  54   )     176  42  STO    213  06   06
140  76  LBL    177  02   02    214  43  RCL
141  45  Y^     178  61  GTO    215  04   04
142  43  RCL    179  30  TAN    216  55   ÷
143  02   02    180  76  LBL    217  02   2
144  85   +     181  30  TAN    218  95   =
145  03   3     182  43  RCL    219  42  STO
146  06   6     183  00   00    220  07   07
147  00   0     184  75   -     221  55   ÷
148  95   =     185  43  RCL    222  43  RCL
149  42  STO    186  02   02    223  06   06
150  02   02    187  95   =     224  39  COS
151  61  GTO    188  42  STO    225  95   =
152  54   )     189  05   05    226  42  STO
153  76  LBL    190  00   0     227  08   08
154  54   )     191  32  X:T    228  61  GTO
155  43  RCL    192  43  RCL    229  33  X²
156  00   00    193  05   05    230  76  LBL
157  32  X:T    194  22  INV    231  28  LOG
158  43  RCL    195  77   GE    232  43  RCL
159  02   02    196  23  LNX    233  02   02
160  77   GE    197  43  RCL    234  32  X:T
161  38  SIN    198  05   05    235  09   9
162  61  GTO    199  55   ÷     236  00   0
163  30  TAN    200  02   2     237  77   GE
164  76  LBL    201  95   =     238  45  Y^
165  38  SIN    202  85   +     239  61  GTO
166  43  RCL    203  09   9     240  54   )
167  00   00    204  00   0     241  00   0
168  42  STO    205  95   =
```

which consists of steps 16 through 29, used to request and store the speed data for the left object and the degree data for the right object. It then treats the degree data in the same manner as before, adding 360 if it is less than 90. In both instances, if the entered degree data is above 90, the computer simply ignores **LBL 05** and **LBL 06.**

```
 01♦LBL "COL"
 02 "COLLISION"
 03 AVIEW
 04 PSE
 05 CF 00
 06 "LEFT OBJECT"
 07 AVIEW
 08 PSE
 09 "DEGREE ?"
 10 PROMPT
 11 STO 00
 12 ENTER↑
 13 90
 14 X>Y?
 15 GTO 05
 16♦LBL 07
 17 "SPEED ?"
 18 PROMPT
 19 STO 01
 20 "RIGHT OBJECT"
 21 AVIEW
 22 PSE
 23 "DEGREE ?"
 24 PROMPT
 25 STO 02
 26 ENTER↑
 27 90
 28 X>Y?
 29 GTO 06
 30♦LBL 08
 31 "SPEED ?"
 32 PROMPT
 33 STO 03
 34 GTO 09
 35♦LBL 05
 36 RCL 00
 37 360
 38 +
 39 STO 00
 40 GTO 07
 41♦LBL 06
 42 RCL 02
 43 360
 44 +
 45 STO 02
 46 GTO 08
 47♦LBL 09
 48 RCL 00
 49 ENTER↑
 50 RCL 02
 51 X>Y?
 52 GTO 10
 53 GTO 11
 54♦LBL 10
 55 RCL 00
 56 STO 12
 57 RCL 02
 58 STO 00
 59 RCL 12
 60 STO 02
 61♦LBL 11
 62 "DIST. APART ?"
 63 PROMPT
 64 STO 04
 65 RCL 00
 66 RCL 02
 67 -
 68 STO 05
 69 0
 70 ENTER↑
 71 RCL 05
 72 X<=Y?
 73 GTO 01
 74 RCL 05
 75 2
 76 /
 77 90
 78 +
 79 180
 80 -
 81 CHS
 82 STO 06
 83 RCL 04
 84 2
 85 /
 86 STO 07
 87 RCL 06
 88 COS
 89 /
 90 STO 08
 91 GTO 02
 92♦LBL 01
 93 " SAFE"
 94 AVIEW
 95 FS? 00
 96 GTO 04
 97 STOP
 98♦LBL 04
 99 RCL 09
100 HR
101 RCL 10
102 HR
103 -
104 ABS
105 HMS
106 STO 12
107 "TIME="
108 ARCL 12
109 AVIEW
110 STOP
111♦LBL 02
112 RCL 08
113 RCL 01
114 /
115 HMS
116 STO 09
117 RCL 08
118 RCL 03
119 /
120 HMS
121 STO 10
122 ENTER↑
123 RCL 09
124 X=Y?
125 GTO 03
126 VIEW 09
127 PSE
128 VIEW 10
129 PSE
130 SF 00
131 GTO 01

132♦LBL 03
133 "COLLISION"
134 AVIEW
135 PSE
136 "CHANGE COURSE"
137 AVIEW
138 PSE
139 RCL 08
140 RCL 01
141 /
142 HMS
143 STO 11
144 FIX 4
145 "TIME="
146 ARCL 11
147 AVIEW
148 STOP
149 END
```

Fig. 10-28. Collision Course for the HP-41C.

Steps 30 through 34 enter the speed data for the right vehicle. At that point the computer is told to **GTO 09**.

Steps 47 through 53 constitute **LBL 09**. Here the computer does some internal housekeeping to arrange things in an easy-to-use way. It compares the degree information for the two vehicles; it could happen that the information for the right vehicle was entered before that for the left vehicle. If that is the case, the computer goes on to **LBL 10** and in steps 54 through 60 reverses the storage registers for the degree data. If it is not the case, the computer goes on to **LBL 11**, ignoring **LBL 10**.

Steps 61 through 91 are **LBL 11** and contain not only the request for the information dealing with the distance between the two vehicles, but also a large portion of the calculation. Steps 65 through 68 deduct one of the degree figures from the other, storing the result in R_{05}. Step 69 places a 0 into the X register and step 70 moves that 0 to the Y register. We now recall the data stored in R_{05} and compare it to 0, asking **x≤y?** or is the data stored in R_{05} and currently in the X register equal to or smaller than zero? If the answer is yes, the two vehicles are on diverging rather than converging courses, and the computer is told to **GTO 01** which consists of **LBL 01** and steps 92 through 97, putting the word *SAFE* into the display.

Steps 74 through 91, the balance of **LBL 11**, calculates the distance to the point of possible collision, based on the direction of movement of the two vehicles and the distance between them.

Steps 98 through 110, which constitute **LBL 04**, steps 111 through 131 representing **LBL 02**, and steps 139 through 143, the latter section of **LBL 03**, are used to figure the time it will take each of the vehicles to get to the point their paths are predicted to cross. In steps 126 through 129 two time figures are briefly placed into the display, each representing the time which either of the vehicles needs to get to the points where the two directions cross. This data is then replaced by the time in hours, minutes and seconds, by which the two vehicles will miss one another.

Program Steps

TI-59

Cards: 1
Passes: 2
Partition: 479 59

1. Enter direction in degrees for object 1
2. Press **A**
3. Enter speed, object 1
4. Press **B**
5. Enter direction in degrees for object 2
6. Press **C**
7. Enter speed, object 2
8. Press **D**
9. Enter distance between the two objects
10. Press **E**
11. Press **2nd E′**
    Display: Either *555555*, in which case there will be no collision; or *99.999999 99* and flashing, in which case a collision is indicated
12. Press **R/S**
    Display: Time by which objects will miss, or time to collision

Cards: 2
Passes: 3
Size: 013
Display: *LEFT OBJECT*, then *DEGREE?*
1. Enter direction of travel; **R/S**
Display: *SPEED?*
2. Enter speed; **R/S**
Display: *RIGHT OBJECT*, then *DEGREE?*
3. Enter direction of travel; **R/S**
Display: *SPEED?*
4. Enter speed; **R/S**
Display: *DIST. APART?*
5. Enter distance between objects; **R/S**
Display: Either *COLLISION*, then *CHANGE COURSE*, then *TIME:* and time until projected collision.
Or, a brief display of the time for object 1 until possible collision point, then same for object 2, then *TIME* = and the time in hours, minutes and seconds by which they will miss each other.

KEEPING TRACK OF THE MARKET BASKET

With food prices what they are these days, it is not unusual to see a shopper going from aisle to aisle, calculator in hand, figuring the bill and basing decisions about additional purchases on that total.

This program does the same thing, but it performs the functions automatically. You need only enter the price of the item and the display will show the total (minus any applicable taxes), and the computer is then ready to accept the price of the next item. The program is written for New Mexico or any state which, like New Mexico, charges taxes on everything, including food and medicine. In states where some items are taxable, and others are not, you will need to add a way of separating the totals, a simple matter in the TI-59, but it might present a more complicated problem with the HP.

First, let's look at the two programs in detail, and then we'll see what would have to be done to keep non-taxable items in a separate column.

TI-59. All purchases are entered by keying in the amount and pressing **A**. Each time **A** is pressed, the total up to that point is displayed. The tax rate is entered by keying in the percentage figure and pressing **B** (Fig. 10-29). This can be done at any time, prior to the start of shopping, during shopping or afterwards.

When shopping is finished and the tax rate has been entered, pressing key **C** will add the applicable taxes to the previous total and display the amount of the final bill.

HP-41C. After identifying the program (Fig. 10-30) with *MARKET BASKET* in the display, it asks *ITEM COST $* Key in the cost of the first item picked up, say, **$2.89** and press **R/S**. The display shows *2.89* and then changes to *ITEM COST $?* again. This time add the next purchase, say **$5.29**, and press **R/S**. The display shows *8.18*, the total of the purchases so far, and again changes to *ITEM COST $?* It will continue to repeat this until we key in **0 and R/S** when it goes to display of *TOTAL = $8.18*. Press **R/S** again and the display asks *TAX RATE %?* The rate here in Santa Fe is

000	47	CMS	014	12	B	028	43	RCL
001	25	CLR	015	42	STO	029	00	00
002	58	FIX	016	01	01	030	95	=
003	02	02	017	91	R/S	031	42	STO
004	76	LBL	018	76	LBL	032	02	02
005	11	A	019	13	C	033	91	R/S
006	44	SUM	020	55	÷	034	85	+
007	00	00	021	01	1	035	43	RCL
008	43	RCL	022	00	0	036	00	00
009	00	00	023	00	0	037	95	=
010	91	R/S	024	95	=	038	42	STO
011	61	GTO	025	42	STO	039	02	02
012	11	A	026	01	01	040	91	R/S
013	76	LBL	027	65	×	041	00	0

Fig. 10-29. The short version of the Market Basket program for the TI-59.

four and a quarter percent, so key in **4.25** and press **R/S.** In a moment the display reads *BILL= $8.53* having added the applicable amount of tax to the total of the purchases.

This is what takes place inside the computer: Steps 01 through 06 identify the program and limit the displayed decimals to two. Step 07 identifies **LBL 00** as the section which records purchases, adds them and then repeats itself indefinitely. Step 08 places a zero into the X register.

```
 01♦LBL "PPG 046"
      02 CLRG
03 "MARKET,BASKET"
      04 AVIEW
      05 PSE
      06 FIX 2
      07♦LBL 00
      08 0
09 "ITEM COST $?"
      10 PROMPT
      11 STO 02
      12 ST+ 00
      13 VIEW 00
      14 PSE
      15 RCL 02
      16 X=0?
      17 GTO 01
      18 GTO 00
      19♦LBL 01
     20 "TOTAL=$"
      21 ARCL 00
      22 AVIEW
      23 STOP
24 "TAX RATE % ?"
      25 PROMPT
      26 STO 01
      27 100
      28 /
      29 STO 01
      30 RCL 00
      31 *
      32 RCL 00
      33 +
      34 STO 00
      35 "BILL=$"
      36 ARCL 00
      37 AVIEW
      38 STOP
      39 "END"
      40 AVIEW
      41 STOP
      42 END
```

Fig. 10-30. The simple version of the Market Basket program for the HP-41C.

This is not a necessary step, it causes the computer to go to **LBL 01** instead of **LBL 00** if the shopper simply presses R/S after the shopping has been completed. Because of that 0 in the X register, regardless of whether you press **0** and then **R/S** or simply **R/S** to indicate you want the display of the final bill, the computer will go on to the end of the program.

Steps 09 through 14 place the last purchase into R_{02} and add it to what has previously been stored in R_{00}. It then recalls the purchase into the X register (in step 15) to permit the computer to compare it to zero in the next step. In steps 16 through 18 the computer compares zero to the X register. If that, too, is zero, the computer will go to **LBL 01**. If the contents of the X register are not zero, it will automatically return to the beginning of section **LBL 00**.

Steps 19 through 23 identify the section as **LBL 01** and once more place the total of the bill, not including taxes, into the display. Steps 24 and 25 enter the tax rate and store it in R_{01}. Having received that information, the computer goes on to the next steps.

Steps 26 through 34 convert the tax rate, entered in the form of percentages, into decimal format for subsequent calculations. It then multiplies the total purchases by the tax rate and adds the result to the previous total. Steps 35 through the end place the final bill into the display and, if **R/S** is pressed again, place the word *END* into display.

If you live in a state where some items are taxed and others are not, or where food might be taxed at a different rate than other items, you must alter the program to keep the two types of items separate.

With the TI-59 that is simple, requiring a minimum of changes in the program shown in Fig. 10-29. Place the non-taxable items in a separate section controlled by key **D** and then use key **E** to produce the grand total. Figure 10-31 shows the program steps involved. Steps 042 through 052 accumulate non-taxable items and successively display the total of those purchases. Then, steps 053 through 063 add the total of the non-taxed items to the total, including tax, of the taxable items, and the grand total of the final bill will appear in the display.

With the HP-41C, it is a bit more complicated. This is the method which appears to me to be the least complicated solution (Fig. 10-32).

I have tried to keep the changes to a minimum. As you can see in Fig. 10-32, the first change is in steps 10 through 12. Step 12 tells the user *NO TAX = LN*, which means that if the item is not taxable, press *LN* (the key in the upper right-hand corner) instead of **R/S**. That key has been assigned to activate a new section identified as **LBL AA** and the computer must be in user mode for this to function.

By pressing **LN** after entering the amount of a purchase, the computer goes to **LBL AA**, steps 50 through 54. The amount is stored in its own register, R_{04}, and any additional amount is automatically added to whatever is stored in that register. Steps 52 and 53 cause that total to be displayed for a moment before the computer asks *ITEM COST $?* followed by *NO TAX = LN*.

```
000  47  CMS     021  01   1      042  14  D
001  25  CLR     022  00   0      043  44  SUM
002  58  FIX     023  00   0      044  03  03
003  02  02      024  95   =      045  43  RCL
004  76  LBL     025  42   STO    046  03  03
005  11  A       026  01   01     047  91  R/S
006  44  SUM     027  65   ×      048  61  GTO
007  00  00      028  43   RCL    049  14  D
008  43  RCL     029  00   00     050  76  LBL
009  00  00      030  95   =      051  15  E
010  91  R/S     031  42   STO    052  43  RCL
011  61  GTO     032  02   02     053  02  02
012  11  A       033  91   R/S    054  85  +
013  76  LBL     034  85   +      055  43  RCL
014  12  B       035  43   RCL    056  03  03
015  42  STO     036  00   00     057  95  =
016  01  01      037  95   =      058  42  STO
017  91  R/S     038  42   STO    059  04  04
018  76  LBL     039  02   02     060  91  R/S
019  13  C       040  91   R/S
020  55  ÷       041  76   LBL
```

Fig. 10-31. The extended version of the same program, as used by the TI-59.

```
 01♦LBL "PPG 046"        21 GTO 00        41 STOP
      02 CLRG            22♦LBL 01        42 RCL 00
03 "MARKET,BASKET"      23 "TOTAL=$"     43 RCL 04
      04 AVIEW           24 ARCL 00       44 +
      05 PSE             25 AVIEW         45 STO 04
      06 FIX 2           26 STOP          46 "PAY=$"
      07♦LBL 00      27 "TAX RATE % ?"    47 ARCL 04
      08 0               28 PROMPT        48 AVIEW
 09 "ITEM COST $?"       29 STO 01        49 STOP
      10 AVIEW           30 100           50♦LBL "AA"
      11 PSE             31 /             51 ST+ 04
 12 "NO TAX = LN"        32 STO 01        52 VIEW 04
      13 PROMPT          33 RCL 00        53 PSE
      14 STO 02          34 *             54 GTO 00
      15 ST+ 00          35 RCL 00        55 "END"
      16 VIEW 00         36 +             56 AVIEW
      17 PSE             37 STO 00        57 STOP
      18 RCL 02          38 "BILL=$"      58 END
      19 X=0?            39 ARCL 00
      20 GTO 01          40 AVIEW
```

Fig. 10-32. The extended Market Basket program for the HP-41C.

The other change is found in steps 42 through 49. Here the computer adds the total of the taxable items, including the tax, to the total of the non-taxable items to tell the purchaser the amount he or she must pay by displaying *PAY= $55.78* (or whatever).

I might point out that the shopper can call up the accumulated totals of either column at any time by simply pressing **RCL 00** for the total of the taxable items up to that point (excluding the tax), or by pressing **RCL 04** for the total of the non-taxable items up to that point. When the final amount has been displayed, pressing **RCL 00** will put the total of the taxable items, including the tax, into the display, and pressing **RCL 04** will put the total of the non-taxable items into the display.

Program Steps (Simple Version)

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter purchase amount
2. Press **A**
Display: Total purchases
Repeat for each item purchased
3. Enter tax rate
4. Press **B**
5. Press **C**
Display: Total tax amount
6. Press **R/S**
Display: Total with tax

HP-41C

Cards: 1
Passes: 1
Size: 002
Display: *ITEM COST $?*
1. Enter cost of item; **R/S**
Display: Total purchase, then again: *ITEM COST $?*
2. Repeat for each item.When last item has been entered, press **0** and **R/S**
Display: *TOTAL=$* and total purchases without tax
3. Press **R/S**
Display: *TAX RATE %?*
4. Enter tax rate; **R/S**
Display: *BILL=$* total bill including tax

Program Steps (Extended Version)

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter cost of item
2. If taxable, press **A** if not, press **D**
3. Repeat the above until all purchases have been made
4. Enter tax rate
5. Press **B**
6. Press **C**
Display: Total tax
7. Press **R/S**
Display: Total cost of taxable items including tax
8. Press **E**
Display: Total bill for taxable and non-taxable items combined.

HP-41C

Cards: 1
Passes: 2

Size: 005
Display: *ITEM COST $?* and then *NO TAX= LN*
1. Enter cost of item. If taxable, press **R/S**; if not taxable press **LN** (top row, key 5)
Display: *ITEM COST $?* and then *NO TAX= LN*
2. When the last purchase has been entered, press **0** and **R/S**
Display: *TOTAL= $* and sum of taxable purchases
3. Press **R/S**
Display: *TAX RATE %?*
4. Enter tax rate; **R/S**
Display: *BILL= $* and total for taxable items, including tax
5. Press **R/S**
Display: *PAY= $* and the total amount of taxable and non-taxable items combined.

TRIP ROUTING

What we have here could be described as an electronic route map. This program is possible only for the HP-41C. One reason for this is because only the HP has a non-volatile memory; you can keep it on the car seat next to you, turn it off and then, when you later turn it on again, the program is where you left it. Another reason is the alphanumeric display available only on the HP. Except for the ability to place actual words into the display at the appropriate moment in the program, the program wouldn't be worth anything.

For this example I have taken a trip from my home in Santa Fe to San Francisco via Moab, Utah, and Reno, Nevada. The program tells us the distance in road miles to the next town or intersection. It gives cumulative information, and tells us when to turn right or left to pick up a new route.

Figure 10-33 is the printout of the program, which involves 254 steps and includes a total of 60 alpha-numeric displays. Figure 10-34 shows what actually happens inside the program when it is being run. Let's walk through it.

When the computer is first turned on it displays *ROUTING*, the name of the program, and then changes to *SANTA FE TO* and then *SAN FRANCISCO*. Pressing **R/S** tells us the initial road to take is *RTE 285 TO* then *POJAQUE* and then *18 MILES*. Start off, pick up Route 285 and head north. When we get to Pojaque push **ON** and **R/S** and the display tells us *ESPANOLA* and *8 MILES*. Once approaching Espanola turn the computer on again and **R/S** tells us to turn *LEFT, ROUTE 84*. Once established on Route 84 and out of Espanola, the computer will tell us *ABIQUIU* and *23 MILES*, *CHAMA* and *50 MILES* and *PAGOSA SPGS* and *35 MILES*. It finally tells us *LEFT, RTE 160*. In this manner it continues to tell us the distance to the next town and to warn us in advance when we should be aware of a change in routes.

By the time we get to Monticello, Utah, (step 80) we're told the total miles driven today are *303* and the computer asks *MOTEL?* The next day,

```
01♦LBL "PPG 048"
02 "ROUTING"
03 AVIEW
04 PSE
05 FIX 0
06 "SANTA FE TO"
07 AVIEW
08 PSE
09 "SAN FRANCISCO"
10 AVIEW
11 STOP
12 "RTE. 285 TO"
13 AVIEW
14 PSE
15 "POJAQUE"
16 AVIEW
17 PSE
18 "18 MILES"
19 AVIEW
20 PSE
21 18
22 STO 00
23 "ESPANOLA"
24 AVIEW
25 PSE
26 " 8 MILES"
27 AVIEW
28 STOP
29 8
30 ST+ 00
31 "LEFT,ROUTE 84"
32 AVIEW
33 STOP
34 "ABIQUIU"
35 AVIEW
36 PSE
37 "23 MILES"
38 AVIEW
39 PSE
40 23
41 ST+ 00
42 "CHAMA"
43 AVIEW
44 PSE
45 "50 MILES"
46 AVIEW
47 PSE
48 50
49 ST+ 00
50 "PAGOSA SPGS."
51 AVIEW
52 PSE
53 "35 MILES"
54 AVIEW
55 PSE
56 35
57 ST+ 00
58 "LEFT,RTE. 160"
59 AVIEW
60 STOP
61 "DURANGO"
62 AVIEW
63 PSE
64 "63 MILES"
65 AVIEW
66 PSE
67 63
68 ST+ 00
69 "CORTEZ"
70 AVIEW
71 PSE
72 "46 MILES"
73 AVIEW
74 STOP
75 46
76 ST+ 00
77 "RIGHT,RTE. 666"
78 AVIEW
79 STOP
80 "MONTICELLO"
81 AVIEW
82 PSE
83 "60 MILES"
84 AVIEW
85 PSE
86 60
87 ST+ 00
88 "TODAY MI.="
89 ARCL 00
90 AVIEW
91 STOP
92 "MOTEL ?"
93 AVIEW
94 STOP
95 "RIGHT,RTE. 163"
96 AVIEW
97 STOP
98 "MOAB"
99 AVIEW
100 PSE
101 "55 MILES"
102 AVIEW
103 PSE
104 55
105 STO 02
106 ST+ 00
107 "CRESCENT JCT"
108 AVIEW
109 PSE
110 "32 MILES"
111 AVIEW
112 PSE
113 32
114 ST+ 02
115 32
116 ST+ 00
117 "LEFT, I-70"
118 AVIEW
119 STOP
120 "SALINA UT."
121 AVIEW
122 PSE
123 "128 MILES"
124 AVIEW
125 STOP
126 128
127 ST+ 02
128 128
129 ST+ 00
130 "JCT. I-15"
131 AVIEW
132 PSE
```

Fig. 10-33. A trip-routing program for the HP-41C.

```
  133 "39 MILES"
    134 AVIEW
    135 STOP
    136 39
    137 ST+ 02
    138 39
    139 ST+ 00
140 "RIGHT, I-15"
    141 AVIEW
    142 STOP
    143 "HOLDEN"
    144 AVIEW
    145 PSE
  146 "44 MILES"
    147 AVIEW
    148 STOP
    149 44
    150 ST+ 02
    151 44
    152 ST+ 00
153 "LEFT,RTE. 26"
    154 AVIEW
    155 STOP
  156 "27 MILES"
    157 AVIEW
    158 PSE
    159 27
    160 ST+ 02
    161 27
    162 ST+ 00
  163 "TO DELTA"
    164 AVIEW
    165 PSE
166 "LEFT,RTE. 50"
    167 AVIEW
    168 STOP
   169 "ELY, NV"
    170 AVIEW
    171 PSE
  172 "135 MILES"
    173 AVIEW
    174 STOP
    175 135
    176 ST+ 02
    177 135
    178 ST+ 00
  179 "TODAY MI.="
    180 ARCL 02
    181 AVIEW
    182 STOP
  183 "TOTAL MI.="
    184 ARCL 00
    185 AVIEW
    186 STOP
   187 "MOTEL ?"
    188 AVIEW
    189 STOP
190 "TO FALLON, NV"
    191 AVIEW
    192 PSE
  193 "260 MILES"
    194 AVIEW
    195 STOP
    196 260
    197 STO 02
    198 260
    199 ST+ 00
   200 "TO I-80"
    201 AVIEW
    202 PSE
  203 "31 MILES"
    204 AVIEW
    205 STOP
    206 31
    207 ST+ 02
    208 31
    209 ST+ 00
210 "LEFT, I-80"
    211 AVIEW
    212 STOP
  213 "TO RENO"
    214 AVIEW
    215 PSE
  216 "33 MILES"
    217 AVIEW
    218 PSE
    219 33
    220 ST+ 02
    221 33
    222 ST+ 00
223 "SACRAMENTO"
    224 AVIEW
    225 PSE
 226 "135 MILES"
    227 AVIEW
    228 STOP
    229 135
    230 ST+ 02
    231 135
    232 ST+ 00
233 "SAN FRANCISCO"
    234 AVIEW
    235 PSE
 236 "92 MILES"
    237 AVIEW
    238 STOP
    239 92
    240 ST+ 02
    241 92
    242 ST+ 00
243 "TODAY MI.="
    244 ARCL 02
    245 AVIEW
    246 STOP
247 "TOTAL MI.="
    248 ARCL 00
    249 AVIEW
    250 STOP
    251 "END"
    252 AVIEW
    253 STOP
    254 END
```

```
01↑LBL "PPG 049"
"ROUTING"
AVIEW
ROUTING
PSE
FIX 0
"SANTA FE TO"
AVIEW
SANTA FE TO
PSE
"SAN FRANCISCO"
AVIEW
SAN FRANCISCO
STOP
RUN
"RTE  285 TO"
AVIEW
RTE. 285 TO
PSE
"POJAQUE"
AVIEW
POJAQUE
PSE
"18 MILES"
AVIEW
18 MILES
PSE
18
STO 00
"ESPANOLA"
AVIEW
ESPANOLA
PSE
" 8 MILES"
AVIEW
 8 MILES
STOP
RUN
8
ST+ 00
"LEFT,ROUTE 84"
AVIEW
LEFT,ROUTE 84
STOP
RUN
"ABIQUIU"
AVIEW
ABIQUIU
PSE
"23 MILES"
AVIEW
23 MILES
PSE
23
ST+ 00
"CHAMA"
AVIEW
CHAMA
PSE
"50 MILES"
AVIEW
50 MILES
PSE
50
ST+ 00
"PAGOSA SPGS."
AVIEW
PAGOSA SPGS.
PSE
"35 MILES"
AVIEW
35 MILES
PSE
35
ST+ 00
"LEFT,RTE. 160"
AVIEW
LEFT,RTE. 160
STOP
RUN
"DURANGO"
AVIEW
DURANGO
PSE
"63 MILES"
AVIEW
63 MILES
PSE
63
ST+ 00
"CORTEZ"
AVIEW
CORTEZ
PSE
"46 MILES"
AVIEW
46 MILES
STOP
RUN
46
ST+ 00
"RIGHT,RTE. 666"
AVIEW
RIGHT,RTE. 666
STOP
RUN
"MONTICELLO"
AVIEW
MONTICELLO
PSE
"60 MILES"
AVIEW
60 MILES
PSE
60
ST+ 00
"TODAY MI.="
ARCL 00
AVIEW
TODAY MI.=303.
STOP
RUN
"MOTEL ?"
AVIEW
MOTEL ?
STOP
RUN
"RIGHT,RTE. 163"
AVIEW
RIGHT,RTE. 163
STOP
RUN
"MOAB"
AVIEW
MOAB
PSE
"55 MILES"
AVIEW
55 MILES
PSE
55
STO 02
ST+ 00
"CRESCENT JCT"
AVIEW
CRESCENT JCT
PSE
"32 MILES"
AVIEW
32 MILES
PSE
32
ST+ 02
32
ST+ 00
"LEFT, I-70"
AVIEW
LEFT, I-70
STOP
RUN
"SALINA UT."
AVIEW
SALINA UT.
PSE
"128 MILES"
AVIEW
128 MILES
STOP
RUN
128
ST+ 02
128
ST+ 00
"JCT. I-15"
```

Fig. 10-34. An example of the trip routing program.

```
                AVIEW
JCT. I-15
                  PSE
            "39 MILES"
                AVIEW
39 MILES
                 STOP
                  RUN
                   39
               ST+ 02
                   39
               ST+ 00
          "RIGHT, I-15"
                AVIEW
RIGHT, I-15
                 STOP
                  RUN
             "HOLDEN"
                AVIEW
HOLDEN
                  PSE
            "44 MILES"
                AVIEW
44 MILES
                 STOP
                  RUN
                   44
               ST+ 02
                   44
               ST+ 00
        "LEFT,RTE. 26"
                AVIEW
LEFT,RTE. 26
                 STOP
                  RUN
            "27 MILES"
                AVIEW
27 MILES
                  PSE
                   27
               ST+ 02
                   27
               ST+ 00
          "TO DELTA"
                AVIEW
TO DELTA
                  PSE
        "LEFT,RTE. 50"
                AVIEW
LEFT,RTE. 50
                 STOP
                  RUN
           "ELY, NV"
                AVIEW
ELY, NV
                  PSE
          "135 MILES"
                AVIEW
135 MILES
                 STOP
                  RUN
                  135
               ST+ 02
                  135
               ST+ 00
          "TODAY MI.="
               ARCL 02
                AVIEW
TODAY MI.=460.
                 STOP
                  RUN
          "TOTAL MI.="
               ARCL 00
                AVIEW
TOTAL MI.=763.
                 STOP
                  RUN
            "MOTEL ?"
                AVIEW
MOTEL ?
                 STOP
                  RUN
        "TO FALLON, NV"
                AVIEW
TO FALLON, NV
                  PSE
          "260 MILES"
                AVIEW
260 MILES
                 STOP
                  RUN
                  260
               STO 02
                  260
               ST+ 00
            "TO I-80"
                AVIEW
TO I-80
                  PSE
          "31 MILES"
                AVIEW
31 MILES
                 STOP
                  RUN
                   31
               ST+ 02
                   31
               ST+ 00
          "LEFT, I-80"
                AVIEW
LEFT, I-80
                 STOP
                  RUN
            "TO RENO"
                AVIEW
TO RENO
                  PSE
          "33 MILES"
                AVIEW
33 MILES
                  PSE
                   33
               ST+ 02
                   33
               ST+ 00
        "SACRAMENTO"
                AVIEW
SACRAMENTO
                  PSE
          "135 MILES"
                AVIEW
135 MILES
                 STOP
                  RUN
                  135
               ST+ 02
                  135
               ST+ 00
      "SAN FRANCISCO"
                AVIEW
SAN FRANCISCO
                  PSE
            "92 MILES"
                AVIEW
92 MILES
                 STOP
                  RUN
                   92
               ST+ 02
                   92
               ST+ 00
          "TODAY MI.="
               ARCL 02
                AVIEW
TODAY MI.=551.
                 STOP
                  RUN
          "TOTAL MI.="
               ARCL 00
                AVIEW
TOTAL MI.=1,314.
                 STOP
                  RUN
              "END"
                AVIEW
END
                 STOP
```

the routine continues until we get to Ely, Nevada, (step 169) when the computer again tells us the daily miles driven, *460*, the total miles driven, *763*, and asks *MOTEL?* On the third day, by the time we get to San Francisco, it displays the total miles for the third day as *551*, followed by the total miles for the trip, *1,314*.

It's a fair amount of work to put such a program into the computer, but en route it is a lot nicer than fighting with road maps which always insist on tearing in just the place we're interested in.

TIRE WEAR

Is there an appreciable difference in gas mileage and speed between driving with new and worn tires? I do a fair amount of driving, so it seemed like a good idea to find out. The depth of the grooves on the average brand-new tire is about ½ inch, which means the difference in diameter between a tire which is new and one which is close to bald is about one inch. A fully-inflated tire for a 14-inch wheel has a diameter of about 26 inches when it is new.

The problem was to design a program that would figure out the difference in actual revolutions of the rear (driving) wheels. In the program reproduced in Figs. 10-35 and 10-36 I have assumed something halfway between new and worn to be the norm, and am comparing the new and worn to the norm and to one another. I must admit that the result is quite startling.

The primary difference between the programs for the computers is that in the TI program a normal speed of 55 mph and a fuel consumption of 15 gph are assumed, and the tire performance is measured against those figures. (The 55-mph entry is in lines 066 and 067 and the 15-mph entry is in steps 106 and 107. It would be a simple matter to change them to fit the driving habits of the operator and the fuel-consumption rate of his car.) In the program for the HP, the operator is expected to enter the speed and fuel-consumption data he feels are applicable to his driving. Otherwise the two programs work quite similarly.

TI-59. Turn the computer on and feed the magnetic card into the lower slot. Press **CLR** and if appropriate, **RST R/S** to clear any stray data out of the computer. Press **25** for the diameter of the worn tire and then press **A**. *25* will be in the display. Now press **26** for the diameter of the new tire, followed by pressing **B**.

Press **C** and the display responds with *889*, representing the number of revolutions per mile of the worn tire. Press **R/S** and the display changes to *855*, apparently agreeing with the findings of its competition. Press **R/S** again and the display shows *55* to remind us of the normal speed we are using. The display changes to *53* for the speed achieved by the worn tire.

Press **R/S**; after a brief *55* reminder, the display shows *57*, the mph figure for the new tire. Press **R/S** again and the display shows *15* as a brief reminder, and then changes to *15.60*, the gas mileage for the new tire. Press **R/S** and after another brief *15* reminder it changes to *14.42*.

```
000  47 CMS   044  95  =    088  66 PAU
001  25 CLR   045  42 STO  089  43 RCL
002  58 FIX   046  03  03  090  08  08
003  00  00   047  43 RCL  091  91 R/S
004  76 LBL   048  04  04  092  58 FIX
005  11  A    049  55  ÷   093  02  02
006  42 STO   050  43 RCL  094  43 RCL
007  00  00   051  03  03  095  05  05
008  91 R/S   052  95  =   096  65  ×
009  76 LBL   053  91 R/S  097  01  1
010  12  B    054  42 STO  098  05  5
011  42 STO   055  06  06  099  95  =
012  01  01   056  65  ×   100  55  ÷
013  91 R/S   057  05  5   101  43 RCL
014  76 LBL   058  05  5   102  06  06
015  13  C    059  95  =   103  95  =
016  43 RCL   060  55  ÷   104  42 STO
017  00  00   061  43 RCL  105  09  09
018  65  ×    062  05  05  106  01  1
019  89  π    063  95  =   107  05  5
020  95  =    064  42 STO  108  66 PAU
021  42 STO   065  07  07  109  66 PAU
022  02  02   066  05  5   110  43 RCL
023  05  5    067  05  5   111  09  09
024  08  8    068  66 PAU  112  91 R/S
025  02  2    069  66 PAU  113  43 RCL
026  00  0    070  43 RCL  114  06  06
027  65  ×    071  07  07  115  65  ×
028  01  1    072  91 R/S  116  01  1
029  02  2    073  43 RCL  117  05  5
030  95  =    074  05  05  118  95  =
031  42 STO   075  65  ×   119  55  ÷
032  04  04   076  05  5   120  43 RCL
033  55  ÷    077  05  5   121  05  05
034  43 RCL   078  95  =   122  95  =
035  02  02   079  55  ÷   123  42 STO
036  95  =    080  43 RCL  124  10  10
037  91 R/S   081  06  06  125  01  1
038  42 STO   082  95  =   126  05  5
039  05  05   083  42 STO  127  66 PAU
040  43 RCL   084  08  08  128  66 PAU
041  01  01   085  05  5   129  43 RCL
042  65  ×    086  05  5   130  10  10
043  89  π    087  66 PAU  131  91 R/S
                           132  00  0
```

Fig. 10-35. The Tire Wear program for the TI-59.

HP-41C. Press **ON** and the user-defined key which starts the program. The display identifies the program with *TIRE WEAR* and then changes to *TIRE: 1 DIAM. ?* Here enter the diameter of a worn tire, say **25** (inches) and press **R/S**. The display now asks for *TIRE:2 DIAM.?*. Key in **26 R/S**, representing the new tire.

The display now tells us the new tire must make 855 revolutions to cover one mile by showing *REV/MI= 855*. Press **R/S** and it changes to *REV/MI= 889* telling us there is a difference of 34 revolutions between the two. Pressing **R/S** results in *MPH?*. Enter **55, R/S**; the display shows *MPH= 57* for the new tire and (after pressing **R/S**) *MPH=53* for the worn tire.

Press **R/S** and the next question is *MPG?* to conform with the TI program, punch in **15, R/S**. The display shows *MPG=15.60* for the new tire and *MPG=14.42* for the worn one.

Some day when I feel ambitious I may add to this program by figuring out the difference in time and money for any distance. It wouldn't be particularly difficult to do; you might like to try it. The necessary steps are these:

1. Arrange to store a distance in terms of miles in some register so you can change it when you want to.

Recall the mph figure for each tire, take the reciprocal by pressing **1/x** and then multiply by the distance figure stored above. Change that readout to the hour:minutes:seconds mode (HP: press **XEQ ALPHA HMS ALPHA**: TI: press **INV 2nd D.MS**), press **FIX 4** and the display will show the hours, minutes and seconds at each of the two speeds.

Recall the mpg figures for each tire. Then again, recall the distance figure initially stored. Take the reciprocal of the mpg figures and multiply by the distance, or take the basic mpg figures and divide the distance figure by them. Be sure to take the computer out of the hours:minutes:seconds mode before this step. (HP: **XEQ ALPHA HR ALPHA**; TI: **2nd D.MS**) and it would also be a good idea to punch in **FIX 0** to do away with all those unwanted decimals. You now have the amount of gas you'll burn with either tire.

4. Enter the average price of gas in some memory register. Recall the previous results (total gallons of gas) and multiply by this figure. Then take the smaller of the two figures and deduct it from the larger one; the result is the effect on your pocket-book of driving a given distance on worn tires.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter diameter of worn tire
2. Press **A**
3. Enter diameter of new tire
4. Press **B**
5. Press **C**
Display: Number of revolutions of worn tire per mile

TI-59

6. Press **R/S**
Display: Ditto for new tire
7. Press **R/S**
Display: *55* followed by mph figure for worn tire
8. Press **R/S**
Display: *55* followed by mph figure for new tire
9. Press **R/S**
Display: *15* followed by gph figure for worn tire
10. Press **R/S**
Display: Ditto for new tire

HP-41C

Cards: 1
Passes: 2
Size: 017
Display: *TIRE:1 DIAM.?*
1. Enter worn tire diameter; **R/S**
Display: *TIRE:2 DIAM?*
2. Enter new tire diameter; **R/S**
Display: *REV/MI=* followed by revolutions per mile for new tire
3. Press **R/S**
Display: *REV/MI=* ditto for worn tire
4. Press **R/S**
Display: *MPH?*
5. Enter mph figure to be used for comparison; **R/S**
Display: *MPH=* and figure for new tire
6. Press **R/S**
Display: *MPH=* ditto for used tire
7. Press **R/S**
Display: *MPG?*
8. Enter mpg figure used for comparison
Display: *MPG=* figure for new tire
9. Press **R/S**
10. *MPG=* figure for used tire

WALKING

Have you ever wondered how long it would take to walk from coast to coast, or any other distance? Finding the answer is very simple. Take the distance covered by each step and the number of steps in a second, perform the necessary multiplication and division and there you have it. Well, while the principle is simple, the arithmetic involved is beyond the average person's ability to figure it out without pencil and paper.

The program assumes the average step, while walking briskly, will cover 30 inches. It also assumes that two steps are taken in each second. Here is how it works:

TI-59. After feeding in the appropriate magnetic program card (Fig. 10-37), enter the distance for which we want an answer, **10** and press **A**. Then press **R/S** and the display shows the number of steps, *21,120*. Pressing **B** produces the minutes it takes to walk that distance, *176*. Pressing **R/S** results in a display of the hours, minutes and seconds, *2.5600* and one final **R/S** turns that into *0.12* days.

HP-41C. The display identifies the program (Fig. 10-38), *WALKING*, then changes to *DISTANCE*. Enter the distance we're interested in, in miles, say **10**. The program starts to compute: a statute mile consists of 5,280 feet. It takes that number and multiplies it by 12 to arrive at the

```
      PRP "PPG 017"
01♦LBL "PPG 017"
      02 CF 00
    03 "TIRE WEAR"
      04 AVIEW
      05 PSE
      06 CLRG
      07 FIX 0
08 "TIRE:1 DIAM. ?"
      09 PROMPT
      10 STO 01
11 "TIRE:2 DIAM. ?"
      12 PROMPT
      13 STO 00
      14 RCL 00
      15 PI
      16 *
      17 STO 02
      18 5820
      19 12
      20 *
      21 STO 04
      22 RCL 02
      23 /
      24 STO 10
      25 STO 11
      26♦LBL A
     27 "REV/MI="
      28 ARCL 10
      29 AVIEW
      30 STOP
      31 FS? 00
      32 GTO B
      33 RCL 01
      34 PI
      35 *
      36 STO 03
      37 RCL 04
      38 RCL 03
      39 /
      40 STO 10
      41 STO 12
      42 SF 00
      43 GTO A
      44♦LBL B
      45 " MPH ?"
      46 PROMPT
      47 STO 13
      48 RCL 12
      49 *
      50 RCL 11
      51 /
      52 STO 07
      53 "MPH="
      54 ARCL 07
      55 AVIEW
      56 STOP
      57 RCL 11
      58 RCL 13
      59 *
      60 RCL 12
      61 /
      62 STO 08
      63 "MPH="
      64 ARCL 08
      65 AVIEW
      66 STOP
      67 " MPG ?"
      68 PROMPT
      69 STO 16
      70 FIX 2
      71 RCL 12
      72 *
      73 RCL 11
      74 /
      75 STO 09
      76 RCL 16
      77 FIX 2
      78 "MPG="
      79 ARCL 09
      80 AVIEW
      81 STOP
      82 RCL 11
      83 RCL 16
      84 *
      85 RCL 12
      86 /
      87 STO 15
      88 FIX 2
      89 "MPG="
      90 ARCL 15
      91 AVIEW
      92 STOP
      93 "END"
      94 AVIEW
      95 STOP
      96 END
```

Fig. 10-36. The Tire Wear program for the HP-41C.

number of inches in a mile. It divides that total by 30, the number of inches covered in each step, and displays the number of steps, *21,120*. Then, when we press **R/S**, the computer performs the computations which transform that figure into the number of minutes it will take to walk 10 miles: *176*.

Pressing **R/S** once more turns that figure into hours, minutes and seconds: *2.5600* (two hours, 56 minutes, no seconds) and one more **R/S** will

```
000  58 FIX     024  76 LBL     048  91 R/S
001  00  00     025  12  B      049  88 DMS
002  76 LBL     026  43 RCL     050  55  ÷
003  11  A      027  01  01     051  02  2
004  42 STO     028  55  ÷      052  04  4
005  00  00     029  02  2      053  95  =
006  91 R/S     030  95  =      054  58 FIX
007  65  ×      031  55  ÷      055  01  01
008  05  5      032  06  6      056  42 STO
009  02  2      033  00  0      057  05  05
010  08  8      034  95  =      058  91 R/S
011  00  0      035  42 STO     059  00  0
012  95  =      036  03  03
013  65  ×      037  91 R/S
014  01  1      038  58 FIX
015  02  2      039  04  04
016  95  =      040  55  ÷
017  55  ÷      041  06  6
018  03  3      042  00  0
019  00  0      043  95  =
020  95  =      044  22 INV
021  42 STO     045  88 DMS
022  01  01     046  42 STO
023  91 R/S     047  04  04
```

Fig. 10-37. The Walking program for the TI-59.

```
01♦LBL "STEP"      19 ARCL 01      37 HMS
 02 "WALKING"      20 AVIEW        38 STO 04
  03 AVIEW         21 STOP         39 "HRS.="
  04 PSE           22 RCL 01       40 ARCL 04
  05♦LBL 00        23 2            41 AVIEW
06 "DISTANCE"      24 /            42 STOP
  07 PROMPT        25 60           43 RCL 04
  08 STO 00        26 /            44 HR
  09 FIX 0         27 STO 03       45 24
  10 5280          28 "MIN.S="     46 /
  11 *             29 ARCL 03      47 STO 05
  12 12            30 AVIEW        48 FIX 1
  13 *             31 STOP         49 "DAYS= "
  14 30            32 RCL 03       50 ARCL 05
  15 /             33 FIX 4        51 AVIEW
  16 STO 01        34 HR           52 STOP
  17 *             35 60           53 GTO 00
  18 "STEPS="      36 /            54 END
```

Fig. 10-38. Walking, as written for the HP-41C.

turn that into days, *0.12*, a figure which becomes important only if we are dealing in really long distances. By the way, those are 24-hour days, so if the computer tells us that it will take 34.2 days to walk from coast to coast, in terms of eight-hours of walking each day that would actually amount to 102.7 days or 3.4 months.

Program Steps

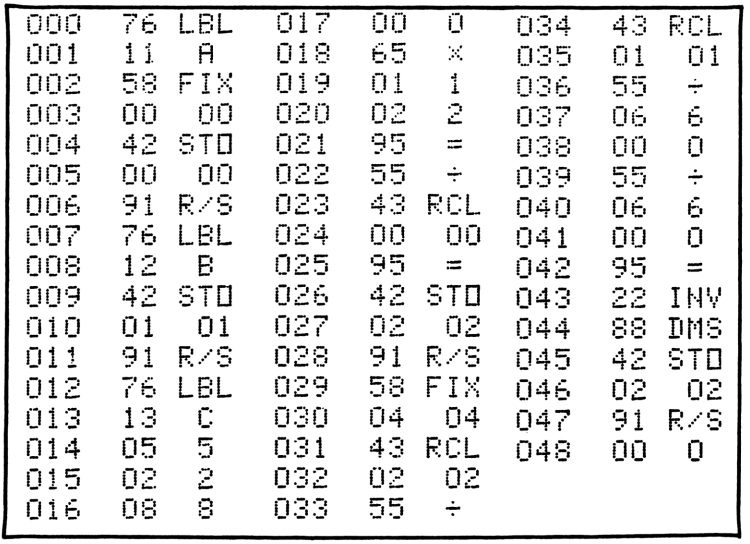
TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 2
Partition: 479 59	Size: 006
1. Enter distance to be walked	Display: *DISTANCE*
2. Press **A**	1. Enter distance to be walked; **R/S**
3. Press **R/S**	Display: *STEPS=* and number
Display: The number of steps	2. Press **R/S**
4. Press **B**	Display: *MIN.S=* and time in minutes and seconds
Display: The number of minutes to walk that distance	3. Press **R/S**
5. Press **R/S**	Display: *HRS.=* and time in hours, minutes and seconds
Display: The above figure in hours, minutes and seconds	4. Press **R/S**
6. Press **R/S**	Display: *DAYS=* and time in terms of 24-hour days and fractions thereof.
Display: The above in days and fractions of days (figured at 24 hours per day)	

RUNNING

It is not possible to use a standard figure for the number of running steps in a second. The length of each step can not be assumed to be standard. Some people take a great many small steps, while others take huge leaps. Therefore, to use this program, you've got to time the number of steps you take in a second, and measure the size of the average step. With that information the computer will compute the number of steps in a mile, and the speed in terms of miles per hour.

TI-59. Key in the length of each step in inches, **24**, and press **A**, which places that figure into R_{00}. Now enter the number of steps per second, **3** and press **B**, placing that number into R_{01}. By pressing **C**, we activate the actual program (Fig. 10-39) and after a brief moment, the display reads *2640*, the number of 24-inch steps needed to cover a mile. Pressing **R/S** will continue the program and the display changes to *0.1440*, the time in hours, minutes and seconds to cover that distance.

If you want to add a section which gives miles per hour, you'd have to press **60 / (RCL 02×100) = STO 03 RS/**. This sequence of key strokes will result in a display of *4.1667* (miles per hour) when **R/S** is pressed once more.



000	76	LBL	017	00	0	034	43	RCL
001	11	A	018	65	×	035	01	01
002	58	FIX	019	01	1	036	55	÷
003	00	00	020	02	2	037	06	6
004	42	STO	021	95	=	038	00	0
005	00	00	022	55	÷	039	55	÷
006	91	R/S	023	43	RCL	040	06	6
007	76	LBL	024	00	00	041	00	0
008	12	B	025	95	=	042	95	=
009	42	STO	026	42	STO	043	22	INV
010	01	01	027	02	02	044	88	DMS
011	91	R/S	028	91	R/S	045	42	STO
012	76	LBL	029	58	FIX	046	02	02
013	13	C	030	04	04	047	91	R/S
014	05	5	031	43	RCL	048	00	0
015	02	2	032	02	02			
016	08	8	033	55	÷			

Fig. 10-39. The Running program written for the TI-59.

HP-41C. After identifying the program (Fig. 10-40) with *RUNNING*, the display asks *STEP INCHES?* Let's enter **24**. Pressing **R/S** brings the request for *STEPS/SECOND*, here we might enter **3**. **R/S** results in *STEPS/MI = 2.640* and another **R/S** changes that to **MI/TIME** = *0.1440*, meaning it takes 14 minutes and 40 seconds to jog a mile at that rate. You could add one more step: multiply this last result by 100 and you'll have the minutes and seconds in the display. Store this figure in R_{03}. Key in **60** and **RCL 03** / and the resulting display, *4.1667* is the speed in terms of miles per hours.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 003
1. Enter length of average step in inches	Display: *STEP INCHES?*
2. Press **A**	1. Enter length of average step in inches; **R/S**
3. Enter number of steps per second	Display: *STEPS/SECOND*
4. Press **B**	2. Enter the number of steps per second; **R/S**
5. Press **C**	Display: *STEPS/MI* = and the number of steps used to run a mile
Display: Number of steps per mile	3. Press **R/S**
6. Press **R/S**	Display: *MI/TIME* = and the time to run a mile in hours, minutes and seconds.
Display: Time to run one mile in hours, minutes and seconds	

```
    01♦LBL "RUN"        17 STO 02        33 ARCL 02
    02 "RUNNING"     18 "STEPS/MI="      34 AVIEW
     03 AVIEW           19 ARCL 02       35 STOP
     04 PSE             20 AVIEW         36 "END"
05 "STEP INCHES?"       21 STOP          37 AVIEW
     06 PROMPT          22 FIX 4         38 STOP
     07 STO 00          23 RCL 02        39 END
     08 FIX 0           24 RCL 01
09 "STEPS/SECOND"       25 /
     10 PROMPT          26 60
     11 STO 01          27 /
     12 5280            28 60
     13 12              29 /
     14 *               30 HMS
     15 RCL 00          31 STO 02
     16 /            32 "MI/TIME="
```

Fig. 10-40. The HP-41C version of the Running program.

TEMPERATURE CONVERSION

For years we have been used to thinking of temperatures in terms of degrees Fahrenheit. Now that the metric system is slowly getting greater acceptance in this country, there will more and more often be a need to convert those degrees Fahrenheit into degrees centigrade (or degrees Celsius, which is the same thing). The standard designations for the two systems are F. for Fahrenheit and C. for centigrade or Celsius.

TI-59. In the program for the TI-59 (Fig. 10-41), the conversion from degrees C. to degrees F. is associated with the **A** key, and the conversion from degrees F. to degrees C. is accomplished by the **B** key. Thus, if we enter **68** and press **B**, *20* will appear in the display. If we now press **A**, the display will return to *68*. It couldn't be simpler.

HP-41C. The way the program is set up for the HP-41C (Fig. 10-42) it first asks *DEGREE C.?*. If you know the temperature in C., it will convert it to degrees F. If that figure is not known, press **R/S** and the computer will ask *DEGREE F.?* Today the outside temperature is 68°F., so punch in **68**, and then press **R/S**. The display reads *DEG. D. = 20*.

To start from the top press **R/S** and the display will once more ask *DEGREE C.?* This time enter **20** and press **R/S**. The display reads *DEG. F. = 68*.

To explain briefly: step 01, **LBL CF**, identifies the program. Step 02, **FIX 0** removes unnecessary decimals from the display. Step 03 places a **0**

000	58	FIX	010	85	+	020	03	3
001	00	00	011	03	3	021	02	2
002	76	LBL	012	02	2	022	95	=
003	11	A	013	95	=	023	55	÷
004	55	÷	014	91	R/S	024	01	1
005	01	1	015	76	LBL	025	93	=
006	93	=	016	12	B	026	08	8
007	08	8	017	42	STO	027	95	=
008	35	1/X	018	00	00	028	91	R/S
009	95	=	019	75	−	029	00	0

Fig. 10-41. Temperature Conversion program for the TI-59.

into the display register (the X register). Steps 04 through 06 place the request for information into the display and then store that information in R_{01}. Step 07 asks whether what is in the X register is equal to 0. If we ignored the first request for information and simply pressed **R/S**, the 0 from step 03 is still in the X register, and the answer, therefore, is yes. In that case the computer is told to go to the section in the program labeled **LBL 00** (the command, step 08, is **GTO 00**). If, on the other hand, we did enter a degree C. figure (any figure other than 0), that figure will have replaced the 0 in the X register and the computer will ignore step 08 (**GTO 00**) and go directly to the next series of steps which convert the data stored in R_{01} into degrees F., which will then be displayed (steps 15 through 17).

If no data was entered, the computer goes directly to **LBL 00** (step 19), produced a request for degrees F. and then computed the degrees C. result, displayed in steps 28 through 30.

```
    01♦LBL "CF"      12 32          23 32
    02 FIX 0         13 +           24 -
    03 0             14 STO 00      25 1.8
04 "DEGREE C. ?"  15 "DEG. F.= "    26 /
    05 PROMPT        16 ARCL 00     27 STO 00
    06 STO 01        17 AVIEW    28 "DEG. C.= "
    07 X=0?          18 STOP        29 ARCL 00
    08 GTO 00        19♦LBL 00      30 AVIEW
    09 1.8        20 "DEGREE F. ?"  31 STOP
    10 1/X           21 PROMPT      32 .END.
    11 /             22 STO 01
```

Fig. 10-42. Converting temperatures using the HP-41C.

Program Steps

TI-59	HP-41C
Cards: 1	Cards: 1
Passes: 1	Passes: 1
Partition: 479 59	Size: 002
1. Enter temperature in degrees C	Display: *DEGREES C.?*
2. Press **A**	1. Enter temperature in degrees C.; **R/S**
Display: Temperature in degrees F	Display: *DEG.F.* = and temperature in degrees F
or	**or**
1. Enter temperature in degrees F	When asked for *DEGREES C.?* simply press **R/S**
2. Press **B**	Display: *DEGREES F.?*
Display: Temperature in degrees C	1. Enter degrees F.; **R/S**
	Display: *DEG.C.* = and degrees C

Chapter 11
Game Programs

This chapter contains programs that are games. Enjoy!

THE SPEED OF RACE HORSES

Have you ever wondered how fast horses run when they compete on the track? Since races are measured in furlongs, and the time is given in minutes, seconds, and decimal fractions of seconds, finding the equivalent in miles per hour is not always simple.

This program does it for you. Enter the number of furlongs and the time it took the horse to run those furlongs in minutes and seconds (format: three minutes and 15.42 seconds is entered as 3.1542) and the computer provides the miles-per-hour figure.

TI-59. The number of furlongs is keyed in followed by **A**. The time in correct format is keyed in, and followed by pressing **R/S**. The display will present the miles-per-hour figure, including one decimal fraction almost instantaneously (Fig. 11-1).

HP-41C. The HP-41C asks the questions: *NO. FURLONGS?* and *TIME, MIN.SEC.?* It then performs the necessary calculations and replies with *MPH = 23.3* (or whatever the result is) (Fig. 11-2).

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Enter number of furlongs
2. Press **A**
3. Enter time as minutes, decimal point, seconds
4. Press **R/S**
Display: Speed in miles per hour

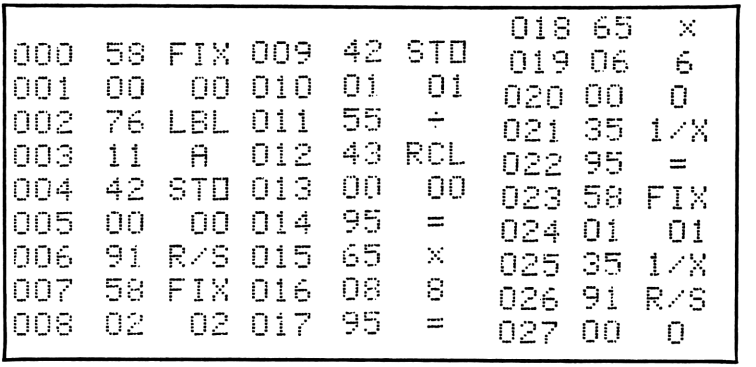


Fig. 11-1. The Race Horses program for the TI-59.

HP-41C

Cards: 1
Passes: 1
Size: 003
Display: *NO. FURLONGS?*
1. Enter number of furlongs; **R/S**
Display: *TIME, MIN.SEC.?*
2. Enter time in minutes, decimal point, seconds; **R/S**
Display: *MPH* = and speed in mph

```
   01♦LBL "PPG 061"      10 PROMPT      19 1/X
    02 "HORSE RACE"      11 STO 01      20 STO 02
       03 AVIEW          12 RCL 00      21 FIX 1
       04 PSE            13 /           22 "MPH="
       05♦LBL 00         14 8           23 ARCL 02
 06 "NO. FURLONGS?"      15 *           24 AVIEW
       07 PROMPT         16 60          25 STOP
       08 STO 00         17 1/X         26 GTO 00
09 "TIME, MIN.SEC.?"     18 *           27 END
```

Fig. 11-2. The Race Horses program on the HP-41C.

GAMBLER'S LOOP

This program determines how many consecutive wins it takes to turn a stake into a fixed amount of winnings or, alternately, how much can be won with a given stake in a fixed number of plays. The program is designed to be used with any odds the operator wants to use, and it will automatically stop when it has reached the predetermined number of plays or the predetermined maximum in terms of winnings.

Figure 11-3 and 11-4 show the two programs. Here is what happens, using a stake of five dollars and odds of two (actually one-to-one, meaning that with each win the player is paid the amount of his or her stake, doubling the stake for the next play).

TI-59. Turn the computer on and feed the magnetic card into the lower slot. Press **CLR**. The program uses the five basic user-defined keys **A** through **E**. **A** is used to record the stake the gambler wants to risk. **B** is the control number representing the maximum number of plays the gambler wants to let his money ride. **C** represents the odds. **D** is the maximum amount of winnings the gambler is interested in winning while continuing to let his original stake ride.

Once these data have been entered into the appropriate positions in the computer's memory, pressing **E** will start the program's execution. Using the same basic data as those we assume with the HP, key in **5** and press **A**, meaning that we're gambling with a $5 stake. Then key in **20** and press **B**, limiting the number of consecutive plays to 20. Key in **2** and press **C** for the odds, and then key in **1,000,000** and press **D** for the maximum

000	25	CLR	030	43	RCL	060	22	INV
001	47	CMS	031	02	02	061	77	GE
002	76	LBL	032	65	×	062	23	LNX
003	11	A	033	43	RCL	063	29	CP
004	58	FIX	034	00	00	064	43	RCL
005	02	02	035	95	=	065	04	04
006	42	STO	036	58	FIX	066	32	X≷T
007	00	00	037	02	02	067	43	RCL
008	91	R/S	038	42	STO	068	01	01
009	76	LBL	039	00	00	069	22	INV
010	12	B	040	66	PAU	070	77	GE
011	58	FIX	041	66	PAU	071	35	1/X
012	00	00	042	66	PAU	072	61	GTO
013	42	STO	043	43	RCL	073	15	E
014	01	01	044	04	04	074	76	LBL
015	91	R/S	045	85	+	075	23	LNX
016	76	LBL	046	01	1	076	58	FIX
017	13	C	047	95	=	077	00	00
018	42	STO	048	58	FIX	078	43	RCL
019	02	02	049	00	00	079	04	04
020	91	R/S	050	42	STO	080	91	R/S
021	76	LBL	051	04	04	081	76	LBL
022	14	D	052	66	PAU	082	35	1/X
023	58	FIX	053	66	PAU	083	58	FIX
024	02	02	054	66	PAU	084	02	02
025	42	STO	055	43	RCL	085	43	RCL
026	03	03	056	00	00	086	00	00
027	91	R/S	057	32	X≷T	087	91	R/S
028	76	LBL	058	43	RCL	088	00	0
029	15	E	059	03	03			

Fig. 11-3. The Gambler's Loop program for the TI-59.

```
    01♦LBL "LLL"
02 "GAMBLER LOOP"
     03 AVIEW
     04 PSE
     05 0
     06 STO 01
  07 "STAKE, $ ?"
     08 PROMPT
     09 STO 00
 10 "CONTROL NO."
     11 PROMPT
     12 STO 02
     13 "ODDS ?"
     14 PROMPT
     15 STO 03
   16 "MAXIMUM ?"
     17 PROMPT
     18 STO 04
   19♦LBL "L1"
     20 RCL 03
     21 ST* 00
     22 FIX 2
     23 VIEW 00
     24 PSE
     25 1
     26 ST+ 01
     27 FIX 0
     28 VIEW 01
     29 PSE
     30 RCL 01
     31 ENTER↑
     32 RCL 02
     33 X=Y?
     34 GTO "L2"
     35 GTO "L3"
   36♦LBL "L2"
     37 VIEW 00
     38 PSE
     39 STOP
   40♦LBL "L3"
     41 RCL 00
     42 ENTER↑
     43 RCL 04
     44 X<=Y?
     45 GTO "L4"
     46 GTO "L1"
   47♦LBL "L4"
     48 VIEW 01
     49 PSE
     50 STOP
     51 .END.
```

Fig. 11-4. The Gambler's Loop program for the HP-41C.

winnings. We're now ready to roll—press **E**. The display shows the progression of the program by displaying *10* and then *1* ($10 is on the table after the first win), followed by *20* and *2, 40* and *3* and so on. The program continues to run until the 18th play, at which point the maximum figure of 1,000,000 has been passed ($1,310,720.00). The controls which cause the computer to get out of the loop and continue with the actual program are contained in steps 055 through 062 and 067 through 074.

HP-41C. Press **ON** and the appropriate user-defined key. The display shows *GAMBLER LOOP* and changes to *STAKE $?* Enter **5** and press **R/S**. The display asks *CONTROL NO.* requesting the maximum number of successive plays during which the gambler wants to let his winnings ride. For fun let's say **20** and then press **R/S**. The display asks *ODDS?* Press **2, R/S**.

The display now wants to know *MAXIMUM?* Let's go completely mad and put in 1,000,000 (without the commas) and press **R/S**. Now the display shows *10* (the amount after the first win), followed by *1* to show that one play has been completed so far. It then goes to *20*, then *2*, then *40*, then *3* and so on. In this case it will stop after the 18th play because the winnings will have exceeded $1,000,000 (actually $1,310,720.00).

By producing a figure equal to or greater than the maximum determined earlier, the computer terminated the loop and continued with the basic program. If it had reached 20 plays before reaching or passing the maximum figure, it would have determined itself to be equal to the control

number and the computer would have come out of the loop and continued to the end of the basic program. These controls are contained in steps 30 through 35 and 41 through 46.

This program is not quite as frivolous as it might seem. It doesn't have to be used to determine winnings in a gambling situation. There are any number of situations in our daily lives when we might want to increase or decrease a certain number by another constant number for a predetermined number of times. Decreasing the original number (the stake) can easily be accomplished by using decimal fractions. Thus, if you want to divide the stake by two with each move, enter the reciprocal of 2 which is 0.5, and the successive results (assuming a stake of 5) are: 2.5, 1.25, 0.63, 0.31, 0.16 and so on (assuming a FIX 2 limit).

Program Steps

TI-59	HP-41C
Cards: 1 Passes: 1 Partition: 479 59 1. Enter the amount to be wagered 2. Press **A** 3. Enter the number of times you want to let the stake ride 4. Press **B** 5. Enter the payoff rate (2 for 2 to 1, etc) 6. Press **C** 7. Enter the maximum winnings to which you want to let the stake ride 8. Press **D** 9. Press **E** 10. Display: Amount of winnings followed by the number of plays until control number (step 3) or maximum (step 7) has been reached	Cards: 1 Passes: 2 Size: 005 1. Enter the amount to be wagered Display: *CONTROL NO.* 2. Enter the number of times you want to let the stake ride Display: *ODDS?* 3. Enter payoff rate (2 for 2 to 1) Display: *MAXIMUM?* 4. Enter maximum winning to which you want to let the stake ride 5. **R/S** starts program Display: The winnings followed by the number of plays until either the control number of the maximum number has been reached.

ROULETTE

This program is another in a series of Las Vegas-type programs. These games are a great way to kill time on a long drive, as long as someone other than the driver is available to push the buttons.

This program not only spins the wheel and tells you whether you've won or lost, it also accepts the bet, and, if you win, tells you how much you

000	47	CMS	044	01	1	088	42	STO
001	25	CLR	045	02	2	089	11	11
002	76	LBL	046	05	5	090	01	1
003	16	A″	047	42	STO	091	01	1
004	01	1	048	25	25	092	04	4
005	00	0	049	02	2	093	42	STO
006	01	1	050	02	2	094	14	14
007	42	STO	051	08	8	095	02	2
008	01	01	052	42	STO	096	01	1
009	02	2	053	28	28	097	07	7
010	00	0	054	02	2	098	42	STO
011	04	4	055	03	3	099	17	17
012	42	STO	056	01	1	100	02	2
013	04	04	057	42	STO	101	02	2
014	01	1	058	31	31	102	00	0
015	00	0	059	01	1	103	42	STO
016	07	7	060	03	3	104	20	20
017	42	STO	061	04	4	105	01	1
018	07	07	062	42	STO	106	02	2
019	02	2	063	34	34	107	03	3
020	01	1	064	02	2	108	42	STO
021	00	0	065	00	0	109	23	23
022	42	STO	066	02	2	110	02	2
023	10	10	067	42	STO	111	02	2
024	02	2	068	02	02	112	06	6
025	01	1	069	01	1	113	42	STO
026	03	3	070	00	0	114	26	26
027	42	STO	071	00	0	115	02	2
028	13	13	072	00	0	116	02	2
029	01	1	073	42	STO	117	09	9
030	01	1	074	49	49	118	42	STO
031	06	6	075	01	1	119	29	29
032	42	STO	076	00	0	120	01	1
033	16	16	077	05	5	121	03	3
034	01	1	078	42	STO	122	02	2
035	01	1	079	05	05	123	42	STO
036	09	9	080	02	2	124	32	32
037	42	STO	081	00	0	125	02	2
038	19	19	082	08	8	126	03	3
039	02	2	083	42	STO	127	05	5
040	02	2	084	08	08	128	42	STO
041	02	2	085	02	2	129	35	35
042	42	STO	086	01	1	130	01	1
043	22	22	087	01	1	131	00	0

Fig. 11-5. The TI-59 version of Roulette.

132	03	3	176	42	STO	220	43	RCL
133	42	STO	177	27	27	221	22	22
134	03	03	178	01	1	222	43	RCL
135	02	2	179	03	3	223	25	25
136	00	0	180	00	0	224	43	RCL
137	06	6	181	42	STO	225	28	28
138	42	STO	182	30	30	226	43	RCL
139	06	06	183	02	2	227	31	31
140	01	1	184	03	3	228	43	RCL
141	00	0	185	03	3	229	34	34
142	09	9	186	42	STO	230	43	RCL
143	42	STO	187	33	33	231	02	02
144	09	09	188	01	1	232	43	RCL
145	01	1	189	03	3	233	48	48
146	01	1	190	06	6	234	43	RCL
147	02	2	191	42	STO	235	05	05
148	42	STO	192	36	36	236	43	RCL
149	12	12	193	05	5	237	08	08
150	02	2	194	05	5	238	43	RCL
151	01	1	195	05	5	239	11	11
152	05	5	196	05	5	240	43	RCL
153	42	STO	197	05	5	241	14	14
154	15	15	198	05	5	242	43	RCL
155	01	1	199	42	STO	243	17	17
156	01	1	200	47	47	244	43	RCL
157	08	8	201	91	R/S	245	20	20
158	42	STO	202	76	LBL	246	43	RCL
159	18	18	203	13	C	247	23	23
160	01	1	204	76	LBL	248	43	RCL
161	02	2	205	23	LNX	249	26	26
162	01	1	206	43	RCL	250	43	RCL
163	42	STO	207	01	01	251	29	29
164	21	21	208	43	RCL	252	43	RCL
165	02	2	209	04	04	253	32	32
166	02	2	210	43	RCL	254	43	RCL
167	04	4	211	07	07	255	35	35
168	42	STO	212	43	RCL	256	43	RCL
169	24	24	213	10	10	257	03	03
170	00	0	214	43	RCL	258	43	RCL
171	42	STO	215	13	13	259	06	06
172	48	48	216	43	RCL	260	43	RCL
173	01	1	217	16	16	261	09	09
174	02	2	218	43	RCL	262	43	RCL
175	07	7	219	19	19	263	12	12

264	43	RCL	312	01	1	360	95	=
265	15	15	313	00	0	361	42	STO
266	43	RCL	314	00	0	362	44	44
267	18	18	315	95	=	363	59	INT
268	43	RCL	316	59	INT	364	42	STO
269	21	21	317	42	STO	365	43	43
270	43	RCL	318	41	41	366	32	X≷T
271	24	24	319	32	X≷T	367	43	RCL
272	43	RCL	320	43	RCL	368	42	42
273	49	49	321	40	40	369	67	EQ
274	43	RCL	322	67	EQ	370	33	X^2
275	27	27	323	33	X^2	371	43	RCL
276	43	RCL	324	43	RCL	372	44	44
277	30	30	325	37	37	373	22	INV
278	43	RCL	326	55	÷	374	59	INT
279	33	33	327	01	1	375	65	×
280	43	RCL	328	00	0	376	01	1
281	36	36	329	00	0	377	00	0
282	22	INV	330	95	=	378	00	0
283	86	STF	331	22	INV	379	95	=
284	00	00	332	59	INT	380	42	STO
285	61	GTO	333	4	STO	381	44	44
286	23	LNX	334	42	42	382	59	INT
287	76	LBL	335	65	×	383	42	STO
288	11	A	336	01	1	384	43	43
289	42	STO	337	00	0	385	32	X≷T
290	38	38	338	00	0	386	43	RCL
291	91	R/S	339	95	=	387	42	42
292	76	LBL	340	42	STO	388	67	EQ
293	12	B	341	42	42	389	33	X^2
294	42	STO	342	43	RCL	390	43	RCL
295	39	39	343	39	39	391	44	44
296	91	R/S	344	59	INT	392	22	INV
297	76	LBL	345	42	STO	393	59	INT
298	15	E	346	43	43	394	65	×
299	42	STO	347	32	X≷T	395	01	1
300	37	37	348	43	RCL	396	00	0
301	55	÷	349	42	42	397	00	0
302	01	1	350	67	EQ	398	95	=
303	00	0	351	33	X^2	399	59	INT
304	00	0	352	43	RCL	400	42	STO
305	95	=	353	39	39	401	43	43
306	59	INT	354	22	INV	402	32	X≷T
307	42	STO	355	59	INT	403	43	RCL
308	40	40	356	65	×	404	42	42
309	43	RCL	357	01	1	405	67	EQ
310	39	39	358	00	0	406	33	X^2
311	55	÷	359	00	0	407	61	GTO

Fig. 11-5. (Continued from page 263.)

408	35	1/X	434	32	X≷T	460	76	LBL
409	76	LBL	435	00	0	461	38	SIN
410	35	1/X	436	67	EQ	462	43	RCL
411	58	FIX	437	38	SIN	463	38	38
412	09	09	438	43	RCL	464	65	×
413	01	1	439	39	39	465	03	3
414	55	÷	440	65	×	466	05	5
415	00	0	441	01	1	467	95	=
416	95	=	442	00	0	468	91	R/S
417	91	R/S	443	00	0	469	76	LBL
418	76	LBL	444	00	0	470	39	COS
419	33	X^2	445	00	0	471	43	RCL
420	58	FIX	446	95	=	472	38	38
421	00	00	447	22	INV	473	65	×
422	43	RCL	448	59	INT	474	01	1
432	47	47	449	32	X≷T	475	07	7
424	66	PAU	450	00	0	476	95	=
425	66	PAU	451	67	EQ	477	91	R/S
426	66	PAU	452	39	COS	478	76	LBL
427	87	IFF	453	61	GTO	479	30	TAN
428	00	00	454	30	TAN	480	43	RCL
429	28	LOG	455	76	LBL	481	38	38
430	43	RCL	456	28	LOG	482	65	×
431	39	39	457	43	RCL	483	08	8
432	22	INV	458	38	38	484	95	=
433	59	INT	459	91	R/S	485	91	R/S
						486	00	0

have won. It also contains a brief recap of the rules of the game, and of the odds and potential payoffs of the different combinations which the player can play.

For those unfamiliar with roulette, it is played on a table numbered from 1 through 36 plus a 0 and a 00. All numbers other than the 0 and 00 are either red or black. A player can bet on red or black, on odd or even, on any of the numbers including 0 and 00, and on any combination of two or four numbers. (At Las Vegas and in other gambling establishments there are several additional betting combinations, but I have limited the program to those listed above).

When betting on red or black, the player wins if a number of the right color comes up. If the other color or a 0 or 00 comes up, he loses. Payoff is 1:1 and the odds are 18:20, meaning there are 18 chances to win and 20 to lose.

When betting odd or even, the same rules apply, except that instead of the right color, the winning number has to be either odd or even.

When betting a combination of four numbers, any four numbers (including 0 and 00) may be chosen. In our game they must be keyed in as seven or

eight digits. Thus, placing a bet on 15, 20, 18 and 6 would look like 15.201806, but it could also be entered as 6.152018. In other words, each number of the bets to the right of the decimal point must consist of two digits. To the left of the decimal point one digit is sufficient. On a four-number bet the player wins if any one of the four numbers comes up. Otherwise it's a loss. The payoff is 8:1 and the odds are 4:38 in favor of the house, naturally.

When betting a combination of two numbers, the same rules apply. A win results when either number comes up. The payoff is 17:1 and the odds are 2:38.

When playing individual numbers, that number must come up. If it does, the payoff is 35:1 and the odds are 1:38.

In actual gambling situations, you can make as many different bets on each turn of the wheel as you wish. In our game that is not provided for.

Now let's play:

TI-59. The program for the TI-59 (Fig. 11-5) uses more than 480 steps as it is (it could probably be edited down to the 480-step limit). Therefore, before the computer will accept the magnetic cards containing the program, it must be repartitioned. Press **5 2nd Op 1 7** and the display will read *559 49*, meaning that there is now room for 560 program steps and 50 data register positions.

Next feed the program cards into the card-reader slot. It will take three passes (two cards) to record and subsequently reuse the complete program.

Once the program has been accepted by the computer, and before playing the first game, press **RST R/S**. There will be several seconds of hesitation, after which the display reads *555555*, to announce the computer is now ready to play the game.

Key in the amount of money you want to wager, say **$5**, and press **A**. Key in where you want to place your bet. Let's put it on red and see what happens. Red is represented by the number 200. Key in **200** and press **B**. Now press **C** and the roulette wheel starts to go 'round and 'round, and it will continue to do so until we press **R/S**. This time pressing **R/S** has resulted in a display of *224*. Now press **E** and after a moment for computing, the display responds with a flickering *555555* to tell us we won. It then changes to a steady *5*, informing us that we won $5.

As with the HP, when playing more than one number, the numbers are entered as first number;decimal point;second, third and and fourth number (15.161718 or 3.040506). Black and odd are entered as 100; red and even as 200; and zero is entered as 0, but 00 is entered in the TI as 1000.

When the player loses, the display responds with a flashing *99.99999999*. When the player wins, the display always shows first a flickering *555555*, which is then replaced by the amount of the winnings (the wager times, 1, 8, 17 or 35).

The game can be played any number of times without using the initialization procedure (**RST R/S**), but the procedure should always be used before the first game after loading the program into the computer.

Figure 11-5 shows how the program is constructed. Steps 000 and 001 clear the computer of any left-over data. Steps 003 and 004 (*LBL A'*) are really unnecessary; they are left over from an earlier version which wouldn't work satisfactorily, but their presence in the program does no harm.

Steps 004 through 192 are used to place the 36 numbers plus 0 and 00 (1000) into registers R_{01} through R_{36} as well as R_{48} and R_{49}, and steps 193 through 200 place 555555 into register R_{47} for future recall.

Step 201, **R/S**, is the place to which the program goes automatically when it is initialized by pressing **RST R/S**.

Steps 202 through 286 represent the roulette wheel, the loop which goes 'round and 'round indefinitely until **R/S** stops it. This loop includes each number, plus a command to clear flag 0 in case it has been set during a previous game (steps 282, 283 and 284).

Steps 287 through 296 accept the bet and information as to where to place it and store that data in R_{38} and R_{39}.

Step 297 through the end of the program (step 485) represent the part that actually does all the work, the computing procedure, which is activated by pressing **E**.

The first thing that happens after pressing **E** is the figure in the display, the winning number, is placed into R_{37} (steps 299 and 300). It is then divided by 100 and that number and the bid are compared to determine if red, black, odd or even have been bet. If so, it determines whether the player won or lost. That determination is made by step 322; in step 323 the computer is told to go to **LBL X^2**, which determines the amount of the winnings.

If the contents of R_{40} and R_{41} are not equal (steps 317 through 322), the computer skips that command (step 323) and goes to the next sequence (steps 324 through 350). Here the computer recalls the winning number as it was in the display, divides it by 100 and then gets rid of the leading 1 or 2 (steps 331, 332), since it has been determined that the bet was not red/black or odd/even. That leading 1 or 2 is no longer needed. The computer compares the remaining number with the number that has been bet, and if it finds they match, it sends the computer again to **LBL X^2**.

Steps 352 through 408 are used to determine whether the player bet on one number, two numbers or four numbers and whether one of those numbers matches the winning number.

Steps 409 through 417 represent **LBL 1/X** which is the section the computer goes to when the player has lost. It creates the display of flashing 9s.

Steps 418 through 485 represent the **LBL X^2** section to which the computer goes when the player has won. It determines the amount of winnings the player is entitled to, and after displaying that flickering 555555 (steps 422 through 426), it displays the amount of winnings.

HP-41C. We have assigned the key in the upper left-hand corner the task of initializing the program, and we have assigned the key in the upper

```
01♦LBL "PPG 034"
  02 " ROULETTE"
     03 AVIEW
     04 PSE
     05 1
06 "SEE RULES ?"
     07 AVIEW
     08 PSE
     09 "  0=YES"
     10 PROMPT
     11 X=0?
     12 GTO C
     13 GTO B

     14♦LBL C
 15 "PAY / ODDS"
     16 AVIEW
     17 PSE
     18 PSE
  19 "RED/BLACK"
     20 AVIEW
     21 PSE
     22 PSE
   23 "ODD/EVEN"
     24 AVIEW
     25 PSE
     26 PSE
27 "1:1 / 18:20"
     28 AVIEW
     29 PSE
     30 PSE
 31 "ANY 4 NO.S"
     32 AVIEW
     33 PSE
     34 PSE
 35 "8:1 / 4:38"
     36 AVIEW
     37 PSE
     38 PSE
 39 "ANY 2 NO.S"
     40 AVIEW
     41 PSE
     42 PSE
43 "17:1 / 2:38"
     44 AVIEW
     45 PSE
     46 PSE
 47 "ANY NUMBER"
     48 AVIEW
     49 PSE
     50 PSE
51 "35:1 / 1:38"
     52 AVIEW
     53 PSE
     54 PSE
 55 "RED:EVEN=200"
     56 AVIEW
     57 PSE
     58 PSE
59 "BLACK:ODD=100"
     60 AVIEW
     61 PSE
     62 PSE
     63 1
 64 "AGAIN? 0=YES"
     65 PROMPT
     66 X=0?
     67 GTO C

     68♦LBL B
     69 CLRG
 70 "PLACE BET $?"
     71 PROMPT
     72 STO 01
  73 "PLACE ON ?"
     74 PROMPT
     75 STO 02
  76 "SPIN WHEEL"
     77 AVIEW
     78 PSE
     79 FIX 0
 80 "R/S TO STOP"
     81 AVIEW
     82 PSE
     83♦LBL A
     84 101
     85 204
     86 107
     87 210
     88 213
     89 116
     90 119
     91 222
     92 125
     93 228
     94 231
     95 134
     96 202
     97 00
     98 105
     99 208
    100 211
    101 114
    102 217
    103 220
    104 123
    105 226
    106 229
    107 132
    108 235
    109 103
    110 206
    111 109
    112 112
    113 215
    114 118
    115 121
    116 224
    117 0
    118 127
    119 130
    120 233
    121 136
    122 GTO A

    123♦LBL "AA"
    124♦LBL 01
    125 STO 07
    126 ENTER↑
    127 1
    128 X>Y?
    129 GTO 02
    130 GTO 03

    131♦LBL 02
    132 RCL 02
    133 ENTER↑
    134 RCL 07
    135 X=Y?
    136 GTO 11
    137 GTO 10

    138♦LBL 03
    139 RCL 02
    140 100
    141 /
    142 FRC
    143 100
    144 *
    145 ENTER↑
    146 0
    147 X=Y?
    148 GTO 00
    149 GTO 04

USER KEYS:
 11 "PPG 034"
 15 "AA"
```

Fig. 11-6. The HP-41C version of Roulette.

```
150♦LBL 00
151 RCL 07
152 100
153 /
154 INT
155 100
156 *
157 ENTER↑
158 RCL 02
159 X=Y?
160 GTO 14
161 GTO 10

162♦LBL 04
163 RCL 07
164 100
165 /
166 FRC
167 100
168 *
169 STO 08

170♦LBL 05
171 RCL 02
172 INT
173 STO 13
174 RCL 02
175 RCL 13
176 -
177 X=0?
178 GTO 06
179 GTO 07

180♦LBL 06
181 RCL 02
182 ENTER↑
183 RCL 08
184 X=Y?
185 GTO 11
186 GTO 10

187♦LBL 07
188 RCL 02
189 100
190 *
191 FRC
192 100
193 *
194 STO 09
195 ENTER↑
196 0
197 X=Y?
198 GTO 08
199 GTO 09
200♦LBL 08
201 RCL 02
202 INT
203 ENTER↑
204 RCL 08
205 X=Y?
206 GTO 12
207 RCL 02
208 FRC
209 100
210 *
211 ENTER↑
212 RCL 08
213 X=Y?
214 GTO 12
215 GTO 10

216♦LBL 09
217 RCL 02
218 INT
219 ENTER↑
220 RCL 08
221 X=Y?
222 GTO 13
223 RCL 02
224 FRC
225 100
226 *
227 INT
228 ENTER↑
229 RCL 08
230 X=Y?
231 GTO 13
232 RCL 09
233 INT
234 ENTER↑
235 RCL 08
236 X=Y?
237 GTO 13

238 RCL 09
239 FRC
240 100
241 *
242 INT
243 ENTER↑
244 RCL 08
245 X=Y?
246 GTO 13
247 GTO 10
248♦LBL 10
249 " YOU LOST"
250 AVIEW
251 STOP
252 GTO 16

253♦LBL 11
254 RCL 01
255 35
256 *
257 STO 00
258 GTO 15

259♦LBL 12
260 RCL 01
261 17
262 *
263 STO 00
264 GTO 15

265♦LBL 13
266 RCL 01
267 8
268 *
269 STO 00
270 GTO 15

271♦LBL 14
272 RCL 01
273 STO 00
274 GTO 15

275♦LBL 15
276 "WIN $ "
277 ARCL 00
278 AVIEW
279 STOP
280 GTO 16

281♦LBL 16
282 1
283 "AGAIN? 0=NO"
284 PROMPT
285 X=0?
286 GTO 17
287 GTO B

288♦LBL 17
289 " END"
290 AVIEW
291 STOP
292 .END.
```

right-hand corner to be used once the winning number has been established. Although the computer will not be in the alpha mode (it must be in the user mode), for simplicity's sake, we'll call the two keys by their alpha identifications: **A** and **E**. (See Fig. 11-6.)

After turning the computer on, press **A** and the display first shows *ROULETTE*, then changes to *SEE RULES?* which then changes to *0=YES*. You may want to be reminded of the rules, payoffs and odds: if you remember them, you'd want to skip this portion of the program. If you press **0** and **R/S**, the rules will be shown. If you simply press **R/S**, the rules will be skipped.

Since this is the first time we're playing, let's look at the rules and press **0** followed by **R/S**. What results is a long series of displays, each kept in view for about a second and a half: *PAY/ODDS* means the display of potential payoffs and the odds will be displayed as payoff-slash-odds: *RED/BLACK; ODD/EVEN; 1:1/18:20; ANY 5 NO.S; 8:1/4:38 ANY 2 NO.S; 17:1/2:38; ANY NUMBER; 35:1/1:38*. The display continues with *RED/EVEN=200; BLACK/ODD=100*, followed by *AGAIN? 0=YES*. If you want to bet on red or on even, enter **200** when it comes time to place your bet. If you want to play black or odd, enter **100**. Also, all numbers other than 0 and 00 will be preceded by a 1 or a 2 when they are displayed as the winning number. Thus, 123 is 23 black, and 204 is 4 red. The *AGAIN* question is simply for those who may have missed some of the foregoing information.

Now press **R/S** and the display tells us *PLACE BET $* meaning the computer wants to know how much we want to bet. Let's be big and throw in $5. We key in **5** and press **R/S**. Now the display asks *PLACE ON?* Let's be cautious and place it on red. Key in **200**. The display says, *SPIN WHEEL*, followed by *R/S TO STOP*. When we're convinced our luck will hold out, press **R/S**. It displays (right now, anyway) *266*. Now press the **E** key and the display announces *WIN $5*.

Okay, so far so good. Let's take our original $5 back and decide to play with the money from the house. Press **R/S** and it asks *AGAIN? 0=NO*. Press **R/S** and we are asked to *PLACE BET $*. We'll play our winnings, $5, and this time we'll place it on four numbers: 15, 16, 17, 18. Key in **15.161718** and press **R/S**. After letting the wheel spin for a while press **R/S** and the display shows **217**. This is obviously our lucky day! Press the **E** key and the display announces *WIN $40*. Being cautious, we'll play only half that. So, next time around we'll play $20 on 00 and take our chances. The winning number is 101 and pressing **R/S** results in *YOU LOST* in display.

Program Steps

TI-59

Cards: 2
Passes:3

Partition: 559 49

1. Press **5, 2nd, Op, 1, 7**

TI-59

Display: *559 49*
2. Press **INV, 2nd, FIX, CLR**
3. Feed in magnetic cards
4. Press **RST, R/S**
Display: *555555*
5. Key in amount of wager
6. Press **A**
7. Key in where to place bet
8. Press **B**
9. Press **C** to start wheel
10. Press **R/S** to stop wheel
Display: Number on which wheel has stopped
11. Press **E**
Display: If you won, a flickering *555555* followed by amount won. If you lost, a flashing *99.99999999 99*

HP-41C

Cards: 4
Passes: 7
Size: 017
Display: *SEE RULES?* then *0=YES*
1. If you want to read the rules, press **0** and **R/S**, otherwise simply **R/S**
Display: *PLACE BET $?*
2. Key in wager; **R/S**
Display: *PLACE ON?*
3. Key in where to place bet: **R/S**
Display: *SPIN WHEEL*, then *R/S TO STOP*
4. Press **R/S**
Display: Number on which wheel has stopped
5. Press **E** (top row, key 5)
Display: Either *WIN $* and the amount or *YOU LOST*
6. Press **R/S**
Display: *AGAIN ? 0=NO*

BLACKJACK

For those not familiar with blackjack, it's a card game in which the dealer deals two or more cards to all players including himself. The object of the game is to come as close to 21 (in the face value of the cards in your hand) as possible without going over. Anyone going over 21 loses. Those under 21 win if their count is higher than that of the dealer. If a player's count is the same as that of the dealer, it's referred to as "push", and the player keeps his or her stake, but there is no payoff. Anyone reaching 21 with two cards has "blackjack" and he or she wins one and one half times the wagered amount. All other payoffs are one to one.

All cards are worth their face value, except that all face cards are worth 10, and the ace can be either 1 or 11, player's option. There are a number of additional refinements to this game, but they don't concern us in the context of the program.

The challenges, at first glance, seemed to be these: first it was necessary to create a series of steps which would represent a full deck of 52 cards, with four cards of each kind. This series of steps would have to be manipulated in some way to simulate shuffling a deck of cards.

I also wanted the face cards to appear by name rather than simply as 10s, which would have been acceptable, but that would have meant there were a total of 16 10s in the deck, which I wanted to avoid. On the other hand, the computer cannot perform calculations with words, like Ace or

King (the display responds to such attempts with *ALPHA DATA*), so these face cards would have to be changed into numerical equivalents at some point in the program.

I decided to set the game up for a dealer and three players, making provisions to permit one player to receive any number of cards (for instance, a group of twos and threes) while another might want to stand on the first two cards dealt. Similarly, the dealer (who, by the way, must take another card when he or she has 16 or less, but who can not take another card when he has 17 or more) must be in a position to take as many cards as he or she needs.

TI-59. There is no alpha-numeric display so all face cards are depicted as 10s, and the ace is a 1. In the way I have written the program (Fig. 11-7), when you initiate by pressing **CLR RST R/S**, it automatically goes through the program steps under **LBL E′** which place the different card values into R_{01} through R_{10} for subsequent use in the loop (steps 000 through 040). When that step has been completed it leaves a *555555* in the display to tell us the computer is ready to play.

The card loop itself is under key **E**, and is composed of steps 041 through 152. It takes the TI about seven seconds to go through the entire loop, easily fast enough to make it impossible to predict what card will be picked. To pick a card press **E**, which gets the loop started, and then press **R/S** which puts the drawn card into display. Press **A** for Player One, **B** for Player Two, **C** for Player Three and **D** for the dealer. In each case the card value is stored in the appropriate place and when additional cards are drawn they are added to that value, each time displaying to total count in the hand as soon as the **A, B, C** or **D** key has been pressed.

That's all there is to it. Cards can be added to any player's hand at random. Consecutive order is of no importance in this program.

HP-41. Figure 11-8 is the printout of the program. Let's play the game one time around, and then look at the individual sections within the program and describe what each does.

Press **ON**. Some meaningless display, left from past application, appears in the display window. Press **shift 1/x**. This key has been assigned to call up the program; the word *BLACKJACK* appears in the display for a moment and is replaced by *SHUFFLE DECK* which in turn is replaced by *TO DEAL: R/S*.

To deal the first card to the first player, press **R/S**. Say that this time an *8* appears in the display. To place the 8 in front of Player One, use **LN**, which has been assigned for that purpose. Press **LN** and the bird-like symbol moving across the display tells us the computer is computing.

Press **R/S**. This time the number in the display is *10*. Press **LN** and the 10 is given to Player Two, while the computer goes back to work. Press **R/S** and the display now shows *KING*. The third player has gotten a face card. Press **LN** and the king is given to Player Three while the computer goes back to work.

Press **R/S** and the display reads *9*, the card which the dealer has dealt

```
000  25 CLR    051  43 RCL    102  10  10    153  76 LBL
001  47 CMS    052  05  05    103  43 RCL    154  11  A
002  76 LBL    053  43 RCL    104  03  03    155  42 STO
003  10 E'     054  07  07    105  43 RCL    156  15  15
004  01  1     055  43 RCL    106  05  05    157  85  +
005  42 STO    056  10  10    107  43 RCL    158  43 RCL
006  01  01    057  43 RCL    108  07  07    159  11  11
007  02  2     058  09  09    109  43 RCL    160  95  =
008  42 STO    059  43 RCL    110  10  10    161  42 STO
009  02  02    060  10  10    111  43 RCL    162  11  11
010  03  3     061  43 RCL    112  09  09    163  91 R/S
011  42 STO    062  02  02    113  43 RCL    164  76 LBL
012  03  03    063  43 RCL    114  10  10    165  12  B
013  04  4     064  04  04    115  43 RCL    166  42 STO
014  42 STO    065  43 RCL    116  02  02    167  16  16
015  04  04    066  10  10    117  43 RCL    168  85  +
016  05  5     067  43 RCL    118  04  04    169  43 RCL
017  42 STO    068  06  06    119  43 RCL    170  12  12
018  05  05    069  43 RCL    120  10  10    171  95  =
019  06  6     070  08  08    121  43 RCL    172  42 STO
020  42 STO    071  43 RCL    122  06  06    173  12  12
021  06  06    072  10  10    123  43 RCL    174  91 R/S
022  07  7     073  43 RCL    124  08  08    175  76 LBL
023  42 STO    074  01  01    125  43 RCL    176  13  C
024  07  07    075  43 RCL    126  10  10    177  42 STO
025  08  8     076  03  03    127  43 RCL    178  17  17
026  42 STO    077  43 RCL    128  01  01    179  85  +
027  08  08    078  10  10    129  43 RCL    180  43 RCL
028  09  9     079  43 RCL    130  03  03    181  13  13
029  42 STO    080  05  05    131  43 RCL    182  95  =
030  09  09    081  43 RCL    132  10  10    183  42 STO
031  01  1     082  07  07    133  43 RCL    184  13  13
032  00  0     083  43 RCL    134  05  05    185  91 R/S
033  42 STO    084  09  09    135  43 RCL    186  76 LBL
034  10  10    085  43 RCL    136  07  07    187  14  D
035  05  5     086  10  10    137  43 RCL    188  42 STO
036  05  5     087  43 RCL    138  09  09    189  18  18
037  05  5     088  02  02    139  43 RCL    190  85  +
038  05  5     089  43 RCL    140  10  10    191  43 RCL
039  05  5     090  10  10    141  43 RCL    192  14  14
040  05  5     091  43 RCL    142  02  02    193  95  =
041  91 R/S    092  04  04    143  43 RCL    194  42 STO
042  58 FIX    093  43 RCL    144  04  04    195  14  14
043  00  00    094  06  06    145  43 RCL    196  91 R/S
044  76 LBL    095  43 RCL    146  06  06    197  00  0
045  15  E     096  08  08    147  43 RCL    198  00  0
046  00  0     097  43 RCL    148  08  08    199  00  0
047  43 RCL    098  10  10    149  43 RCL    200  00  0
048  01  01    099  43 RCL    150  10  10    201  00  0
049  43 RCL    100  01  01    151  61 GTO    202  00  0
050  03  03    101  43 RCL    152  15  E     203  00  0
                                             204  00  0
```

Fig. 11-7. The TI-59 version of Blackjack.

him- or herself. Press **LN** and the display now shows: *DEALER: 9*. Press **R/S** and the computer goes back to work for the second round. Steps 3 through 10 are repeated and Player One draws a 10, Player Two draws an 8, Player Three draws a king and the dealer draws a 2. The display now reads *DEALER : 11*. At this point each person has two cards and the other players will want to look at their hands to see if they might want another card.

Press Σ +; the display shows *18*, the total of the two cards in the hands of Player One. Obviously Player One should not take another card.

Press **1/x**; the display once more shows *18*. Player Two also has two cards adding up to 18. Press $\sqrt{x}$ and the display shows *20*. Player Three has two kings. Press **LOG** and the display shows *11*, telling us once more that the dealer has only 11 and must take another card.

Player One wants no card so press **0**. Press **LN**, and the computer works. Press **R/S**; a number or face card will appear in the display, but since Player Two wants no card, press **0**. Press **LN** and the computer works again. Press **R/S** and again there is a number or face card which Player Three doesn't want, so press **0**. Press **LN** and the computer works again. Press **R/S** and the display shows the dealer has drawn a *JACK*.

Press **LN** and the display reads *DEALER: 21*. At this point, if we want

```
01♦LBL "PPG 036"
    02 CLRG
    03 CF 01
    04 CF 02
    05 CF 03
    06 " JACK"
    07 ASTO 10
    08 " QUEEN"
    09 ASTO 11
    10 " KING"
    11 ASTO 12
    12 " ACE"
    13 ASTO 13
  14 "BLACKJACK"
    15 AVIEW
    16 PSE
17 "SHUFFLE DECK"
    18 AVIEW
    19 PSE
20 "TO DEAL : R/S"
    21 AVIEW
    22 PSE
    23 FIX 0

    24♦LBL 00
    25 RCL 13
    26 3
    27 5
    28 RCL 10
    29 7
    30 RCL 11
    31 9
    32 RCL 12
    33 2
    34 4
    35 RCL 10
    36 6
    37 8
    38 RCL 11
    39 RCL 13
    40 3
    41 RCL 12
    42 5
    43 7
    44 RCL 10
    45 9
    46 10
    47 2
    48 4
    49 6
    50 8
    51 RCL 11
    52 RCL 13
    53 3
    54 10
    55 5
    56 RCL 12
    57 7
    58 9
    59 10
    60 2
    61 4
    62 6
    63 8
    64 RCL 10
    65 RCL 13
    66 3
    67 5
    68 7
    69 9
    70 10
    71 2
    72 4
    73 RCL 11
    74 6
    75 8
    76 RCL 12
    77 GTO 00
    78♦LBL "Z"
    79 STO 05
    80 ENTER↑
    81 RCL 12
    82 X=Y?
    83 GTO 05
    84 RCL 05
    85 ENTER↑
    86 RCL 11
    87 X=Y?
    88 GTO 05
    89 RCL 05
    90 ENTER↑
    91 RCL 10
    92 X=Y?
    93 GTO 05
    94 RCL 05
    95 ENTER↑
    96 RCL 13
    97 X=Y?
    98 GTO 06
    99 GTO 07
100♦LBL 05
101 10
102 GTO 01

103♦LBL 06
104 1
105 GTO 01

106♦LBL 07
107 RCL 05

108♦LBL 01
109 FS? 01
110 GTO 02
111 FS? 02
112 GTO 03
113 FS? 03
    114 GTO 04
    115 ST+ 01
    116 SF 01
    117 GTO 00

    118♦LBL 02
    119 ST+ 03
    120 CF 01
    121 SF 02
    122 GTO 00

    123♦LBL 03
    124 ST+ 03
    125 CF 02
    126 SF 03
    127 GTO 00

    128♦LBL 04
    129 ST+ 04
    130 CF 03
131 "DEALER: "
    132 ARCL 04
    133 AVIEW
    134 STOP
    135 GTO 00

    136♦LBL "Y"
    137 VIEW 01
    138 STOP

    139♦LBL "X"
    140 VIEW 02
    141 STOP

    142♦LBL "W"
    143 VIEW 03
    144 STOP

    145♦LBL "V"
    146 VIEW 04
    147 STOP
    148 GTO 00
    149 END
```

Fig. 11-8. Blackjack game on the HP-41C.

to, we can again look at the three player's hands, but one way or the other, the dealer has won this round and the players have lost their stakes. Better luck next time.

Let's look closer at Fig. 11-9 and see what actually went on inside the computer while we played the game.

When we initialized the program it executed step 02 which cleared all registers of leftover data and steps 03, 04, 05 which would clear any flags which might have been left if the program was played before but turned off without finishing. Steps 06 through 13 place the words designating the face cards into four different memory registers. The **ASTO** (instead of **STO**) simply means that what is being stored are alpha characters, not digits.

In step 14 the name of the game is displayed, (**AVIEW**) and it is held in the display long enough to read (**PSE**). The same is true of the next two displays. In step 23 the computer is told not to bother to display decimals.

The **LBL 00** section consists of steps 24 through 77, representing a deck of 52 cards. As you can see, step 77 tells the computer to **GTO 00**, back to the beginning of that section. This is an infinite loop and unless we stop it somewhere, it will continue to go 'round and 'round indefinitely. It has the effect of shuffling a deck of cards. Stop it by pressing **R/S**. There is no way to know which of the 52 cards is going to come up in the display when we press **R/S**. It takes the computer slightly less than four seconds to make a complete run through all 52 cards, meaning that it encounters a new card every 8/100 of a second, or runs by 14 cards each second.

When we press **R/S**, the card that happened to be in the right place at the right moment will show up in the display, and the action of the revolving loop is temporarily halted. Since pressing **R/S** a second time would simply get the loop going again, a different way had to be found to place the card, currently in the display window, into the hand of the player. This is accomplished by pressing **LN** instead of **R/S**. Pressing **LN** tells the computer to go to **LBL Z** which is at step 78.

Here the face cards are turned into 10s, and aces into 1s. (Anyone wanting the ace to represent 11 will simply have to add the 10 mentally.) The **STO 05** in step 79 places the drawn card into R_{05}. In steps 80 through 97, that card in R_{05} is compared to the data stored in R_{10} (jack), R_{11} (queen), R_{12} (king), and R_{13} (ace). If the question $x = y?$ (does R_{05} equal any data in the other registers) is answered in the affirmative, the computer is told to go to **LBL 05 (GTO 05)** or to **LBL 06 (GTO 06)**, found in steps 100 through 105, in which the alpha characters are replaced by a digital 10 or 1. If the answer is negative each time, the computer is told to go to **LBL 07 (GTO 07)** which consists of steps 106 and 107.

Regardless of whether the computer has arrived at this point via **LBL 05, LBL 06** or **LBL 07**, the next step is **LBL 01**. Steps 109 through 114 of this section ask whether flags 01, 02 or 03 have been set. If none have been set, the computer goes on to step 115 in which it places the amount of the drawn card into the hand of Player One. If flag 01 is set, the computer will instead go to **LBL 02** and give the card to Player Two. If flag 02 is set, it

```
01♦LBL "PPG 036"
CLRG
CF 01
CF 02
CF 03
" JACK"
ASTO 10
" QUEEN"
ASTO 11
" KING"
ASTO 12
" ACE"
ASTO 13
"BLACKJACK"
AVIEW
BLACKJACK
PSE
"SHUFFLE DECK"
AVIEW
SHUFFLE DECK
PSE
"TO DEAL : R/S"
AVIEW
TO DEAL : R/S
PSE
FIX 0

24♦LBL 00
RCL 13
" ACE"  ***
3
5
RCL 10
" JACK"  ***
7
RCL 11
" QUEEN"  ***
9
RCL 12
" KING"  ***
2
XEQ "Z"
78♦LBL "Z"
STO 05
ENTER↑
RCL 12
" KING"  ***
X=Y?
RCL 05
2.  ***
ENTER↑
RCL 11
" QUEEN"  ***
X=Y?
RCL 05
2.  ***
ENTER↑
RCL 10
" JACK"  ***
XEQ "Z"
78♦LBL "Z"
STO 05
ENTER↑
RCL 12
" KING"  ***
X=Y?
RCL 05
" JACK"  ***
ENTER↑
RCL 11
" QUEEN"  ***
X=Y?
RCL 05
" JACK"  ***
ENTER↑
RCL 10
" JACK"  ***
X=Y?
GTO 05
XEQ "Z"
78♦LBL "Z"
STO 05
ENTER↑
RCL 12
" KING"  ***
X=Y?
RCL 05
" JACK"  ***
ENTER↑
RCL 11
" QUEEN"  ***
X=Y?
RCL 05
" JACK"  ***
ENTER↑
RCL 10
" JACK"  ***
X=Y?
GTO 05
100♦LBL 05
10
GTO 01
108♦LBL 01
FS? 01
FS? 02
FS? 03
ST+ 01
SF 01
GTO 00

24♦LBL 00
RCL 13
" ACE"  ***
3
5
RCL 10
" JACK"  ***
7
RCL 11
" QUEEN"  ***
9
RCL 12
" KING"  ***
2
4
RCL 10
" JACK"  ***
6
8
```

Fig. 11-9. The trace of Blackjack.

```
        RCL 11
" QUEEN"   ***
        RCL 13
 " ACE"   ***
           3
        RCL 12
 " KING"   ***
           5
           7
        RCL 10
 " JACK"   ***
           9
          10
           2
           4
           6
           8
        RCL 11
" QUEEN"   ***
        RCL 13
 " ACE"   ***
           3
          10
           5
        RCL 12
 " KING"   ***
           7
           9
          10
           2
           4
           6
           8
        RCL 10
 " JACK"   ***
        RCL 13
 " ACE"   ***
           3
           5
           7
           9
          10
           2
```

```
           4
        RCL 11
" QUEEN"   ***
           6
           8
        RCL 12
 " KING"   ***
      GTO 00
   24↑LBL 00
        RCL 13
 " ACE"   ***
           3
           5
        RCL 10
 " JACK"   ***
           7
        RCL 11
" QUEEN"   ***
           9
        RCL 12
 " KING"   ***
           2
           4
        RCL 10
 " JACK"   ***
           6
           8
        RCL 11
" QUEEN"   ***
        RCL 13
 " ACE"   ***
           3
        RCL 12
 " KING"   ***
           5
           7
        RCL 10
 " JACK"   ***
           9
          10
           2
           4
           6
```

```
           8
        RCL 11
" QUEEN"   ***
      XEQ "Z"

   78↑LBL "Z"
      STO 05
      ENTER↑
      RCL 12
 " KING"   ***
        X=Y?
      RCL 05
" QUEEN"   ***
      ENTER↑
      RCL 11
" QUEEN"   ***
        X=Y?
      GTO 05

  100↑LBL 05
          10
      GTO 01

  108↑LBL 01
      FS? 01
      GTO 02
  118↑LBL 02
      ST+ 02
       CF 01
       SF 02
      GTO 00

   24↑LBL 00
      RCL 13
 " ACE"   ***
           3
           5
      RCL 10
 " JACK"   ***
           7
      RCL 11
" QUEEN"   ***
           9
```

```
        RCL 12
" KING"    ***
             2
             4
        RCL 10
" JACK"    ***
             6
             8
        RCL 11
" QUEEN"   ***
        RCL 13
 " ACE"    ***
             3
        RCL 12
" KING"    ***
      XEQ "Z"

    78↑LBL "Z"
        STO 05
        ENTER↑
        RCL 12
" KING"    ***
          X=Y?
        GTO 05

    100↑LBL 05
            10
        GTO 01

    108↑LBL 01
        FS? 01
        FS? 02
        GTO 03

    123↑LBL 03
        ST+ 03
         CF 02
         SF 03
        GTO 00
     24↑LBL 00
        RCL 13
 " ACE"    ***
             3
             5
        RCL 10
" JACK"    ***
             7
        RCL 11
" QUEEN"   ***
             9
        RCL 12
" KING"    ***
             2
             4
        RCL 10
" JACK"    ***
             6
             8
        RCL 11
" QUEEN"   ***
        RCL 13
 " ACE"    ***
      XEQ "Z"

    78↑LBL "Z"
        STO 05
        ENTER↑
        RCL 12
" KING"    ***
          X=Y?
        RCL 05
 " ACE"    ***
        ENTER↑
        RCL 11
" QUEEN"   ***
          X=Y?
        RCL 05
 " ACE"    ***
        ENTER↑
        RCL 13
 " ACE"    ***
          X=Y?
        GTO 06
    103↑LBL 06
             1
        GTO 01

    108↑LBL 01
        FS? 01
        FS? 02
        FS? 03
        GTO 04

    128↑LBL 04
        ST+ 04
         CF 03
    "DEALER: "
        ARCL 04
DEALER 1   AVIEW
        XEQ "Y"

    136↑LBL "Y"
        VIEW 01
            10.
          STOP
        XEQ "X"

    139↑LBL "X"
        VIEW 02
            10.
          STOP
        XEQ "W"

    142↑LBL "W"
        VIEW 03
            10.
          STOP
        XEQ "V"

    145↑LBL "V"
        VIEW 04
             1.
          STOP
```

Fig. 11-9. Continued from page 277.

goes to **LBL 03** and gives the card to Player Three, and if flag 03 is set it goes to **LBL 04** and gives the card to the dealer. In each instance the flag is cleared as soon as it has served its purpose (**CF 01, 02** etc., steps 120, 125 and 130). Whenever the dealer draws a card for himself the computer goes on to steps 131 through 134 which display the total of the dealer's cards. At this point the players may want to look at their own cards. The user keys come into play: key $\Sigma+$, calls up **LBL Y**, representing Player One, key **1/x** calls up **LBL X**, representing Player Two, key $\sqrt{x}$ calls up **LBL W**, representing Player Three, and key **LOG** calls up **LBL V**, representing the dealer.

When **R/S** is pressed at any time (except when the loop is running), it will start the loop and new cards can again be picked by pressing **R/S**. To start a new game, press **shift 1/x** and start from the top, otherwise data left over from the previous game might remain in the registers and they would then automatically be added to the new cards being drawn.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59
1. Press **CLR**
2. Press **RST**
3. Press **R/S**
Display: *555555*
4. Press **E**
Display: Flickering *C* as card loop is in operation
5. Press **R/S**
Display: Card value
6. Press **A** for player 1, **B** for player 2, **C** for player 3 or **D** for dealer.
Display: Total count of cards drawn so far by that player or the dealer
7. Repeat steps 4 through 6 for additional cards. Consecutive order of players and dealer is of no importance.

HP-41C

Cards: 2
Passes: 3
Size: 015
Display: *SHUFFLE DECK*, then *TO DEAL: R/S*
1. Press **R/S**
Display: card picked for player 1
2. Press **E** (top row, key 5)
3. Press **R/S**
Display: card picked for player 2
4. Press **E**
5. Press **R/S**
Display: Card picked for player 3
6. Press **E**
7. Press **R/S**
Display: card picked by dealer
8. Press **E**
Display: *DEALER*: and card count
9. Press **A** (top row, key 1)
Display: card count for player 1
10. Press **B** (top row, key 2)
Display: card count for player 2
11. Press **C** (top row, key 3)
Display: card count for player 3
If you want to see the dealer's card once more:
12. Press **D** (top row, key 4)
Display: card count, dealer
13. Press **R/S**

Display: Moving symbol showing card loop in operation
14. Press **R/S**
Display: card 2 for player 1
15. Press **E**
16. Press **R/S**
Display: card 2 for player 2
17. Press **E**
18. Press **R/S**
Display: card 2 for player 3
19. Press **E**
20. Press **R/S**
Display: card 2 for dealer
21. Press **E**
Display: *DEALER*: and total
Pressing **A, B, C** shows totals for three players, **D** repeats total for dealer
Repeat for additional cards.
Press **0** for players or dealer who want no additional cards.

THE DICE GAME

This game simulates a dice game. Using this program will let you play the game in a moving car, airplane or wherever you like, without having to have a pair of dice.

This is one of those programs better suited to the capabilities of the HP-41C, though it can be done with the TI-59 too, and I have prepared the program for both computers (Figs. 11-10 and 11-11).

TI-59. The program, illustrated in Fig. 11-10, uses the letter keys **A, B,** and **C.** Key **A** controls the loop which determines the throw of the dice. Key **B** is used for the first throw, which determines either the point which has to be made, or results in an immediate win or loss. Key **C** is used for all subsequent throws of the dice which are made attempting to match the point before a 7 comes up.

Here is how to play it:

First, to clear out left-over data and to prepare the loop, press **RST R/S**, and after a brief hesitation the display will show *555555* telling us the program is ready to run. Press **A** and after a few moments press **R/S.** The display will show the result of the first throw of the dice. I had a *6*. Now press **B** and the display returns to show *6* as the point to make. Press **A** again, wait a moment and then press **R/S.** I got a *10*. Press **C** and the display returns with *6* as the point to make. Once more press **A**, wait a moment and then press **R/S.** This time I've got a **7**. Press **C** and after a moment the display shows *777777*, meaning I have lost.

If I had made my point, *6*, and had pressed **C,** the display would have responded with a flashing *99.9999999 99*. Similarly, if my first throw had been a *7* or an *11*, the same flashing display of nines would have appeared after pressing **B**. On the other hand, if the first throw had resulted in a *2, 3* or *12*, the display would have shown a row of 7s to let me know I had lost.

In Fig. 11-10, steps 000 through 044 are used to prepare the loop. Steps 045 through 092 are the loop itself.

Steps 093 through 129 are used to store and evaluate the first throw. It is stored in R_{00} and compared to 2, 3, 12, 7, and 11 to determine if there is an

immediate win or loss. If it's a 7 or 11, the computer is told to go to **LBL Yx** which contains the routine for the flashing 9s. If it's a loss, the computer is told to go to **LBL 1/X** which is the section with all the 7s. If neither is the case, the computer is told to go to **LBL** $\sqrt{x}$ which contains the command to recall the point stored in R_{00} and place it into display.

Steps 130 and those following are **LBL C**. Here the second and subsequent throws of the dice are stored in R_{13} and are then again compared to 7 and to the point, the amount stored in R_{00}. If it proves equal to 7, the computer must go to **LBL 1/X** and produce the string of 7s which stand for loss. If the throw is equal to the point, the computer goes to **LBL Yx** and greets the player with flashing 9s. If neither is the case, the computer goes to **LBL** $\sqrt{x}$ to again display the point as a reminder that the dice must be thrown again.

HP-41C. When we initialize the program, the display first reads *DICE*, then changes to *1st THROW* and then to *R/S = RESULT*. Now press **R/S** and the display will show any number from 2 to 12, representing the result of the first throw of the two dice. Mine was 10.

Press **shift D** and the display reads *THE POINT IS 10* followed by *THROW DICE*, followed again by *R/S = RESULT*. Press **R/S**—this time I got a **4**. Press **shift D**. The display reads: *THE POINT IS 10* then *THROW DICE*, then *R/S = RESULT*. This will continue to happen until you roll either a *10* and win, or a *7* and lose.

Here is what happens if you roll a *10*: the display will read *YOU WIN*. If you roll a *7* the display will read *YOU LOSE*, followed by *NEW SHOOTER*, followed by *FIRST THROW*, followed by *R/S = RESULT*.

Either way, we start all over again. If you won you keep the dice, if you lose you hand the dice over to the next player.

On the first throw, if you throw a 7 or 11 you win, but if you throw a 2, 3 or 12 you lose your money but keep the dice. Here is what happens under those conditions:

You throw a *2*, the computer says, *YOU LOSE* and then goes on to *FIRST THROW*, etc.

You throw a *3* and the same thing happens.

You throw a *12*, and the same thing happens.

You throw a *7*, and the computer simply says *YOU WIN*. Pressing **R/S** will start a new game.

You throw an *11*, and the same thing happens.

In Fig. 11-11 the first 14 steps display the various legends. Each is labeled separately to permit the computer to go to whichever is needed, depending on what took place before the command to return there.

Steps 15 through 35 are the loop. It contains all the numbers which can be thrown with two dice, each one repeated as often as the different dice combinations indicate (2, 3, 11, and 12 appear only once because there is only one possible combination which can result in those figures: 1+1, 1+2, 5+6 and 6+6. For numbers 4, 5, 9 and 10 there are two possible combinations, so they are represented twice in the loop. Numbers 6, 7 and 8 can be

```
000  47  CMS     048  23  LNX     096  00   00
001  25  CLR     049  43  RCL     097  32  X:T
002  02   2      050  02   02     098  43  RCL
003  42  STO     051  43  RCL     099  11   11
004  02   02     052  04   04     100  67   EQ
005  03   3      053  43  RCL     101  45  Y^
006  42  STO     054  03   03     102  43  RCL
007  03   03     055  43  RCL     103  00   00
008  04   4      056  05   05     104  32  X:T
009  42  STO     057  43  RCL     105  02   2
010  04   04     058  06   06     106  67   EQ
011  05   5      059  43  RCL     107  35  1/X
012  42  STO     060  04   04     108  43  RCL
013  05   05     061  43  RCL     109  00   00
014  06   6      062  07   07     110  32  X:T
015  42  STO     063  43  RCL     111  03   3
016  06   06     064  05   05     112  67   EQ
017  07   7      065  43  RCL     113  35  1/X
018  42  STO     066  08   08     114  43  RCL
019  07   07     067  43  RCL     115  00   00
020  08   8      068  06   06     116  32  X:T
021  42  STO     069  43  RCL     117  43  RCL
022  08   08     070  12   12     118  12   12
023  09   9      071  43  RCL     119  67   EQ
024  42  STO     072  07   07     120  35  1/X
025  09   09     073  43  RCL     121  43  RCL
026  01   1      074  08   08     122  00   00
027  00   0      075  43  RCL     123  32  X:T
028  42  STO     076  11   11     124  07   7
029  10   10     077  43  RCL     125  67   EQ
030  01   1      078  08   08     126  45  Y^
031  01   1      079  43  RCL     127  61  GTO
032  42  STO     080  07   07     128  34  FX
033  11   11     081  43  RCL     129  91  R/S
034  01   1      082  06   06     130  76  LBL
035  02   2      083  43  RCL     131  13   C
036  42  STO     084  09   09     132  42  STO
037  12   12     085  43  RCL     133  13   13
038  05   5      086  10   10     134  32  X:T
039  05   5      087  43  RCL     135  07   7
040  05   5      088  09   09     136  67   EQ
041  05   5      089  43  RCL     137  35  1/X
042  05   5      090  10   10     138  43  RCL
043  05   5      091  61  GTO     139  13   13
044  91  R/S     092  23  LNX     140  43  RCL
045  76  LBL     093  76  LBL     141  00   00
046  11   A      094  12   B      142  32  X:T
047  76  LBL     095  42  STO     143  43  RCL
```

Fig. 11-10. The TI-59 version of the Dice game.

```
144  13  13    158  07  7     172  07  7
145  67  EQ    159  32  X:T   173  07  7
146  45  Y^    160  43  RCL   174  07  7
147  61  GTO   161  13  13    175  07  7
148  34  FX    162  67  EQ    176  07  7
149  76  LBL   163  35  1/X   177  91  R/S
150  45  Y^    164  61  GTO   178  76  LBL
151  58  FIX   165  34  FX    179  34  FX
152  09  09    166  76  LBL   180  58  FIX
153  01  1     167  35  1/X   181  09  09
154  55  ÷     168  58  FIX   182  43  RCL
155  00  0     169  09  09    183  00  00
156  95  =     170  07  7     184  91  R/S
157  91  R/S   171  07  7     185  00  0
```

achieved with three combinations, which is why there are three of each). This loop begins to go around and around (at about 18 different possible results per second) and is stopped when the player presses **R/S**, putting whatever happens to be in the X register at that second into the display.

The **LBL DD** section is called up by pressing the user-assigned key **shift D** (the computer must be in user mode). In this section the computer determines if the first throw was a 7, in which case it is directed to **GTO A** where the *YOU WIN* legend is located (steps 91 through 94), after which it is directed by step 95 to **GTO C** which starts a new game with *1ST THROW*. If it is not a 7, we establish whether it is an 11, which would have the same result. It checks for a 2, 3 or 12, in which case the computer goes to **LBL B** and the announcement *YOU LOSE*. If none of these are the throw, the computer is directed to continue to **LBL 04,** telling the player what his point is and to throw the dice. The **shift D** key is then pressed again, but this time when the computer encounters the question **FS? 01** (is flag 01 set?), the answer is yes and it is directed to **GTO 03**. There, in section **LBL 03** the throw is checked again. If it's a 7, the command is to go to **LBL B** and *YOU LOSE*. If it's not a 7, the computer goes to the next section, **LBL 05** where it is asked if the result of the throw is equal to the result of the first throw (**x=y?**, step 88). If the answer is yes, it goes to **LBL A** and *YOU WIN*. If the answer is no, it goes back to **LBL 04** to display once more *THE POINT IS* followed by *THROW DICE*, and the whole routine starts over again.

Program Steps

TI-59

Cards: 1
Passes: 1
Partition: 479 59

1. Press **RST**
2. Press **R/S**
Display: *555555*

```
 01♦LBL "PPG 035"
      02 " DICE"
      03 AVIEW
      04 PSE

      05♦LBL C
   06 "1ST THROW"
      07 AVIEW
      08 PSE
      09 CF 01
      10 FIX 0

      11♦LBL 02
12 "R/S = RESULT"
      13 AVIEW
      14 PSE

      15♦LBL 00
      16 7
      17 4
      18 6
      19 8
      20 5
      21 7
      22 9
      23 6
      24 8
      25 10
      26 5
      27 6
      28 9
      29 4
      30 10
      31 8
      32 3
      33 7
      34 11
      35 2
      36 12
      37 GTO 00
     38♦LBL "DD"
     39♦LBL 01
     40 FS? 01
     41 GTO 03
     42 STO 01
     43 ENTER↑
     44 7
     45 X=Y?
     46 GTO A
     47 RCL 01
     48 ENTER↑
     49 11
     50 X=Y?
     51 GTO A
     52 RCL 01
     53 ENTER↑
     54 2
     55 X=Y?
     56 GTO B
     57 RCL 01
     58 ENTER↑
     59 3
     60 X=Y?
     61 GTO B
     62 RCL 01
     63 ENTER↑
     64 12
     65 X=Y?
     66 GTO B
     67 ENTER↑
     68♦LBL 04
 69 "POINT IS "
     70 ARCL 01
     71 AVIEW
      72 PSE
 73 "THROW DICE"
      74 AVIEW
      75 PSE
      76 SF 01
      77 GTO 02

     78♦LBL 03
      79 STO 02
      80 ENTER↑
      81 7
      82 X=Y?
      83 GTO B

      84♦LBL 05
      85 RCL 01
      86 ENTER↑
      87 RCL 02
      88 X=Y?
      89 GTO A
      90 GTO 04

      91♦LBL A
    92 "YOU WIN"
      93 AVIEW
      94 STOP
      95 GTO C

      96♦LBL B
   97 "YOU LOSE"
      98 AVIEW
      99 PSE
100 "NEW SHOOTER"
     101 AVIEW
     102 PSE
     103 GTO C
     104 END
```

Fig. 11-11. Dice for the HP-41C.

TI-59

3. Press **A**
4. After a moment, press **R/S**
Display: Result of the first throw of the dice
5. Press **B**
Display: Same as above, the point to make
6. Press **A**
7. Press **R/S**
Display: The result of the throw of the dice
8. Press **C**
Display: If the last throw made the point, the display is a flashing *99.9999999*
If the throw had produced a *7*, the display would have been *777777*: you lost.
9. Steps 6, 7 and 8 can be repeated indefinitely until the player either wins or loses.

HP-41C

Cards: 2
Passes: 3
Size: 005

Display: *1ST THROW* and then *R/S=RESULT*
1. Press **R/S**
Display: Result of first throw:
2. Press **shift D** (top row, key 4)
Display: If the throw was a 7 or 11: *YOU WIN*. If the throw was a 2, 3 or 12: *YOU LOSE*. For any other result: *POINT IS* and the result of the first throw, then *THROW DICE* and then *R/S=RESULT*
3. Press **R/S**
Display: Result of throw of the dice
4. Press **shift D**
Display: If the point was made: *YOU WIN*, if it was a 7: *YOU LOSE*, for anything else: *THROW DICE* and *R/S = RESULT*.
If you do throw a 7 and the display shows *YOU LOSE*, that is followed by *NEW SHOOTER*, then *1ST THROW* and then again *R/S=RESULT*.

Appendix: Software

Texas Instruments and Hewlett-Packard have prepared a large selection of special-purpose programs available as PROMs (Permanent Read-Only Memories), consisting of plug-in modules. Each may, contain as many as several dozen programs dealing with a specific field of science, engineering, business, and so on. It is beyond the scope of this book to examine this software in detail so we have provided a list of the categories which are available.

TEXAS INSTRUMENTS

Agriculture

Metric Conversions
Batch Mix
Beef Cow Ration Analyzer
Feed Lot Ration Analyzer
MP and UFP Determination
Dairy Ration Balancer
Swine and Poultry Ration Formulation
Swine and Poultry Ration Analysis
Relative Value of Various Feeds for Swine Gestation Management
Beef Weaning and Yearling Weight Adjustment
Production Work Sheet
Cattle, Pig or Lamb Feeding Work Sheet
Land Purchase: Financial and Economic Analysis
Farm Loans Analysis

Applied Statistics

Sample Design
Data Entry
Data Evaluation
Model Fitting
Theoretical Distributions

Aviation

Flight Plan with Wind

Flight Plan and Verification
Long Range Flight Plan
Atmosphere, Speed, Temperature and Altitude
Predicting Freezing Level, Lowest Usable Flight Level
Wind Components and Average Vector
Wind Triangle
Dead Reckoning
Rhumbline Navigation
Great Circle Flying
Line of Sight Distance and Altitude; DME Speed Correction
Position and Navigation by One VOR
Area Navigation
Course Correction
Rate of Climb; Turn Performance
General Weight and Balance
Customized Weight and Balance
Pilot Unit Conversions
RNAV Flight Plan
Time Zone Conversions

Business Decisions

Long-Term Financing Requirements
Debt Financing
Investment Evaluation
Project Planning and Budgeting
Break-Even Analysis
Facility Capacity
Economic Reordering and Production Runs
Reorder Timing
Demand Forecasting
Assembly Line Balancing
Short-Term Financing Requirements
Facility Scheduling

Electrical Engineering

Phase-Locked Loop
S=Y Parameter Conversions
Complex Arithmetic
Complex Functions
Complex Trigonometric Functions
Decibels, Nepers, Power, Voltage, Current Ratio Conversions
Signal Detection
Roots of a Polynomial
Chained Multiplication of Polynomials
Reactance Chart
Series-Parallel Impedance Conversions
Active Lowpass, Highpass, and Bandpass Filters
Passive Lowpass Filters
Convolution
Root Locus Calculations
Discrete Fourier Transformations
Smith Chart Calculations
Network Analysis

Leisure Library

Photo I: Exposure Compensation
Photo II: Fill-In Flash
Football Predictor
Bowling Scorekeeper
Chess Rater
Golf Handicapper
Bridge Scorer
Codebreaker
Memo Pad
Blackjack
Acey-Deucy
Craps
Mars Lander
Jive Turkey
Hangman
Learning Nim
Football
Computer Art
Sea Battle
Biorhythms

Marine Navigation

Time-Speed-Distance with Current Sailing
Distance Short of, Beyond, or to Horizon
Velocity Needed to Change Relative Position
Course to Steer and SMG (Planning)
Distance Off One Object and Time of Nearest Approach
DMG, SMG, CMG from Two Objects
Course Made Good from Three Bearings
Dead Reckoning
Rhumbline (mercator) Navigation
Chart Initialization
Running Fix from One Object (Lat/Lon)
Fix from Two Objects (Lat/Lon)
Celestial Navigation:
- Time of Sunrise/Sunset/Twilight
- Planet Location
- Star Identification
- Sextant Correction
- Sight Reduction (Sun, Moon, Planet, Star)
- Fix by Two Observations
- Time of Local Apparent Noon and Sun Lines
- Noon Sight Fix
- Great Circle Sailing

Sailing and Tactics:
- Modified Wind
- SMG, CMG, and Time to Lay-Line
- Distance and Bearing to the Mark
- Unit Conversions

Master Library

Matrix Inversion, Determinants and Simultaneous Equations
Matrix Addition and Multiplication
Complex Arithmetic
Complex Functions
Complex Trig Functions
Polynomial Evaluation
Zeros of Functions
Simpson's Approximation (Continuous)
Simpson's Approximation (Discrete)
Triangle Solution (1)
Triangle Solution (2)
Curve Solution
Normal Distribution
Random Number Generator
Combinations, Permutations, Factorials
Moving Averages
Compound Interest
Annuities
Day of the Week, Days between Dates
Hi-Lo Game
Checking/Savings Account Management
DMS Operations
Unit Conversions

Math/Utilities

Alpha Messages
Printer Formatting
Superplotter
Sorting
Data Arrays
Data Packing
Prime Factors
Hyperbolic Functions
Gamma/Factorial
Random Numbers
Normal Distribution
Interpolation
Roots of Function
Minimax
Romberg Integration

Differential Equations
Discrete Fourier Series
Variable Arithmetic

Pool Water Analysis

Water Balance Program
Drain Routine Program

Real Estate/Investment

Annuities
Remaining Balance/Accumulated Interest
Compound Interest
Straight Line Depreciation
Declining Balance Depreciation
Sum-of-the-Years' Digits Depreciation
Component and Composite Depreciation
Excess Depreciation Recapture
Curve Fits
Forecasting—Automatic Curve Choice
Internal Rate of Return
Cash Flow Analysis
Yearly Amortization Schedule
Investment Feasibility Analysis
Residential Purchase Analysis

Securities Analysis

Earnings Per Share Estimation
Compound Interest
Annuities
Uneven Cash Flow
Stock Valuation
Option Valuation
Option Writing
Warrant Valuation
Bond Valuation
Stock Indicators
Portfolio Selection
Portfolio Bookkeeping
Capital Accumulation Planning
Days Between Dates

Surveying

Traverse Programs:
- Azimuth/Bearing Traverse
- Inverse Traverse
- Field Angle Traverse
- Circle Arc Traverse
- Closure
- Balance—Compass Rule
- Balance—Transit Rule
- Vertical Curve Design

Horizontal Curve Design:
- Horizontal Curve Design (1)
- Horizontal Curve Design (2)
- Horizontal Curve Layout

Slope Reduction:
- EDM Slope Reduction (Slope Angle)
- EDM Slope Reduction (Elevation)
- Stadia Reductions and Traverse

Intersection:
- Intersection (Bearing/Bearing)
- Intersection (Distance/Distance)
- Intersection (Bearing/Distance)
- Intersection of a Perpendicular from a Point to a Line
- Three-Point Resection

Earthwork:
- Borrow Pit Volume
- Earthwork Volume (by Average End Area)

Triangle and Curve Solutions:
- Triangle Solution (1)
- Triangle Solution (2)
- Curve Solution

HEWLETT-PACKARD

(Prerecorded library programs for the HP-41C)

Aviation

Flight Management

General Aircraft Weight and Balance
Determining In-Flight Winds
Flight Plan
Position by One or Two VORs
Course and Speed Corrections

Clinical Lab and Nuclear Medicine

Clinical Chemistry
- Beer's Law
- Body Surface Area
- Creatinine Clearance
- Blood Acid-Base Status
- Oxygen Saturation and Content
- Red Cell Indices

Nuclear Medicine
- Total Blood Volume
- Thyroid Uptake
- Radioactive Decay Corrections

Radioimmunoassay Statistics
- Basic Statistics
- Chi-Square Evaluation and Distribution
- t Statistics
- t Distribution

Circuit Analysis

General Network Analysis
Ladder Network Analysis

Financial Decisions

Compound Interest Solutions
Internal Rate of Return
Modified Internal Rate of Return (FMRR)
Net Present Value
Loan Amortization Schedules
Depreciation Schedules
Bond Price and Yield
Days Between Dates

Games

Super Bagels
Biorhythms
Craps
Hangman
Pinball
Space War
Submarine Hunt

Home Management

Home Budgeting
Travel Expense Record
Stock Portfolio Evaluation
Your Financial Calculator
Remaining Balance and Accumulated Interest
Tax Free Individual Retirement Account (IRA) or Keogh Planning
The True Cost of an Insurance Policy
Checking Account Reconcilation
Home Owner's Equity Analysis

Machine Design

Circular Cams
Generation of a Four Bar System
Progression of a Four Bar System
Progression of a Slider Crank
Gear Forces
Standard External Involute Spur Gears
Helical Spring Design
Forced Oscillator with Arbitrary Function
Coordinate Transformation
Points on a Circle
Circle by Three Points
Unit Conversions

Mathematics

Matrix Operations
Solution to $f(x) = 0$ on an Interval
Polynomial Solutions/Evaluation
Numerical Integration
Differential Equations
Fourier Series

Complex Operations
Hyperbolics
Triangle Solutions
Coordinate Transformations

Navigation

Rhumbline and Great Circle Sailing
Great Circle Course and Distance
Dead Reckoning
Sight Reduction
Perpetual Almanac—Stars, Sun, Planets, Moon
Almanac Interpolator
Great Circle Position

Real Estate

Internal Rate of Return
Modified Internal Rate of Return
Net Present Value
Income Property Analysis
Wrap-Around Mortgage
The Rent or Buy Decision
Present Value of an Increasing/Decreasing Annuity
Annual Percentage Rate of a Loan with Fees

Securities

Bond/Note Price and Yield
Routines for Option Writers Using the Black-Scholes Evaluation Method
Warrant and Option Hedging
Yield on Call Option Sales
Butterfly Options
Bull Spread Option Strategy
Convertible Security Analysis
Convertible Bond Investment Analysis
Stock Portfolio Valuation
Bond Speculation Using Margin

Statistics

Basic Statistics for Two Variables
Moments, Skewness and Kurtosis
Analysis of Variance (One Way)
Analysis of Variance (Two Way, No Replications)
Analysis of Covariance (One Way)
Curve Fitting (Linear, Exponential, Logarithmic, and Power Curve)
Multiple Linear Regression
Polynomial Regression
t Statistics
Chi-Square Evaluation
Contingency Table
Spearman's Rank Correlation Coefficient
Normal and Inverse Normal Distribution
Chi-Square Distribution

Stress Analysis for Mechanical Engineers

Section Properties
Beams
Simply Supported Continuous Beams
Columns
Mohr Circle Analysis
Strain Gauge Data Reduction
Soderberg's Equation for Fatigue
RPN Vector Calculator

Structural Analysis for Civil Engineers

Section Properties
Beams
Simply Supported Continuous Beams
Settling of Continuous Beams
Continuous Frame Analysis
Steel Column Formula
RPN Vector Calculator
Reinforced Concrete Beam
Concrete Columns
Effective Moment of Inertia for Concrete Sections

Surveying

Traverse, Inverse and Sideshots
Compass Rule Adjustment
Transit Rule Adjustment
Intersections
Curve Solutions
Horizontal Curve Layout
Vertical Curves and Grades
Resection
Predetermined Area
Volume by Average End Area
Volume of a Borrow Pit
Coordinate Transformation

Thermal and Transport Science

Unit Management System
Equations of State
Polytropic Processes for an Ideal Gas
Isentropic Flow for Ideal Gases
Conduit Flow
Energy Equation for Steady Flow
Heat Exchangers
Black Body Thermal Radiation

Index

0-8306-2442-2