HEWLETT [hp] PACKARD

Sales and service from 172 offices in 65 countries.
19310 Pruneridge Avenue, Cupertino, CA 95014

For Additional Sales and Service Information Contact Your Local Hewlett-Packard Sales Office or Call 408/996-0100 (Ask for Calculator Customer Service).

00065-90200 Rev. 7/74 Printed in U.S.A.

HEWLETT-PACKARD

HP-65
Owner's Handbook

CuVee Software

WILLY KUNZ
Dipl. El. ing. ETH

HEWLETT PACKARD

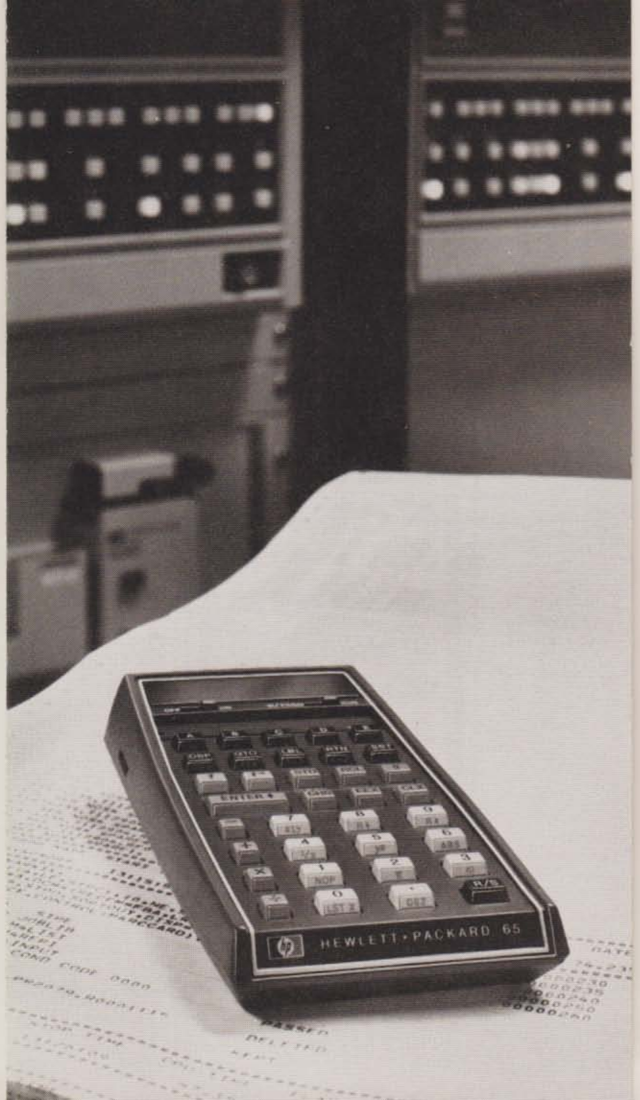
HP-65 Owner's Handbook

July 1974

00065-90200 Rev. 7/74

PRINTED IN U.S.A.

Contents



Introducing the HP-65

Your HP-65 is one of the most advanced pocket calculators in the world. In addition to the computational capabilities of the operational stack that have made the earlier HP-35 and HP-45 models so popular, your new HP-65 is the first pocket calculator to provide *true programmability*.

Three Ways to Use the HP-65

You can use this powerful device in three ways:

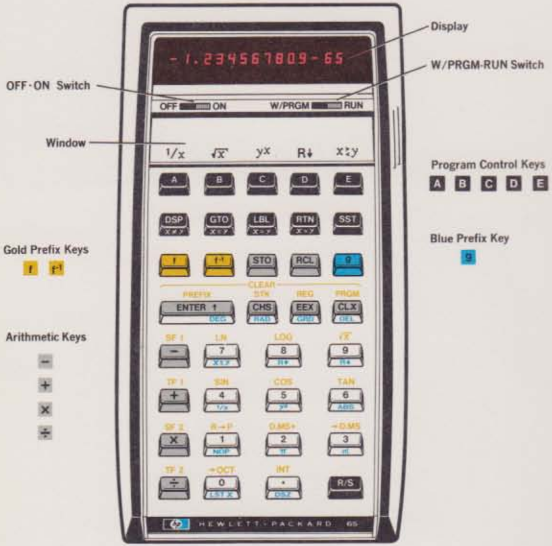
1. To calculate manually.
2. To run a prerecorded program.
3. To write, run, and record your own programs.

1. To Calculate Manually

Figure 1, inside the foldout, illustrates the keyboard layout. Almost every key performs multiple functions. The symbol for the primary function appears on the key. Symbols for alternate functions appear above the key, LN 7, or on the inclined lower key surface, 7 x⇄y.

> **Note:** Designations for alternate functions from above the key or from the inclined lower key surface will appear in the manual in the appropriate color (gold or blue) outlined by a black box like this: LN , x⇄y .

To execute a blue alternate function, press the g prefix key and then the key with the desired blue function. To execute a gold alternate function, press the f prefix key and then the key associated with that function. To execute the inverse (or complement) of that same gold function, press the f^{-1} prefix key and then the key associated with that same gold function. For example:



Before using the calculator, you may need to charge its battery pack as described in appendix A. The calculator can then be operated while the battery is charging or, later, on battery power alone.

Calculate	By Pressing	See Displayed
sin (90°) = 1	90 f SIN	1.00
arc sin (1) = 90°	1 f⁻¹ SIN	90.00
5! = 120	5 g n!	120.00

Now calculate the area of a circle with a radius of 25 using the equation area $= r^2 \cdot \pi$.

Press	See Displayed	
25	25.	Key in the radius.
f^{-1} $\sqrt{x}$	625.00	Calculate r^2.
g π	3.14	Recall pi accurate internally to 10 places.
×	1963.50	Area of the circle.

Sections 1 through 3 are devoted to a description of how to calculate manually.

2. To Run a Prerecorded Program

By using prerecorded magnetic cards, like those supplied in the Standard Pac shipped with your calculator, you can do complex calculations with minimal effort or study of the calculator itself. Let's try running one of these programs now.

1. Select the Compound Interest Program from the Standard Pac card case.

2. Set the W/PRGM-RUN switch to RUN.

3. Insert the card in the right lower slot as shown. When the card is part way in, the motor engages and passes the card through the calculator and out the left side. Let it move freely.

4. The display will read 0.00. If the display blinks, the card did not read properly and program memory will be cleared. Press CLX and reinsert the card.
5. Upon completion, insert the card in the upper "window" slot to identify the top row keys.

You are now ready to use the program.

Example: Investment Plan

What amount must be invested today to have $15,000 at the end of 20 years if the interest rate is 7% compounded quarterly? To solve the problem, just follow the instructions given in the standard format in figure 0-1. You read the "instructions," line by line, key in the required "input," press the indicated "key(s)," and observe the displayed "output." In this case, the answer is displayed after pressing E and C in step 4.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
1	Enter program (Compound				
	Interest as shown on page 7				
	of this manual)				
2	Initialize		RTN	R/S	0.00
3	Input				
	n (n = 20 x 4)	80	A		80.00
	and i (i = 7 ÷ 4)	1.75	B		1.75
	and FV	15000	D		15000.00
4	Compute PV		E	C	3744.02

Figure 0-1. *Instructions for Running the Compound Interest Program*

You can run the program again, if you like, using different values just by keying them in. Your work is minimal because the HP-65 has stored the long keystroke sequence required for this tedious calculation.

3. To Write, Run, and Record Your Own Program

No prior programming experience is necessary to program the HP-65. In fact, before you have finished this brief introduction, you'll have written your first program. It's that easy!

To calculate the area of a circle manually, you have to press five keys after keying in the radius of the circle (see example, page 6). If you had to calculate the area of 10 different circles, it would require too much work. So we'll write a *program* to calculate the area of a circle given its radius.

1. Set the program mode switch to W/PRGM.
2. Press f PRGM to clear the calculator.
3. Press the keys in the order shown below, ignoring the display for now:

Key	Comments
LBL A	Defines the beginning of the program.
f⁻¹ √x g π ×	These are the same keys you pressed to solve the problem manually.
RTN	Defines the end of the program.

If you make a mistake, or your program doesn't work, start again at step 2. Later, in section 5, you will learn how to correct mistakes and the meaning of the numbers in the display.

In the simple program above, you have added LBL A to the top and RTN to the bottom of the same list of keys that solved the problem manually. The program is then controlled by the A key, although any one of the program control keys (A *thru* E) could have been used to execute the program if the appropriate label had been used. You'll find that the ability to define programs using the top row keys is one of the most convenient and powerful features of your HP-65.

To Run Your Program First set the W/PRGM-RUN switch to RUN. Now calculate the areas of circles with radii of 10, 19, and 24.

Press	See Displayed	
10 A	314.16	Area of the first circle.
19 A	1134.11	Area of the second circle.
24 A	1809.56	Area of the third circle.

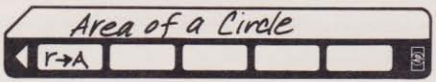
To Record Your Program.

1. Select a blank, unprotected (unclipped) magnetic card.

unprotected

2. Switch to W/PRGM mode.
3. Pass the card through the right lower slot exactly as you did when entering a prerecorded program.

Your program is now recorded on the magnetic card. Be sure and mark the card so you don't forget what the program does. The finished card might look like this when you are through:



protected

And that's all there is to it! Of course, this is only a simple program written in the most convenient way possible. For a more complete picture of programming, you'll want to read section 4.

Onward

If you are a beginner, you will appreciate the step-by-step explanations in this handbook. But even if you are an old hand at using calculators, you can minimize the time you spend calculating by following the procedures presented throughout. For those of you who are familiar with Hewlett-Packard pocket calculators, you may want to skip directly to the programming section and cover the remaining material at your leisure.

If the manual does not answer all your questions, contact your nearest HP Sales and Service Office, or, if you are in the U.S. dial (408) 996-0100 and ask for Customer Service. We want you to be completely satisfied with your HP-65.

Section 1

How To Get Started

Power On

Your HP-65 calculator is shipped fully assembled and is ready to operate after making a few simple checks. If you have just received your calculator, please be sure that you have all of the standard accessories and that the calculator's battery pack has been charged. *(Refer to appendix A.)* If the battery pack is already charged or if you plan to run the calculator from the charger, here's how to get started:

- Set the W/PRGM-RUN switch to RUN.
- Set the power switch to ON.

You should now see displayed 0.00; if not, please turn to appendix C.

Initial Display

Basically, numbers are stored and manipulated internally in the machine in "registers." Each number, no matter how simple (i.e., 0, 1, or 5) or how complex (i.e., 3.141592654, −23.28362, or $2.87148907 \times 10^{27}$) occupies one entire register. Whenever you switch the calculator ON with the W/PRGM-RUN switch set to RUN, the display shows 0.00. This represents the contents of the display, or X-register. Every number keyed into the calculator goes first to the X-register, which is the only visible register. Similarly, you must bring every computed result to the X-register before it can be viewed.

The displayed X-register is one of the four registers inside the calculator that are positioned to form the "operational stack." We label these registers, X, Y, Z, and T. They are "stacked" one on top of the other with the displayed X-register on the bottom. When the calculator is switched ON, these four registers are cleared to 0.00.

Name	Register	
T	0.00	
Z	0.00	
Y	0.00	
X	0.00	(always displayed)

As you'll see, the "stack" allows you to solve almost any equation without storing intermediate results, helping make your calculator one of the most powerful on the market.

Keying In and Entering Numbers

Key in numbers from left to right and include the decimal point if it is a part of the number. For example, 314.32 is keyed in by pressing: 3 1 4 • 3 2

Why not try it yourself now? If you make a mistake, clear the entire number by pressing **CLX** (clear X); then key in the number correctly. Your stack registers now look like this:

Name	Register
T	0.00
Z	0.00
Y	0.00
X	314.32

In order to key in a second number, you must tell the calculator that you're done with the first number. For example, if you were to key in 567 right now, the number in the displayed X-register would be 314.32567 and the calculator would still not know if you were through. (It's clever, but it can't read your mind.)

One way to tell the calculator you're through with a number is to press **ENTER↑**.* When you press **ENTER↑**, the contents of the registers are changed

	from this:		to this:
T	0.00	T	0.00
Z	0.00	Z	0.00
Y	0.00	Y	314.32
X	314.32	X	314.32

As you can see, the number in the displayed X-register is copied in Y. *(The numbers in Y and Z have also been transferred to Z and T, respectively, and the number in T has been lost. But this will be more apparent when we have different numbers in all four registers.)*

Immediately after pressing **ENTER↑**, the X-register is prepared for a new number. And that new number writes over the number in X. For example, key in the number 543.28 and the contents of the stack registers change

	from this:		to this:
T	0.00	T	0.00
Z	0.00	Z	0.00
Y	314.32	Y	314.32
X	314.32	X	543.28

CLX also prepares the displayed X-register for a new number by replacing any number in the display with zero. Any new number then writes over the zero in X. For example, if you pressed **CLX** now, the stack would change

*A detailed discussion on number termination can be found in appendix B.

from this:

T	0.00
Z	0.00
Y	314.32
X	543.28

to this:

T	0.00
Z	0.00
Y	314.32
X	0.00

And if you then keyed in 689.4, the stack would change

from this:

T	0.00
Z	0.00
Y	314.32
X	0.00

to this:

T	0.00
Z	0.00
Y	314.32
X	689.4

Note that the numbers in the stack do not move when a new number is keyed in immediately after pressing ENTER↑ or CLX .

Simple Arithmetic

Hewlett-Packard calculators do arithmetic by positioning the numbers in the stack the same way you would on paper. For instance, if your numbers were 34 and 21 you would write 34 on a piece of paper and then write 21 underneath it like this:

$$\begin{array}{r} 34 \\ \underline{21} \end{array}$$

and *then* you'd add like this:

$$\begin{array}{r} 34 \\ \underline{+21} \\ 55 \end{array}$$

Numbers are positioned the same way in the HP-65.

Here's how it is done:
(Clear the previous number entry first by pressing CLX .)

Press	See Displayed	
34	34.	34 is keyed into X.
ENTER↑	34.00	34 is copied into Y.
21	21.	21 writes over the 34 in the display.

Now 34 and 21 are sitting vertically in the stack, so we can add.

+	55.00	The answer.

The simple old-fashioned math notation explains how to use your calculator. Both numbers are always keyed in first and then the operation is executed. ***There are no exceptions to this rule.***

Subtraction, multiplication, and division work the same way. In each case, the data must be in the proper position before the operation can be performed.

To subtract 21 from 34 $\left(\begin{array}{r} 34 \\ \underline{-21} \end{array}\right)$:

Press	See Displayed	
34	34.	34 is keyed into X.
ENTER↑	34.00	34 is entered into Y.
21	21.	21 writes over the 34 in X.
−	13.00	Answer.

To multiply 34 by 21 $\left(\begin{array}{r} 34 \\ \underline{\times 21} \end{array}\right)$:

Press	See Displayed	
34	34.	34 is keyed into X.
ENTER↑	34.00	34 is entered into Y.
21	21.	21 writes over the 34 in X.
×	714.00	Answer.

To divide 34 by 21 $\left(\frac{34}{21}\right)$:

Press	See Displayed	
34	34.	34 is keyed into X.
ENTER↑	34.00	34 is entered into Y.
21	21.	21 writes over the 34 in X.
÷	1.62	Answer.

Arithmetic and the Stack

You've already learned how to enter numbers into the calculator and perform calculations with them. In each case you needed to position the numbers in the stack manually. However, the stack also performs many movements automatically. It's these automatic movements that give the stack its tremendous computing efficiency and ease of use. The stack automatically "lifts" every calculated answer in the stack when a new number is keyed in because it *knows* when it completes a calculation that any digits you key in are a part of a new number. For example, calculate $16 + 30 + 11 + 17 = ?$

> **Note:** For the purposes of the remaining examples, it is assumed that the stack is cleared of the previous problem. You can do this yourself by pressing f STK.

Press	Stack Contents		Comments
	T	0.00	
16	Z	0.00	16 is keyed into the displayed X-register.
	Y	0.00	
	X	16.	
	T	0.00	
ENTER↑	Z	0.00	16 is copied into Y.
	Y	16.00	
	X	16.00	

Press	Stack Contents		Comments
	T	0.00	
30	Z	0.00	30 writes over the 16 in X.
	Y	16.00	
	X	30.	
	T	0.00	
+	Z	0.00	16 and 30 are added together. The answer, 46, is displayed.
	Y	0.00	
	X	46.00	
	T	0.00	
11	Z	0.00	11 is keyed into the displayed X-register. The 46 in the stack is automatically raised.
	Y	46.00	
	X	11.	
	T	0.00	
+	Z	0.00	46 and 11 are added together. The answer, 57, is displayed.
	Y	0.00	
	X	57.00	
	T	0.00	
17	Z	0.00	17 is keyed into the displayed X-register. 57 is automatically entered into Y.
	Y	57.00	
	X	17.	
	T	0.00	
+	Z	0.00	57 and 17 are added together for the final answer.
	Y	0.00	
	X	74.00	

After any calculation or number manipulation, the stack automatically lifts when a new number is keyed in. *(See Number Termination in appendix B.)*

In addition to the automatic stack lift after a calculation, the stack automatically drops *during* every calculation involving both X- and Y-registers. It happened in the above example, but let's do the problem differently to see this feature more clearly.

	Stack Contents		Comments
	T	0.00	
	Z	0.00	16 is keyed into the displayed X-register.
16	Y	0.00	
	X	16.	
	T	0.00	
	Z	0.00	16 is copied into Y.
ENTER↑	Y	16.00	
	X	16.00	
	T	0.00	
	Z	0.00	30 is written over the 16 in X.
30	Y	16.00	
	X	30.	
	T	0.00	
	Z	16.00	30 is entered into Y. 16 is lifted up to Z.
ENTER↑	Y	30.00	
	X	30.00	

Press	Stack Contents		Comments
	T	0.00	
	Z	16.00	11 is keyed into the displayed X-register.
11	Y	30.00	
	X	11.	
	T	16.00	
ENTER↑	Z	30.00	11 is copied into Y. 16 and 30 are lifted up to Z and T respectively.
	Y	11.00	
	X	11.00	
	T	16.00	
	Z	30.00	17 is written over the 11 in X.
17	Y	11.00	
	X	17.	
	T	16.00	
+	Z	16.00	17 and 11 are added together and the rest of the stack drops! 16 is duplicated in T and Z. 30 and 28 are ready to be added.
	Y	30.00	
	X	28.00	
	T	16.00	
	Z	16.00	30 and 28 are added together and the stack drops again. Now 16 and 58 are ready to be added.
+	Y	16.00	
	X	58.00	
	T	16.00	
+	Z	16.00	16 and 58 are added together for the final answer and the stack continues to drop.
	Y	16.00	
	X	74.00	

This same dropping action also occurs with **−**, **×**, and **÷**.* The number in T is duplicated in T and Z, the number in Z drops to Y, and the numbers in Y and X combine to give the answer, which is visible in the X-register.

Left to Right Execution

The automatic stack lift and automatic stack drop let you retain and position intermediate results without reentering the numbers. This is the great advantage the stack has over all other data handling methods. As a matter of fact, Hewlett-Packard calculators are the only pocket calculators with a specially designed system for evaluating algebraic expressions with maximum efficiency and overall ease of use. Many problems can be solved by keying in the numbers in left to right order. For example:

$$(35 + 45) \times (55 + 65)$$

Press	See Displayed	
35	35.	The left-most number is keyed into the X-register.
ENTER↑	35.00	No operations can be performed so press ENTER↑.
45	45.	The next number is keyed into X.
+	80.00	The intermediate result of the addition operation is displayed.
55	55.	The next number is keyed into X.
ENTER↑	55.00	The multiplication operation cannot be performed yet, so you press ENTER↑.
65	65.	The next number is keyed into X.
+	120.00	The addition operation is performed next.
×	9600.00	The answer is calculated without repositioning the numbers.

*The stack also drops during yˣ, f D.MS+, and f⁻¹ D.MS+ operations, which are discussed later.

Of course, you don't have to work problems from left to right. Many people start in the middle and only key in numbers as they need them. Either way, the more complex the problem, the more you'll appreciate the capabilities of the operational stack. Try these additional examples.

Sample Case: Calculate $5 \times [(3 \div 4) + (5 \div 2) + (4 \div 3)] \div (3 \times .213)$

Press	See Displayed	
3	3.	
ENTER↑	3.00	
4	4.	
÷	0.75	$(3 \div 4)$
5	5.	
ENTER↑	5.00	
2	2.	
÷	2.50	$(5 \div 2)$
+	3.25	$(3 \div 4) + (5 \div 2)$
4	4.	
ENTER↑	4.00	
3	3.	
÷	1.33	$(4 \div 3)$
+	4.58	$(3 \div 4) + (5 \div 2) + (4 \div 3)$
3	3.	
ENTER↑	3.00	
.213	.213	
×	0.64	$(3 \times .213)$
÷	7.17	
5	5.	The first number is keyed in.
×	35.86	The Answer.

Constant Arithmetic

Sample Case: The growth of $1000 invested at 10% per period would constitute a geometric series in which the first term is 1000 and the growth factor is 1.10. Follow the example below to calculate the first six periods of growth and watch your savings grow!

Press	See Displayed	
1.10	1.10	Growth factor.
ENTER↑	1.10	
ENTER↑	1.10	
ENTER↑	1.10	Growth factor now in T.
1000	1000.	Original amount.
×	1100.00	Amount after 1 period.
×	1210.00	Amount after 2 periods.
×	1331.00	Amount after 3 periods.
×	1464.10	Amount after 4 periods.
×	1610.51	Amount after 5 periods.
×	1771.56	Amount after 6 periods.

What we've done is put the growth factor *(1.10)* in the Y-, Z-, and T-registers and put the first term *(1000)* in the X-register. Thereafter, you get the next term whenever you press ×. For example, when you press × the first time, you calculate 1.10 × 1000. The result *(1100.00)* is displayed in the X-register and a new copy of the growth factor drops into the Y-register. Since a new copy of the growth factor is generated in T each time the stack drops, you never have to reenter it.

Manipulating Numbers

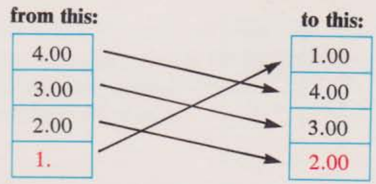
ENTER↑ is not the only key that positions numbers in the stack. The g R↑ (roll up), g R↓ and g x⇄y (roll down) key sequences reposition numbers in the stack without any danger of losing numbers from the T-register.

Rotating the Stack

The g R↑ and g R↓ keys let you review the entire stack contents at any time. To see how these key sequences work, load the stack with the numbers 1 through 4 by pressing:

4 ENTER↑ 3 ENTER↑ 2 ENTER↑ 1

If you then press g R↑, the stack contents are rotated



Now watch the stack contents that follow as we use the g R↑ and g R↓ keys to bring numbers in the stack one-by-one into the displayed X-register.

Press	Stack Contents		Comment
	T	2.00	
g R↓	Z	1.00	Once again all of the numbers are rearranged in the stack. 3.00 is now in the displayed X-register.
	Y	4.00	
	X	3.00	
	T	3.00	
g R↓	Z	2.00	The numbers are rotated down one by one again. 4.00, which was in the T-register, is now in the X-register.
	Y	1.00	
	X	4.00	

Keys		Stack	
g R↓	T	4.00	All the numbers are back in the registers they started in. No numbers have been lost.
	Z	3.00	
	Y	2.00	
	X	1.00	

Keys		Stack	
g R↑	T	3.00	The number in T replaces the number in the X-register this time. The other numbers shift up one place.
	Z	2.00	
	Y	1.00	
	X	4.00	

Keys		Stack	
g R↑	T	2.00	And again the numbers are rearranged in the stack. 3.00 is now in the displayed X-register.
	Z	1.00	
	Y	4.00	
	X	3.00	

Keys		Stack	
g R↑	T	1.00	The numbers are rotated up one place again. 2.00, which was in the T-register, is now in the displayed X-register.
	Z	4.00	
	Y	3.00	
	X	2.00	

Keys		Stack	
g R↑	T	4.00	All the numbers are back where they started. No numbers have been lost.
	Z	3.00	
	Y	2.00	
	X	1.00	

These keys are used primarily to position numbers in the stack. However, if you're unsure of the contents of the stack, use g R↓ and g R↑, as we have done here, to verify the location of the data.

Exchanging X and Y

The g x⇄y (x exchange y) keys exchange the contents of the X- and Y-registers without affecting Z- and T-registers. If you press g x⇄y with the data intact from the previous example, the numbers in the X- and Y-registers will be changed

from this:	to this:
4.00	4.00
3.00	3.00
2.00	1.00
1.00	2.00

Similarly, pressing g x⇄y again will restore the numbers in the X- and Y-registers to their original places. These keys are used to position numbers in the stack or simply to view the Y-register.

D and E

You may notice that R↓ and x⇄y are also available on the D and E keys when the power is first switched ON. The five functions shown in the window were selected because they are the most commonly used. Their primary intent is for manual use from the keyboard. They each permit single keystroke operation of functions that otherwise would require two keystrokes. When the A thru E keys are redefined by a program (*or whenever* f PRGM *has been pressed*), the window functions are still available by two keystrokes.

General Operations

In this section we will describe how to: ▪ perform the clear operations ▪ control the display ▪ enter negative numbers and numbers in scientific notation ▪ recover from wrong keystrokes using the Last X feature ▪ recall π ▪ use the addressable storage registers.

> **Note:** Lower-case letters are used to denote values in corresponding registers; e.g., "x" for the value in the X-register. Upper-case letters are used to denote the register itself.

Clear Operations

Five separate clearing operations are available on your HP-65, using the f functions of the fourth row of keys.

Clearing Unwanted Prefix

f PREFIX cancels the effect of a prefix so that a non-prefix operation can be done. Let's say you accidentally press f, f^{-1}, or g, before keying in a number. If you then press the number key, you will get an alternate function of that key instead of the desired number-entry operation. To prevent this from happening, press f PREFIX to cancel the effect of the unwanted prefix key, then key in a number. If a wrong prefix key is pressed when another prefix is wanted, the error can be corrected by simply pressing the correct prefix and proceeding from there.

The above procedure can also be used to clear these additional keys:

STO, RCL, DSP, GTO, LBL, STO ·, RCL ·

Clearing Stack Registers

f STK clears all four registers (*X, Y, Z, and T*) of the operational stack. Although this operation may be comforting at times, it is never really necessary and is provided only as a convenience. To clear only the X-register, press CLX.

Clearing Addressable Registers

f REG clears all nine addressable registers. Be sure these are cleared before doing storage register arithmetic.

Clearing Entire Calculator

The entire calculator can be completely cleared by turning the power switch OFF, then ON. When the power comes on, however, default programs for the functions corresponding to the window legends above the top row keys (1/x, √x, y^x, R↓, x⇄y) will be automatically placed in program memory.

Clearing Program Memory

f PRGM clears the HP-65's 100-step program memory but is effective only when the W/PRGM-RUN switch is in W/PRGM position. In RUN position, f PRGM has the same effect as CLX.

Display

The display is used to show results, operational errors, low battery condition, programs in execution, and in W/PRGM mode the display allows you to "see" each step of a program in memory (*this use of the display will be described in section 4*).

Setting Display

The HP-65 displays up to 15 characters: mantissa sign, 10-digit mantissa, decimal point, exponent sign, and 2-digit exponent. In RUN mode, the display shows a rounded version of the number in the X-register. Two display modes (*fixed and scientific notation*) with a variety of rounding options may be selected from the keyboard. (*Rounding options affect the display only; the HP-65 always maintains full accuracy internally.*)

Fixed Display. Fixed notation is specified by pressing DSP · followed by the appropriate number key to specify the number of decimal places (*0–9*) to which the display is to be rounded. Fixed notation allows all answers to be displayed with the same precision. The display is left-justified and includes trailing zeros within the selected setting. When the calculator is turned OFF,

then ON, it always reverts to fixed notation with the display rounded to two decimal places. For example:

Press	See Displayed
(*Make sure W/PRGM-RUN switch is set to RUN. Turn the calculator OFF, then ON.*)	0.00
123.4567	123.46
DSP · 4	123.4567
DSP · 6	123.456700
DSP · 2	123.46
DSP · 0	123.

Scientific Display. This is useful when you are working with large or very small numbers and allows answers to be displayed with the same number of significant digits. It is specified by pressing DSP followed by the appropriate number key to specify the number of decimal places to which the mantissa is rounded. Again, the display is left-justified and includes trailing zeros within the selected setting. For example:

Press	See Displayed	
(Turn the calculator OFF, then ON.)	0.00	
123.4567 ENTER↑	123.46	
DSP 2	1.23 02	Equals 1.23×10^2.
DSP 4	1.2346 02	Equals 1.2346×10^2.
DSP 8	1.23456700 02	Equals 1.234567×10^2.

Next, set the display to show eight decimal places in fixed notation:

Press	See Displayed	
DSP · 8	1.234567000 02	*Equals 1.23456700×10^2.

* If a number is too large to fit the specified display, the number is displayed in full (10 digit) scientific notation.

Now return to two decimal places in fixed notation:

Press	See Displayed
DSP [•] [2]	123.46
[•] 0005 CHS ENTER↑	−0.00 (*)

Blinking Display

The display blinks when any of several improper operations are attempted. Pressing any key stops the blinking without otherwise performing the key function. CLX is the recommended blink stopper. Figure 2-1 lists these improper operations.

Illegible Display

During execution of a stored program, the display continuously changes and is purposely illegible to indicate that the program is running. When the program stops, the display is steady.

Multiple Decimal Point Display

The battery provides approximately 3 hours of continuous operation. By turning off the power when the calculator is not in immediate use, the battery power will be conserved. To conserve power without losing program or results, leave the calculator on, key in a [•], and leave it there until ready to resume calculation.

All decimal points light in the display when 2 to 5 minutes of operation time remain in the battery pack. Even when all decimal points are turned on, the true decimal position is known because an entire digit position is allocated to it.

True Decimal Position

*If a result develops that is too small to be expressed in the specified display, zero is displayed (with minus sign in case of a negative result).

Keys	Function	Error
f LN	Natural log (base e)	$x \le 0$
f LOG	Common log (base 10)	$x \le 0$
f √x	Square root	$x < 0$
f^{-1} SIN	Arc sine	$\|x\| > 1$
f^{-1} COS	Arc cosine	$\|x\| > 1$
{ f / f^{-1} } D.MS+	{ Add / Subtract } degrees, minutes, seconds	$\|x\|$ or $\|y\|$ or $\|y \pm x\| > 99999.59599$ D.MS
{ f / f^{-1} } →D.MS	Convert angle expressed decimally to/from degrees, minutes, seconds	$\|x\| > 99999.99999$ degrees or equivalent in radians or grads
f →OCT	Decimal to octal	x is noninteger or $\|x\| > 1073741823_{10} = 7777777777_8$
f^{-1} →OCT	Octal to decimal	x is noninteger or $\|x\| > (12222222221)_8 = 9999999999_8 = 1380525201_{10}$
g 1/x	Reciprocal	$x = 0$
g y^x	Exponential	$y \le 0$
g n!	Factorial	x is noninteger or $x < 0$
÷	Divide	$x = 0$
	Magnetic card read	Blank card; bit or word dropped during reading

Figure 2-1. *Blinking Display Errors*

If the decimal points light while the reader/writer motor is running and then go out, the battery is almost discharged.

Operating the calculator for more than 2 to 5 minutes after this low power indication first occurs may result in wrong answers. The battery pack must be replaced or recharged by connecting the calculator to the battery charger. Be sure to start with at least partially charged batteries before using the card reader/writer.

Negative Numbers

To key in a negative number, press CHS (change sign) after keying in the positive value. For example, to key in —12:

Press: 12 CHS

To change the sign of a negative or positive number, press CHS . For example, to change the previous number back to a positive 12:

Press: CHS

Keying in Large and Small Numbers

You can key in numbers having power of ten multipliers (scientific notation) by pressing EEX (enter exponent). For example, key in 15.6 trillion (15.6×10^{12}), and multiply it by 25.

Press	See Displayed	
15.6 EEX	15.6 00	
12	15.6 12	15.6×10^{12} *
ENTER↑	1.560000000 13	1.56×10^{13}
25 ×	3.900000000 14	Answer

*To key in a negative number (e.g., -15.6×10^{12}) you would press CHS before pressing EEX .

Exact Powers of Ten

You can save time when keying in exact powers of ten by pressing EEX and then pressing the desired power of ten. For example, key in 1 million (10^6) and divide by 52.

Press	See Displayed
EEX 6	1. 06
ENTER↑	1000000.00
52 ÷	19230.77

Small Numbers (Negative Exponents)

To key in negative exponents, key in the number, press EEX , press CHS to make the exponent negative, then key in the power of ten. For example, key in Planck's constant (h) — roughly, 6.625×10^{-27} erg-s — and multiply it by 50.

Press	See Displayed
6.625 EEX	6.625 00
27	6.625 27
CHS	6.625 −27
ENTER↑	0.00
DSP 6	6.625000 −27
50	50
×	3.312500 −25
DSP · 2	0.00

Regardless of the display format, the number (6.625×10^{-27} in this case) is maintained internally to an accuracy of 10 digits.

Last X

Last X is the name of the register reserved for storing the last number displayed that precedes the last function performed. Last X is set to zero when you switch the calculator ON and it remains unchanged until a calculation is performed. At such time the number displayed is saved in Last X as an automatic prelude to the calculation. The saved value is recallable to the X-register (repeatedly, if desired) by pressing g LSTX.

Last X is particularly useful in expressions like the following:

$$\frac{\sin x}{x}, \quad y^x - \sqrt{x}, \quad \sin x + \cos^3 x$$

Let's try the first expression in an example to see how this works.

Sample Case. Calculate $\frac{\sin x}{x}$ for $x = 52.47°$. (*Assume degrees mode is set.*)

Press	See Displayed
52.47	52.47
f SIN	0.79
g LSTX	52.47
÷	0.02

Last X is also useful in recovering from accidental wrong keystrokes such as pressing the wrong arithmetic key or entering a wrong number. For example, if you were performing a long calculation where you meant to subtract 3 from 12 and you divided instead, you could compensate as follows:

Press	See Displayed	
12 ENTER↑ 3 ÷	4.00	Oops — you wanted to subtract.
g LSTX	3.00	Retrieves last number preceding division operation.
×	12.00	Reverses division operation: you are back where you started.
g LSTX	3.00	Retrieves last number displayed before multiplication operation.
−	9.00	Correct operation produces desired results.

If you want to correct a number in a long calculation, Last X can save you from starting over. For example, divide 12 by 2.157 after you have divided by 3.157 by mistake.

Press	See Displayed	
12 ENTER↑ 3.157 ÷	3.80	You wanted to divide by **2.157**, not **3.157.**
g LSTX	3.16	Retrieves last number displayed preceding operation.
×	12.00	You're back at the beginning.
2.157 ÷	5.56	Correct operation produces desired results.

The following operations (*including inverses*) save the X value in Last X: +, −, ×, ÷, →D.MS, D.MS+, INT, LN, LOG, →OCT, R→P, SIN, COS, TAN, n!, √x, 1/x, yˣ, ABS. Note that: CLX, STO + n, STO − n, STO × n, STO ÷ n, x≠y, x≤y, x=y, and x>y do not affect the Last X register.

Recalling π

π is a fixed constant provided in your HP-65. Merely press g π whenever you need it in a calculation.

Sample Case: Calculate the area of a circle with a radius of 3. Area $= \pi 3^2$.

Press	See Displayed	
g π	3.14	Recall π to X.
3 ENTER↑ ×	9.00	Calculate 3 × 3.
×	28.27	The answer.

Addressable Registers

Registers R_1 thru R_9 constitute the addressable registers. Their respective contents are referred to as $r_1, r_2, \ldots, r_9$. Operations refer to them by number. The registers are typically used to accumulate sums or to store constants or intermediate results. You can store the value of the stack's X-register in any addressable register, or you can recall the value in any addressable register to the X-register. Additionally, you can calculate in any register an arithmetic sum, difference, product, or quotient of the contents of the given register and the X-register.

Storing and Recalling Data

To store a number appearing in the display (*whether the result of a calculation or keystroke entry*):

1. Press STO .
2. Press a number key 1 thru 9 to specify in which of the nine registers the number is to be stored.

If the selected storage register already has a number in it, the old number will be overwritten by the new one. The value in X will remain unchanged.

To recall a number previously stored in one of the nine addressable memory registers:

1. Press RCL .
2. Press a number key (1 thru 9) to specify which of the nine registers the number is to be recalled from.

Recalling a number does not remove it from the storage register. Rather, a copy of the stored number is transferred to the display —the original remains in the storage register until either: (1) a new number is stored in the same register, (2) the calculator is turned OFF, or (3) all nine storage registers are cleared by pressing f REG . Recalling a number from a register will cause the stack to lift unless preceded by CLX or ENTER↑ .

Sample Case 1. A customer has bought three items priced at \$1,000, \$2,000, and \$3,000, respectively. Your policy is to grant a 5% discount on all purchases over \$500. How much will the customer pay for each of the three items? What is the total cost?

Solution:

Press	See Displayed	
1 ENTER↑ .05 − STO 1	0.95	Stores constant 0.95 (95%) in register R_1.
1000 RCL 1 ×	950.00	Amount customer will pay for first item.
2000 RCL 1 ×	1900.00	Amount customer will pay for second item.
3000 RCL 1 ×	2850.00	Amount customer will pay for third term.
+ +	5700.00	Total cost.

Sample Case 2. The capacity and height of three tanks are listed below in U.S. units. What is the capacity and height of each tank in metric units?

	Capacity (gal.)	**Height** (in.)
Tank 1	3.6	13.5
Tank 2	5.5	20.9
Tank 3	11.3	32.8

Remember that: 1 U.S. gallon = 3.7854 liters
1 inch = 2.5400 centimeters

We will store these constants in R_1 and R_2.

Solution:

Press	See Displayed	
DSP · 4	0.0000	Set display.
3.7854 STO 1	3.7854	Stores liters/gallons conversion constant in R_1.
2.54 STO 2	2.5400	Stores centimeters/inch conversion constant in R_2.
3.6 RCL 1 ×	13.6274	Capacity of tank 1 in liters.
13.5 RCL 2 ×	34.2900	Height of tank 1 in centimeters.
5.5 RCL 1 ×	20.8197	Capacity of tank 2 in liters.
20.9 RCL 2 ×	53.0860	Height of tank 2 in centimeters.
11.3 RCL 1 ×	42.7750	Capacity of tank 3 in liters.
32.8 RCL 2 ×	83.3120	Height of tank 3 in centimeters.
DSP · 2	83.31	Resets display.

Choosing Addressable Registers

Except for the case of registers R_8 and R_9, it is immaterial which registers you use.

R_8 is the special object of the g DSZ operation (*presented in section 4*), which uses it as a descending counter (*index*) in program applications. R_8 should be avoided for other uses when g DSZ is used in your programs.

R_9 is subject to alteration by the trigonometric functions, rectangular/polar conversions, and the relational tests (*used in programs*). The trigonometric functions and rectangular/polar conversions use R_9 for intermediate calculations. When executing a relational test, R_9 serves as a Last X register. At other times R_9 is available for your use.

Calculating in Addressable Registers

Thus far, all calculations have involved the X-register or the X- and Y-registers to produce a result in X. In the case of addressable register arithmetic, the result is left in the addressable register and the number in X is unchanged.

Subtraction.	To subtract the number in X from r_n, press:	STO − n
Addition.	To add the number in X to r_n, press:	STO + n
Multiplication.	To multiply the number in X by r_n, press:	STO × n
Division.	To divide the number in X into r_n, press:	STO ÷ n

For example, store 6 in register R_1 and then increment it by 2.

Press	See Displayed	
6 STO 1	6.00	Stores 6 in R_1.
2 STO + 1	2.00	Adds 2 to r_1.
RCL 1	8.00	Confirms that r_1 equals 8.

Now, subtract 5 from the contents of R_1.

5 STO − 1	5.00	
RCL 1	3.00	Confirms that r_1 has been reduced to 3.

Finally, multiply the remaining contents of R_1 by 2:

2 STO × 1	2.00	
RCL 1	6.00	Confirms that r_1 has been increased to 6.

Section 3

Functions

You have already learned to use the arithmetic functions (+, −, ×, ÷) in both the stack and the addressable registers. You have also learned to move numbers among the calculator's registers and to enter and display data in both fixed and scientific format. To complete the subject of manual calculation, we will return to the non-arithmetic functions, things like sine, logarithm, square root . . .

Keys Introduced in this Section

ABS	→D.MS	INT	n!	R→P	1/x
COS	D.MS→	LN	→OCT	SIN	√x
DEG	GRD	LOG	RAD	TAN	y^x

These functions are both easy to learn and easy to use. In the introduction you learned to execute a function by pressing prefix key (f, f^{-1}, or g) and following it with the desired function key: you use the g prefix to calculate a function having a blue symbol, you use f to calculate a function having a gold symbol, and you use f^{-1} to calculate the inverse (or complement) of the function denoted by a gold symbol.

As might be expected, the x's and y's you see on the keyboard for these functions refer to the contents of the X- and Y-registers. For example, y^x means raise the number in the Y-register to the power of the number in the X-register.

Figures 3-1, 3-2, and 3-3 present a systematic review of which functions are available and the respective conditions that apply to each of them. To calculate a given function, the respective table entry shows any conditions that apply to the input value(s), the keys to use, and conditions applying to the result(s). If your need is to start calculation immediately, you might even end your study of functions with the tables, skipping the sample cases.

Functions Involving Angles

These functions are listed in figure 3-1. They include the trigonometric functions (*sine, cosine, tangent and their inverses*), the rectangular/polar conversions, the addition and subtraction of angles expressed in degrees, minutes, seconds, and conversions of angles expressed decimally to and from degrees, minutes, and seconds.

Angular Mode

Operations involving these functions assume the angles to be expressed in units of the prevailing *angular mode*, which is set to *decimal degrees* whenever the calculator is switched on. You can set the mode to *radians* or *grads* or *decimal degrees* by using the mode functions.

Angular Mode Functions

Keys	Function
g GRD	Set mode to grads
g RAD	Set mode to radians
g DEG	Set mode to degrees

400 grads = 360 degrees = 2 π radians

Keys to which Angular Mode applies:

SIN COS TAN R→P →D.MS

In the examples, the *degree* mode is assumed except as noted otherwise.

Degrees, Minutes, Seconds

You can convert from the decimal form of an angle to degrees, minutes, seconds. You can also do the inverse. When converting from the decimal form of the angle to degrees, minutes, seconds,

Keys	Function	Input Value(s)	Result(s)
f COS	Cosine	Angle*	Cosine (x) in X
f⁻¹ COS	Arc cosine	x not be greater than 1 or less than -1 ($\lvert x\rvert \leq 1$)	Principal value of arc cosine (x) in X ($0° \leq$ result $\leq 180°$)**
f SIN	Sine	Angle*	Sine (x) in X
f⁻¹ SIN	Arc sine	x not be greater than 1 or less than -1 ($\lvert x\rvert \leq 1$)	Principal value of arc sine (x) in X ($-90° \leq$ result $\leq 90°$)**
f TAN	Tangent	Angle*	Tangent (x) in X
f⁻¹ TAN	Arc tangent	Unrestricted x	Principal value of arc tangent (x) in X ($-90° \leq$ result $\leq +90°$)**
f R→P	Convert rectangular coordinates (x, y) to polar form (r, θ)	x, y in X, Y	r, θ* in X, Y
f⁻¹ R→P	Convert polar coordinates (r, θ) to rectangular form (x, y)	r, θ in X, Y	x, y in X, Y. (Program halt on underflow in X.)
f →D.MS	Convert decimal angle to DDDDD.MMSS format***	Decimal angle****	DDDDD.MMSS in X
f⁻¹ →D.MS	Convert DDDDD.MMSS*** angle to decimal format	DDDDD.MMSS	Decimal angle**** in X
f D.MS+	Add (y+x) in DDDDD.MMSS format***	y: } x: } DDDDD.MMSS****	DDDDD.MMSS in X (Sum)****
f⁻¹ D.MS+	Subtract (y−x) in DDDDD.MMSS format***	y: } x: } DDDDD.MMSS****	DDDDD.MMSS in X (Difference)****

* Decimal angle in prevailing angular mode.
** Or equivalent in grads or radians.
*** DDDDD.MMSS format. D = degrees, MM = minutes, SS = seconds.
**** Magnitude of angle should not exceed 99999.99999 decimal degrees (or equivalent in radians or grads) or 99999.5959 in DDDDD.MMSS format.

Figure 3-1. *Functions Involving Angles*

all 10 digits are evaluated. When converting from degrees, minutes, seconds to the decimal form of the angle, the angle is rounded to the nearest second before the conversion is made. The format for degrees, minutes, and seconds is DDDDD.MMSS. Thus, you use DSP · 4 to display this format. This function depends on the mode setting as illustrated below.

Sample Case Part 1. Convert $\frac{\pi}{7}$ radians to degrees, minutes, seconds.

Press	See Displayed	
DSP · 4	0.0000	Set display.
g π 7 ÷	0.4488	π/7
g RAD	0.4488	Set radian mode.
f →D.MS	25.4251	Answer: 25° 42′ 51″.

Sample Case Part 2. Now do the inverse, but converting back to grads *(instead of radians)*.

> **Note:** This method allows you to convert between angle modes, i.e. decimal degrees ⇄ radians, decimal degrees ⇄ grads, radians ⇄ grads.

Press	See Displayed	
g GRD	25.4251	Set grad mode.
f⁻¹ →D.MS	28.5713	Answer in grads.

Sample Case: *Adding/Subtracting DDDDD.MMSS.* Find the sum of 45° 10′ 50″ and 44° 49′ 10″.

Press	See Displayed	
DSP · 4	0.0000	Set display.
45.1050	45.1050	Key in first angle to X.
ENTER↑	45.1050	
44.4910	44.4910	Key in second angle.
f D.MS+	90.0000	Answer, 90° 00′ 00″.

A musical selection begins at 9:25′ 7″ and ends at 9:39′ 47″. How long is the piece?

Press	See Displayed	
DSP · 4	0.0000	Set display.
9.3947	9.3947	Completion time.
ENTER↑	9.3947	
9.2507	9.2507	Starting time.
f⁻¹ D.MS+	0.1440	Answer, 14′ 40″ duration.
DSP · 2	0.14	Reset display to two places.

Sample Case: *Trigonometric Functions.* Compute cosine 60°.

Press	See Displayed	
g DEG 60	60.	
f COS	0.50	Answer.

Compute *arc cosine* (—1.) expressed in radians.

Press	See Displayed	
g RAD 1 CHS	−1.	
f⁻¹ COS	3.14	Answer in radians.

Compute *sine* 30°.

Press	See Displayed	
g DEG 30	30.	
f SIN	0.50	Answer.

Compute *arc sine* (1.00) expressed in radians.

Press	See Displayed	
g RAD 1	1.	
f⁻¹ SIN	1.57	Answer in radians.

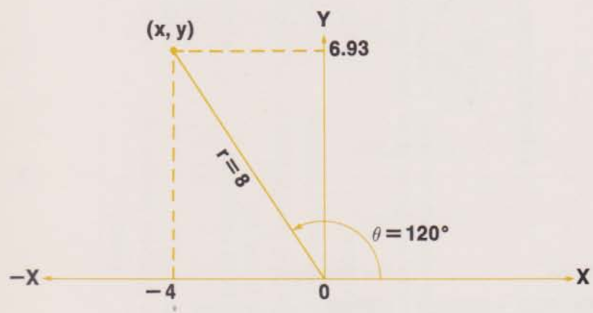
Compute *tangent* 45°

Press	See Displayed	
g DEG 45	45.	
f TAN	1.00	Answer.

Compute *arc tangent*(39.4), expressed in radians.

Press	See Displayed	
g DEG 39.4	39.4	
f⁻¹ TAN	1.55	Answer in radians.

Sample Case: *Polar to Rectangular**. Convert polar coordinates (r=8, θ=120°) to rectangular coordinates:

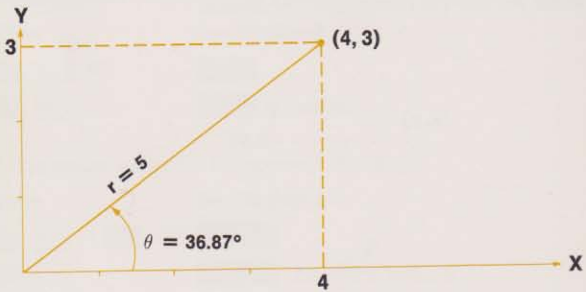


*Note that if r is equal to 1.00, then x is equal to sinθ and y is equal to cosθ; a fact that is often useful in programming applications.

Underflow in polar to rectangular conversion may leave out-of-range values in Y. When these values are brought to the X-register, they are set to zero; an executing program halts.

Press	See Displayed	
g DEG	0.00	
120 ENTER↑	120.00	θ
8	8.	r
f^{-1} R→P	−4.00	x coordinate.
g x⇄y	6.93	y coordinate.

Sample Case: *Rectangular to Polar**. Convert rectangular coordinates (x=4, y=3) to polar form with the angles expressed in degrees:



Press	See Displayed	
g DEG 3 ENTER↑	3.00	y coordinate.
4	4.	x coordinate.
f R→P	5.00	r (magnitude).
g x⇄y	36.87	θ (angle).

* Rectangular to polar can be used to calculate the arc tangent of Y/X. The advantage of using rectangular to polar for this calculation is that the resultant angle is automatically resolved to the proper quadrant.

Keys	Function	Input Value(s)	Result
f →OCT	Convert decimal integer to octal (base 8).	x_{10} a decimal integer of magnitude less than 1073741824_{10}	x_8 in X
f^{-1} →OCT	Convert octal integer to decimal (base 10).	x_8 an octal integer*	x_{10} in X
f INT	Truncate to signed integer.	±integer.fraction in X	±integer.0 in X
f^{-1} INT	Truncate to signed fraction.	±integer.fraction in X	±0.fraction in X
g ABS	Absolute value.	±x	If −x, change its sign; otherwise, no change.

*As an additional feature, the "octal to decimal" conversion will accept non-octal arguments containing the digits 8 or 9. A non-octal number such as 998 will be interpreted as $(9 \times 8^2) + (9 \times 8) + 8 = 656$

998 f^{-1} →OCT → 656_{10}

656_{10} f →OCT → 1220_8

Figure 3-2. *Conversions of x*

Conversions

The conversions are listed in figure 3-2. The conversions all expect an input value in the X-register. Note that angle conversions are given in figure 3-1.

Sample Case: *Octal/Decimal Conversions.* Many computers are designed to work with octal (*base 8*) numbers instead of decimal (*base 10*) numbers. The [→OCT] function on your HP-65 allows you to make octal/decimal conversions with ease. For example, find the octal equivalent of the decimal number 512.

Press	See Displayed	
512 [f] [→OCT]	1000.00	Octal representation of 512_{10}.

Convert the octal number 2000 to its decimal equivalent:

Press	See Displayed	
2000 [f⁻¹] [→OCT]	1024.00	Decimal equivalent of 2000_8.

Sample Case: *Truncating at Decimal Point.* Some application pac programs expect you to key in dates using the format *mm.yyyy*. The program separates *mm* from *yyyy* using the truncation functions. Do the same for the date 12.1980 *(December 1980).*

Press	See Displayed	
[DSP] [•] [4]	0.0000	Set display.
12.1980	12.1980	Key in date to X.
[f] [INT]	12.0000	Answer: integer part.
[g] [LST X]	12.1980	Recall original value.
[f⁻¹] [INT]	0.1980	Answer: fractional part.
[DSP] [•] [2]	0.20	Reset display to two places.

Sample Case. *Absolute Value.* Some calculations require the magnitude of a number. To get this from the keyboard, you could observe the number and change the sign if negative (*using* [CHS]). From a program, you use the [ABS] function which changes the sign, if negative. For example, calculate the absolute value of 3 and −3.

Press	See Displayed	
3 [g] [ABS]	3.00	$\|+3\|$
[CHS]	−3.00	$\|-3\|$
[g] [ABS]	3.00	

Functions of x and the Exponential Function (y^x)

These functions are listed in Figure 3-3. All expect an input value to be in the X-register. [y^x] expects, in addition, a y value in the Y-register. It is worth noting that the conditions given for INPUT VALUE(S) can generally be predicted by common sense. For example, the table tells us that to calculate the reciprocal, the input value cannot be 0, which is exactly what we would expect because we ordinarily attach no meaning to $1 \div 0$. If we attempt to calculate the reciprocal of zero, the blinking display emphatically warns us of the error. Try it. Just press [CLX] [g] [1/x]. You can stop the blinking by pressing any key.

Sample Case. *Common Logarithm.* Calculate the power gain in decibels of an amplifier yielding twice the value of the input power.

Note: decibels = 10 log (2)

Press	See Displayed	
10 [ENTER↑]	10.00	Save value 10.
2 [f] [LOG]	0.30	Log 2.
[×]	3.01	Answer.

Sample Case: e^x. Display the contant e to nine places ($e = e^1 =$ *natural antilog 1*).

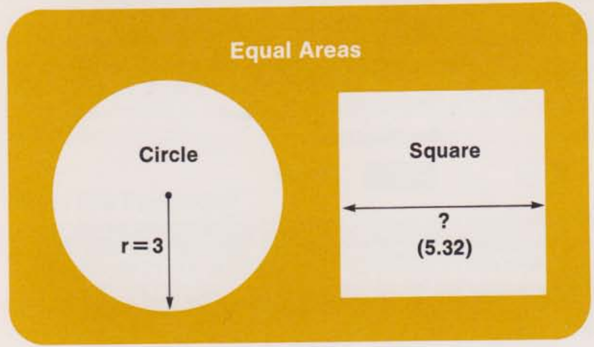
Press	See Displayed	
1 [f⁻¹] [LN]	2.72	
[DSP] [•] [9]	2.718281828	Answer.
[DSP] [•] [2]	2.72	Reset display.

Keys	Function	Input Value(s)	Result
f LN	Natural logarithm (base e)	x not zero or less ($x>0$)	ln (x) in X
f^{-1} LN	Natural antilogarithm (e^x)	Unrestricted x	e^x in X
f LOG	Common logarithm (base 10)	x not zero or less ($x>0$)	log (x) in X
f^{-1} LOG	Common antilogarithm (10^x)	Unrestricted x	10^x in X
f $\sqrt{x}$	Square root ($\sqrt{x}$)	Non negative x ($x\geq 0$)	$\sqrt{x}$ in X
f^{-1} $\sqrt{x}$	Square (x^2)	Unrestricted x	x^2 in X
g 1/x	Reciprocal (1/x)	Non zero x ($x\neq 0$)	1/x in X
g n!	Integer factorial (n!) $n! = 1\cdot 2\cdot 3\cdot \ldots \cdot (n-1)\cdot n$ $0! = 1$	Non negative integer n in x ($x\geq 0$; x an integer)	n! in X
g y^x	Exponential (y^x)	Positive y ($y>0$) and unrestricted x	y^x in X; stack drops.

Figure 3-3. *Functions of x and the Exponential Function (y^x)*

Sample Case: *Square and Square Root*. What size square has the same area as a circle whose radius is 3?

Method. $\pi\times 3^2$ is the area of the circle. The square root of this value gives the side of a square of equal area.



Press	See Displayed	
g π	3.14	π
3 f^{-1} $\sqrt{x}$	9.00	3^2
×	28.27	Area of circle.
f $\sqrt{x}$	5.32	Size of square.

Sample Case: *Reciprocals*. Calculate: $1/4 = .25$.

Press	See Displayed	
4 g 1/x	0.25	Reciprocal of 4.

Naturally, you can use this value in another calculation. For example, to go on and calculate

$$\frac{1}{1/4 + 1/3},$$

1/4 is already calculated.

Press	See Displayed	
	0.25	Reciprocal of 4.
3 g 1/x	0.33	Reciprocal of 3.
+	0.58	Sum of reciprocals.
g 1/x	1.71	Answer: reciprocal of sum.

Sample Case: *Factorial.* Calculate the number of ways 6 people can line up for a photograph.

Press	See Displayed	
6 g n!	720.00	Answer.

Sample Case: *Exponential.* In the preceding section we calculated the successive terms of a geometric series to find that after 6 periods, $1000 invested at 10% grows to $1771.56. Using the y^x function, the same result is obtained by evaluating the following:

$$1000(1.10)^6$$

Press	See Displayed	
1000 ENTER↑	1000.00	Original amount.
1.10 ENTER↑ 6		
g y^x	1.77	$(1.10)^6$
×	1771.56	Answer.

Section 4

Programming

You've finally reached the section that describes the reason you probably bought an HP-65 in the first place–programming! But relax. The keyboard programming language used by the HP-65 is not complicated or difficult to understand. By taking your time and working through the sample programs as you read, you'll progress from writing simple programs like the one you wrote in the introduction to the advanced programs found in the application pacs.

What Is a Program?

A program is nothing more than a sequence of keystrokes stored in the calculator and executed automatically with the press of a button—one keystroke replacing many! In the previous sections of this handbook, whenever an example was done, *you*, the operator, were programmed. You were asked to press keys in a given sequence to obtain a particular result. In most cases, if the sequence was not followed exactly, the result was not correct. Similarly, in a program, the calculator is given a sequence of keystrokes. The calculator "memorizes" the keystroke sequence and then can execute it automatically any number of times, and much faster than you could yourself!

What key sequence do you give the calculator? The bulk of every program you write will be the same keys you would press manually in RUN mode to solve your problem. In fact, from the entire keyboard there are three key sequences that cannot be given to the calculator for later execution:

SST, f PRGM, g DEL

These three key sequences are *the only active operations in W/PRGM mode.* All other keys pressed in W/PRGM mode are stored in program memory to be executed later.

As you know, in RUN mode pressing any key produces an immediate result. However, every operation in RUN mode can be generated in two ways: from the keyboard or from program memory (*if the keys have first been stored in program memory*).

And, **the only keys that work differently** from the keyboard than they do from program memory are:

RTN, A thru E, GTO, R/S

These instructions control program execution and should be studied carefully.

Looking at a Program

Earlier, you may recall, you learned that five functions/operations are accessible in two different ways. You can press g 1/x or A; f √x or B; and so on. The five keys A thru E are used to control program execution. Each key is defined by the program it controls. Default programs for 1/x, √x, yˣ, R↓, and x⇄y are automatically stored in program memory for these five keys when the calculator is switched ON. This is for your convenience when doing manual calculation, so that you can use these common functions and operations *(indicated in white above the* A *thru* E *keys)* by pressing one key instead of two; e.g., A instead of g 1/x. But the A thru E keys can be redefined by any program you choose. The short program you wrote in the introduction is an example of how this is done. You redefined the A key to calculate the cube of a number.

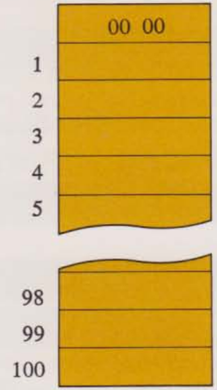
Program Memory

Now let's use these default programs to find out a little more about the program memory of the HP-65. Switch the calculator OFF and then ON again. The A thru E keys are now defined by the default programs. Next, slide the mode switch to W/PRGM (write program). You should see the following display:

Top of Memory Marker

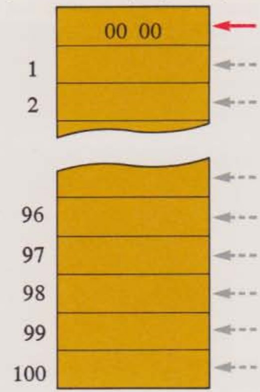
Whenever you see this display, you know that you are at the top of memory. The HP-65 program memory consists of 100 usable steps and a top of memory marker. The following drawing is a

graphic representation of program memory. Notice that the top of memory marker does occupy a step (*not one of your 100*), but that no keys may be stored there. The other steps can store one and sometimes two keystrokes.



Program Pointer

When a program is run, the calculator executes each step sequentially downward by means of a program "pointer."



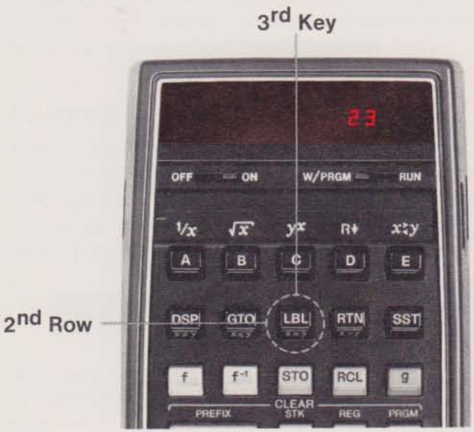
The program pointer takes the place of your finger, pushing the keys one by one. The calculator executes each step as the program pointer points to it.

Single Step

SST (single step) cannot be stored in program memory. In W/PRGM mode, it enables you to review a program one step at a time. Pressing SST advances the program pointer to the next step in memory – showing you the steps but not executing them.

Keycodes. Now let's use the SST key to take a look at the program defining the A key. Press SST one time and the display changes to:

This is the keycode for the first step of the program. How can you tell what key it is? Simply count down 2 rows and count over 3 keys. You should find the LBL (label) key. The codes represent the number of rows down and the number of keys across.



The digit keys are the exception. For ease of recognition, the digit keys 0 thru 9 and the blue and gold functions associated with them are displayed simply as 00 thru 09. Press SST again and the display changes to:

This represents the A key (*first row, first key*). Press SST again and the keycode for the blue prefix key g is displayed:

Again pressing SST changes the display to:

Notice here that because the previous code was for the blue prefix key g, this code will be interpreted by the calculator as 1/x, the blue alternate function of the 4 key. Pressing SST one more time displays the last keycode of the program controlled by the A key which is RTN (return):

As you can see, the default program executed by the A key is:

Keycodes	Keys	Comments
23	LBL	Execution begins here when A is pressed.
11	A	
35	g	These keys produce the same result here as they do from the keyboard.
04	1/x	
24	RTN	Defines the end of the program.

Now continue pressing SST to see how the default programs for the B and C keys are written. The keycodes and keys are shown below:

Keycodes	Keys	Comments
23	LBL	Execution begins here when B is pressed.
12	B	
31	f	Once again, the keys here produce the same result as they do from the keyboard.
09	√x	
24	RTN	Defines the end of the program.
23	LBL	Execution begins here when C is pressed.
13	C	
35	g	Calculates y^x as you would from the keyboard.
05	y^x	
24	RTN	Defines the end of the program.

Merged Keycodes. To conserve memory, the most frequently used prefix-suffix pairs are merged into single codes (*internal restrictions prohibit merging all such pairs*). This is illustrated in the default program executed by the D key. If you are not already at the D key, single-step through memory until you reach it. The program looks like this:

Keycodes	Keys	Comments
23	LBL	Execution begins here when D is pressed.
14	D	
35 08	g R↓	The same as from the keyboard.
24	RTN	Defines the end of the program.

You can see how the keys g and R↓ were combined and represented by the keycode 35 08. Continue to press SST to view the

E program which also contains a merged code. The keys and keycodes are listed below.

Keycodes	Keys	Comments
23	LBL	Execution begins here when E is pressed.
15	E	
35 07	g x⇄y	Again the keys you would press from the keyboard go here.
24	RTN	Defines the end of the program.

The keys that are merged are listed below:

Keycodes	Keys	Keycodes	Keys
35 00	g LST X	35 09	g R↑
35 07	g x⇄y	35 08	g R↓
33 01	STO 1	34 01	RCL 1
33 02	STO 2	34 02	RCL 2
33 03	STO 3	34 03	RCL 3
33 04	STO 4	34 04	RCL 4
33 05	STO 5	34 05	RCL 5
33 06	STO 6	34 06	RCL 6
33 07	STO 7	34 07	RCL 7
33 08	STO 8	34 08	RCL 8
35 01	g NOP		

Note particularly that when a g NOP (no operation) is encountered by the pointer, no operation occurs.

Also notice that STO 9 and RCL 9 are not merged. This serves as a helpful reminder that the HP-65 uses R_9 to store intermediate results when using trigonometric functions, rectangular/polar conversions, or numerical comparison tests.

Bottom Memory Display. If you pressed SST repeatedly, you would eventually reach step 100 and two dashes would appear in the display:

This is only to let you know that you are at the bottom of memory. If you press SST one more time, the program pointer comes again to the top of memory.

Full Memory Display. If the 100th step of program memory contains anything other than g NOP, the display in W/PRGM mode always appears with a dash on the right to let you know that program memory is full. For example, if the program pointer was pointed at a RTN somewhere in the middle of a program and the program memory was full, the display would look like this:

Clearing Memory

The key sequence f PRGM cannot be stored in program memory. It is used to clear program memory. Whenever you intend to redefine one or more of the program control keys A thru E, you must clear program memory first. Otherwise, as you key in your program, the default programs are pushed down in memory and unless your program is 100 steps, you may end up with two programs controlled by one program control key.

To clear program memory, switch to W/PRGM mode and press:

f PRGM

This fills the entire 100-step memory with g NOP codes and sets the program pointer to the top of memory.

Writing Your Own Program

Now that you know a little more about the program memory of your calculator, let's write another program.

This program will calculate the volume of a sphere using the simple formula: Volume $= r^3 \times \pi \times 4/3$. All you will have to do is key in the radius (r) and press A. To key in the program follow the procedure below:

1. Set the program mode switch to W/PRGM.
2. Press f PRGM to clear program memory and set the program pointer to the top of memory marker.
3. Press the keys in the order shown. Take the time to identify each key by its keycode.

Keycodes	Keys	Comments
23	LBL	Program execution begins here when A is pressed.
11	A	
03	3	Calculates r^3.
35	g	
05	y^x	
35	g	Calculates $r^3 \times \pi$.
02	π	
71	×	
04	4	Calculates $r^3 \times \pi \times 4/3$.
71	×	
03	3	
81	÷	
24	RTN	Defines the end of the program.

If you make a mistake, clear the program by pressing f PRGM and start over. You'll learn how to correct mistakes and edit your programs shortly.

Running the Program

To run the program, set the W/PRGM-RUN switch to RUN. Now find the volume of a sphere with a radius of 10.

Press	See Displayed	
10 A	4188.79	Volume of the sphere.

When you pressed A, the program pointer searched through program memory from its current position until it found LBL A. Program execution then started from this point. If there had been no label A, the calculator would have begun execution at the top of memory. If you've just run the program in the above example, switch to W/PRGM mode. The display shows the code of the last instruction executed:

24

The RTN at the end of the program stops calculator execution, halts the stepping of the program pointer, and returns control to the keyboard.

If you now need to calculate the total volume of five spheres of radius 10, you can simply multiply your answer by 5. The program is not affected by any calculations you perform. Switch back to RUN mode and try it.

Press	See Displayed	
10 A	4188.79	Volume of one sphere.
5 ×	20943.95	Volume of five spheres.

Now switch again to W/PRGM mode. The display shows:

71

This is the keycode for multiply. Although the program pointer stays at RTN, the display shows the last key pressed during a calculation.

Magnetic Cards

Now record your program on a magnetic card as you did in the introduction by:

1. Selecting a blank, unprotected *(unclipped)* magnetic card.
2. Switching to W/PRGM mode.
3. Passing the card through the right lower slot exactly as you did when entering a prerecorded program.

The position of the W/PRGM-RUN switch is very important when recording programs or using prerecorded cards. There is an easy way to remember which position the switch should be in for each use.

To Record Your Own Program. The switch belongs in the W/PRGM mode position. Think of it this way: In W/PRGM (write program) mode I *write* my programs onto the magnetic card.

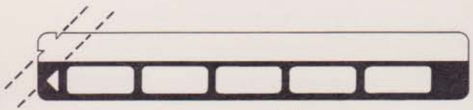
Prerecorded Programs. The switch belongs in the RUN mode position. Remember it by saying to yourself: When I want to *run* a program from a prerecorded card I put the switch in RUN mode to read the card in.

Read/Write Operations

Reading or writing a card records all 100 steps of the program memory. However, it does not change the contents of the registers, which enables you to utilize data developed by a prior program. If a read operation fails, program memory is cleared to g NOP codes and the display blinks. Reading a blank card will have the same effect.

Protecting a Card

To protect a card containing a stored program, clip through the notches with scissors as shown below.



A further precaution is to record the program on the opposite edge of the card as well. If by accident you erase your program you can always insert the other end (*opposite to the arrowhead*) of the card. However, for permanent program storage we recommend that you use only one track since:

1. The second program cannot easily be labelled.
2. Extreme care must be taken to protect the second program. (*Do not clip more than you would on the first track or you may lose information.*)
3. The motor roller is over the second track. Over a period of time, the second track may not read properly.

Marking a Card

You can write on the non-magnetic side of your card using any writing implement that does not emboss the card. It is customary to write a program name on the top of the card and to write symbols identifying the functions of the top row keys in the spaces below. Annotating magnetic cards with a typewriter may impair the read/write properties of the cards. To permanently mark a card, clean it first of grease, oil, etc. Then use a pen with india ink.

Editing the Program

You can easily edit (*correct or change*) your HP-65 programs by using the editing features built into the calculator. These features allow you to insert or delete a step anywhere in the program.

Positioning the Pointer

Before you can edit a program, you must first position the program pointer at the particular step to be edited. You have already learned one way to do this. By pressing SST you advance the program pointer one step at a time. However, if the step to be corrected is far down in memory, this method may not be convenient. There is an easier way.

You can move the pointer to any LBL (label) in the program by switching to RUN mode and pressing GTO (go to) [A thru E].* The program pointer searches through memory from its current position, finds the LBL, and stops. For example, if you press GTO C, the pointer searches for LBL C. Then, if the pointer finds the label, it stops at the step containing the C key. If the label is not found, the pointer goes to the top of memory and stops. With the pointer positioned at the C key, you can then switch to W/PRGM mode and use the SST key to move the pointer to the correct step, having bypassed long sections of the program.

To return the program pointer to the top of memory, you have two choices:

1. Press SST until you complete the cycle through memory and once again reach the top of memory marker.
2. Switch to RUN mode and press RTN.

Naturally, your position in the memory will determine which method you use. In most cases, pressing RTN in RUN mode is more convenient. (*Note that* RTN *operates differently in RUN mode than in W/PRGM mode.*)

Insert Operation

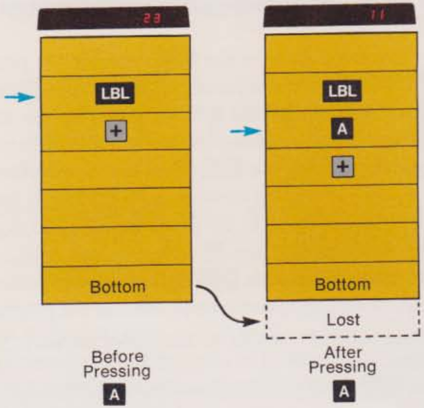
Whether you know it or not, you have already learned how to insert steps in your program. Effectively, when you were writing a program and you pressed a key in W/PRGM mode, it was inserted between the displayed step and the following step. The program pointer then moved to display the inserted step.

To summarize the procedure for inserting program steps:

1. Position the program pointer so that the code of the instruction that is to precede the insertion is displayed.
2. Press the key or keys to be inserted. The rest of the program is pushed down to make room.

* You can also move the pointer to labels 1 thru 9 by pressing GTO [1 thru 9]. Labels are discussed on page 71.

As you can see in the drawing below, when a step is inserted, the bottom step of memory is lost.



Do not concern yourself with the bottom step of memory when inserting unless the display indicates that memory is full.

Insert operations are not performed for the second key of a merged code since the second keystroke uses the same memory location as the first.

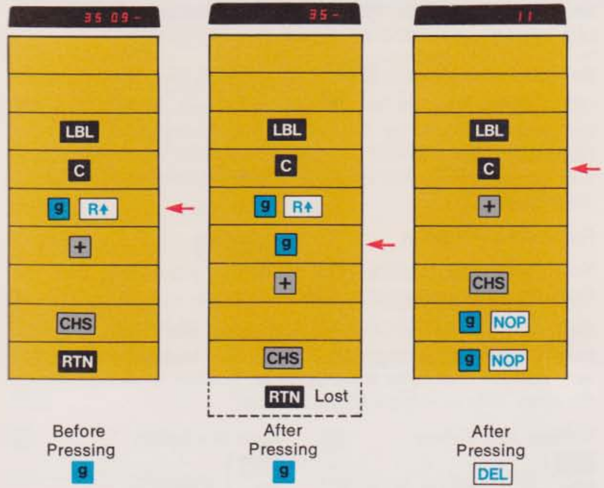
When the program pointer is at the bottom of memory, insert operations are not performed. If the pointer is at the bottom, and you try to insert a step, the code(s) will be generated in the display, but will not go into memory.

Delete Operation

Deleting steps in a program is easily accomplished by following the procedure below:

1. Position the program pointer to display the code of the instruction to be deleted.
2. Press g DEL (delete) in W/PRGM mode. The instruction is removed and the program pointer moves up to the previous step in memory.

The drawings below will help you to understand what happens in program memory when you delete a step.



As seen above, the g key is first interpreted as part of a program operation and it is inserted into memory, pushing down all steps below it. Since memory is full, the bottom step of memory is therefore lost, as in any insert operation. When the suffix key DEL is keyed in, the delete operation is recognized by the calculator and both the g key and the incorrect step are deleted.

The program pointer moves up two steps and g NOP's fill the two vacant steps at the bottom of memory. You will, of course, want to reinsert the last step when this happens. (*For programs shorter than 100 steps—no dashes in the display—no concern need be given the bottom of memory.*)

Deleting Consecutive Steps. To delete a sequence of program steps, position the program at the last step in the sequence. Each time g DEL is pressed, the pointer will backstep to display the next step to be deleted.

Deleting the Bottom Step. If the program pointer is at the bottom step of memory, pressing g DEL deletes two steps in memory: the 100th step *and* the 99th step. When deleting the bottom step of a program, remember to reinsert the extra lost step.

Backstepping. If, using SST, you happen to overshoot the mark only slightly, you can use g DEL to recover. Simply backstep the program pointer by deleting the intervening steps, make the required insertion or deletion, and then reenter the deleted steps. This procedure is often easier than repositioning the pointer by other means.

Revising a Program

Now that you're familiar with the editing procedures, let's put that knowledge into practice with an example.

We'll take the volume of a sphere program and change it to calculate the area of a sphere ($r^2 \times \pi \times 4$). The two programs are very similar. Otherwise it wouldn't be feasible to change one to the other. Side by side they look like this:

Volume of a Sphere		Area of a Sphere	
LBL A	Beginning of program.	LBL A	Beginning of program.
3 g y^x	Calculates r^3.	2 g y^x	Calculates r^2.*
g π ×	Times π.	g π ×	Times π.
4 ×	Times 4.	4 ×	Times 4.
3 ÷	Divided by 3.	RTN	End of program.
RTN	End of program.		

* These steps could be changed to f^{-1} $\sqrt{x}$ or ENTER↑ × to save space but it would have made this example more difficult to follow.

As you can see, there is little to change. Key in the sphere volume program now if you have not already done so by following this procedure:

1. Switch the calculator to W/PRGM mode.
2. Press f PRGM to clear program memory.
3. Key in the keystroke list on the left.
4. Switch back to RUN mode.

Use the following example to check your program before we edit it. **Example.** Find the volume of a sphere of radius 25.

Press	See Displayed
25 A	65449.85

In order to change the sphere volume program to a sphere area program, we need to make the following changes:

Volume of a Sphere		Area of a Sphere	
LBL		LBL	
A		A	
3	← Delete this step.	2	← Insert this step.
g		g	
y^x		y^x	
g		g	
π		π	
×		×	
4		4	
×		×	
3	← Delete this step.	RTN	
÷	← Delete this step.		
RTN			

Here's how we do it:

1. Switch to RUN mode.
2. Press GTO A to return the pointer to LBL A .
3. Switch back to W/PRGM mode.
4. Press SST once to position the pointer at the step being deleted. The display should show code 03.
5. Press g DEL to delete the unwanted step. You should see keycode 11 displayed.
6. Press 2 to insert the new step.
7. Press SST nine times to position the pointer at the second of the two consecutive steps to be deleted. The display should show keycode 81.
8. Press g DEL to delete the ÷ key. The pointer backs up to display 03.
9. Press g DEL to delete the 3 key. The display should show keycode 71.
10. Now switch back to RUN mode to run the program.

Run the program by keying in a value for r and pressing A.

Example. Calculate the area of a sphere with r = 25.

Press	See Displayed
25 A	7853.98

For additional practice, try changing this program back again so that it calculates the volume of a sphere.

Branching

Although program execution is normally sequential, with one step executed after another, the calculator has the ability to jump (branch) to any labelled section of a program and continue execution there.

Labels

A label consists of the LBL key and a digit key (0 thru 9) or a program control key (A thru E). Any or all of these 15 unique labels can be used in a program, although only program control key labels (LBL A thru LBL E) can mark a section of program that can subsequently be executed directly from the keyboard.

Direct Branching

A direct branch in a program consists of the GTO key and a digit key (0 thru 9) or a program control key (A thru E). Each such direct branch should be paired with a corresponding label somewhere within the program. If there is no corresponding label, the calculator will continue execution at the top of memory. When the calculator executes a direct branch, the program pointer searches downward in memory for the label from the GTO, not from the top of memory. Program execution continues at the corresponding label. For example, GTO 3 branches the program pointer to LBL 3 and program execution continues there. Remember that GTO 3 produces the same result from the keyboard, except that program execution does not continue.

Writing a Program with a Direct Branch. Direct branching is commonly used when two or several functions have a common section. Let's write a program to illustrate this. Suppose you needed to write programs for two similar equations:

$$y = \frac{\sin x}{3(\sin x)^2 + 2} \qquad y = \frac{\cos x}{3(\cos x)^2 + 2}$$

You could, easily enough, write a separate program for each and control one with the A key and the other with the B key.

Keys	Comments
LBL A	Execution begins here.
f SIN	Calculates sin x.
ENTER↑	Saves copy for numerator.
ENTER↑ ×	$(\sin x)^2$
3 ×	$3(\sin x)^2$
2 +	$3(\sin x)^2+2$
÷	The answer.
RTN	

Keys	Comments
LBL B	Execution begins here.
f COS	Calculates cos x.
ENTER↑	Saves copy for numerator.
ENTER↑ ×	$(\cos x)^2$
3 ×	$3(\cos x)^2$
2 +	$3(\cos x)^2+2$
÷	The answer.
RTN	

As you can see, the last nine steps of each program are the same. What we can do is write a program containing these nine steps and branch our LBL A and LBL B programs to it. This third program will be controlled by the C key and will calculate $\frac{a}{3a^2+2}$. For one program, "a" will equal sin x, while for the other, "a" will equal cos x. Switch to W/PRGM mode, press f PRGM, and key in this program now.

Keys	Comments
LBL C	Beginning.
ENTER↑	Save copy of a.
ENTER↑ ×	a^2
3 ×	$3a^2$
2 +	$3a^2+2$
÷	The answer.
RTN	End.

Now key in the A and B programs which have been shortened to this:

Keys	Comments
LBL A	Beginning.
f SIN	Calculates sin x.
GTO C	Then branches to label C.

Keys	Comments
LBL B	Beginning.
f COS	Calculates cos x.
GTO C	Then branches to label C.

Notice immediately that these last two programs did not end in RTN because they branch directly to LBL C and continue execution there. Also notice that in entering these three programs you keyed in the C program first, then the A program, and finally the B program. The order of entry or use is immaterial. Now switch to RUN mode and let's run the programs.

Example. Calculate $\frac{\sin x}{3\,(\sin x)^2+2}$ and $\frac{\cos x}{3\,(\cos x)^2+2}$ for $x = 60°$.

Press	See Displayed	
g DEG	0.00	Set degrees mode if not already set.
60 A	0.20	The answer.
60 B	0.18	The answer.

As a further improvement to the program *(if you are interested in conserving steps in memory)*, rearrange the labels as shown below:

Keys	Comments
LBL	This program calculates sin x and then branches to label 1.
A	
f	
SIN	
GTO	
1	
LBL	This program calculates cos x and then continues execution through label 1.
B	
f	
COS	

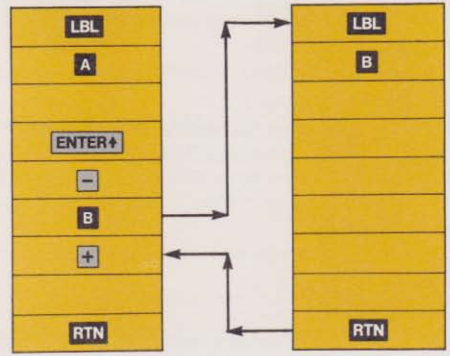
Keys	Comments
LBL	Beginning of common section.
1	
ENTER↑	$\frac{a^2}{3a^2+2}$
ENTER↑	
×	
3	
×	
2	
+	
÷	
RTN	End of both programs.

First of all, notice that LBL C has been replaced by LBL 1. Since we are not planning on executing that portion of the program from the keyboard, it is not necessary to use a valuable program control key. Secondly, notice that we've eliminated two steps in the program by positioning LBL B directly before LBL 1 *(previously* LBL C). In this way, program execution doesn't have to transfer from LBL B to LBL 1 using GTO 1, it can continue sequentially.

Subroutine Branching

A second method of transferring program execution is by means of subordinate programs or "subroutines." When a series of steps is repeated in a program or is common to a number of programs, a single subroutine containing the steps may be written.

Just as you use the A thru E keys to control the steps between the corresponding LBL and RTN, so can the calculator use these keys. When the RTN is reached, instead of stopping the program, program execution automatically branches back (*returns*) to the step following the original branch instruction.



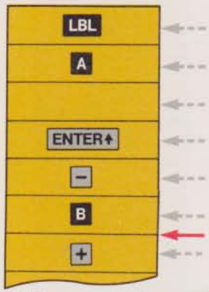
Program A　　Subroutine B

As you can see, a subroutine is a program. The only difference is the usage. In the above illustration, if you press B, the program controlled by B is executed and the calculator stops at the RTN. However, if you press A, the calculator executes the program controlled by A sequentially until it reaches the B program step. Then program execution transfers to LBL B. When the calculator reaches that same RTN this time, it now branches back to the A program and continues execution sequentially, starting with the step that follows the B key.

In other words, in the A program the B key is just one more key in the program. The program executes just as if the keys were pressed from the keyboard. Note that only LBL 's A thru E can designate subroutines, not LBL 's 0 thru 9 .

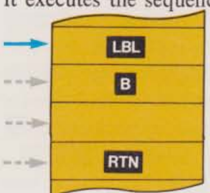
A good thing to remember when using subroutines is that your subroutine often repositions or changes data in the stack. Be sure to allow for this by first storing away values needed later.

Secondary Pointer. How does the calculator keep track of where to return from a subroutine call? It uses a second program pointer. In the previous drawing, when A is pressed, execution proceeds sequentially until the main program pointer reaches B . Here the pointer stops and marks its place directly after the B key.* And there the main program pointer waits.



Main pointer deactivates and marks its place after executing subroutine call B .

Meanwhile, a secondary pointer is activated, and it searches for LBL B , starting at the top of memory. When it finds the label, it executes the sequence of keys in the subroutine.

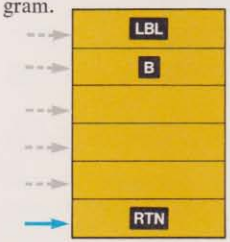


Secondary pointer executes subroutine B.

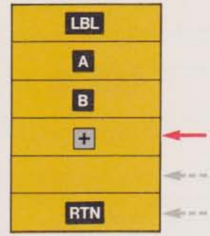
Subroutine B

* In marking its place, the main program pointer is inserted into memory, though it does not take up one of your 100 steps. If you stop the secondary pointer in the middle of a subroutine and manually reposition it, you can see the main pointer in the display. It appears as keycode 41 and would execute as ENTER↑ in your program.

The execution of the RTN at the end of the subroutine deactivates the secondary pointer and reactivates the main program pointer. Program execution then continues sequentially in the main program.



Secondary pointer deactivated after executing RTN .



Main pointer reactivated. Execution continues at + .

Writing a Program with a Subroutine. In order to calculate the area and volume of a sphere efficiently we would use a subroutine. The equations for these two problems are:

$$\text{Area} = r^2 \times \pi \times 4 \quad \text{and} \quad \text{Volume} = \frac{r^3 \times \pi \times 4}{3}$$

The volume equation can easily be expressed in terms of the area equation:

$$\text{Volume} = \frac{r \times \text{Area}}{3}$$

And that is the way we'll write our programs. The program controlled by A will calculate the area of the sphere. Switch to W/PRGM mode, press f PRGM to clear the default programs, and key in the following list of keys.

Keys	Comments
LBL A	Beginning of program.
STO 1	Store r for later use.
f⁻¹ √x	Calculate r^2.
g π	Recall π.
×	
4	
×	The area.
RTN	End of program.

Now switch to RUN mode and try this program to make sure it works.

Example. Find the area of a sphere with r = 15.

Press	See Displayed	
15 A	2827.43	Area of sphere.

Now let's find the volume of the same sphere using this program.

Press	See Displayed	
15 A	2827.43	Area of sphere.
RCL 1	15.	Recall the radius value r.
×	42411.50	r × Area.
3	3.	
÷	14137.17	Volume of sphere.

In order to make this key sequence a separate program we need only add LBL to the top and RTN to the bottom.

Keys	Comments	Keys	Comments
LBL	Beginning of program.	×	r × Area.
B		3	
A	Call subroutine A.	÷	Volume of sphere.
RCL 1	Recall the radius value r.	RTN	End of program.

Notice that instead of having to key in the radius again, we can simply recall it from R_1. Switch to W/PRGM mode and key in this new program. Don't press f PRGM this time because we want to keep the A program in the calculator.

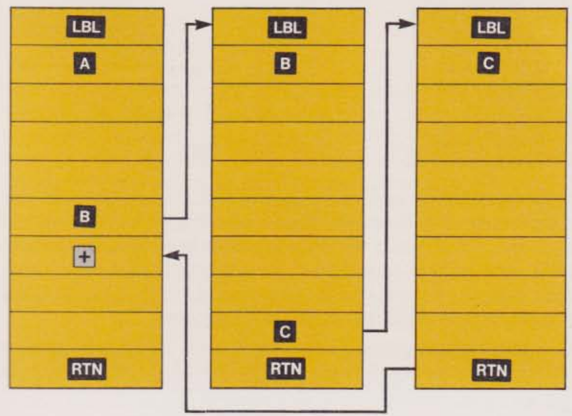
Now let's use both programs.

Example. Find the area and volume of a sphere with a radius of 20.

Press	See Displayed	
20 A	5026.55	Area of the sphere.
B	33510.32	Volume of the sphere.

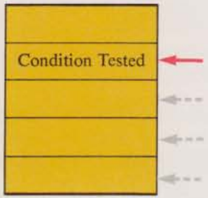
The calculator finds the volume of the sphere in this example in the same way you did in the previous example.

Second Subroutines. A subroutine cannot call a subroutine of its own. There is simply no third pointer to keep track of things. If you try to call a second subroutine, you'll find that program execution transfers from that subroutine back to the main program, not the first subroutine.

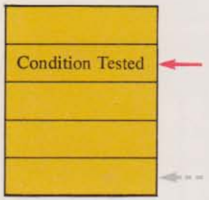


Conditional Testing

Nine different program instructions give your HP-65 the ability to make decisions within a program. These "conditionals" modify program execution depending on conditions in the program. They all work similarly.



Condition Met
(*continue*)



Condition Not Met
(*skip two steps*)

Continue here.

If the condition *is* met, the program will execute the next two steps, which often contain a branch instruction. If the condition *is not* met, the program will skip over these two steps. Sometimes, you'll even be able to condense the operations that would normally require a branch into the two steps. There will be examples of both these possibilities in the text to follow.

Numerical Comparisons

Four tests compare the contents of the X- and Y-registers. These are:

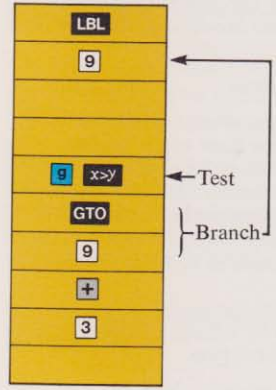
g x≠y Are the values in X and Y unequal?

g x≤y Is the value in X less than or equal to the value in Y?

g x=y Are the values in X and Y equal?

g x>y Is the value in X greater than the value in Y?

In each case, if the answer to the question is **YES**, then program execution **continues** sequentially. If the answer to the question is **NO**, then the program pointer **skips two** steps before continuing.



In this program segment, execution of g x>y compares the current values in X and Y.

1. If the value in X **is** greater than the value in Y, GTO 9 is executed and the preceding section is repeated.
2. If the value in X **is not** greater than value in Y, GTO 9 is skipped and + is executed.

Each time a comparison test is made in a program, a copy of the value in X is stored in R_9. The value in the Last X register does not change. R_9 should therefore be used with caution for storage purposes when these tests are a part of your program. Now let's write some programs using these numerical comparisons.

Two Programs Using Numerical Tests. This first program is derived from the following anecdote:

According to unreliable sources, many years ago there was a prosperous kingdom where a tired and grumpy king ruled. One day, looking for new amusement, the king sent out the following message throughout his kingdom: "Whosoever finds a game of suitable amusement for me, shall be granted any wish he desireth."

Lo and behold, a young gentleman presented the king with the game of chess. The king was ecstatic! "What is your wish?" asked the delighted king. Replied the gentleman, "O wise and noble king, all I ask is that you put down one stalk of wheat for the first square on the chessboard, merely double this amount for the second square, then double the new amount for the third square, and so on for the remaining squares. All I wish to be given is the amount of wheat put down for the final chess square."

To this the king replied, "But, generous gentleman, this is a prosperous kingdom; surely I can do more for you than that!" But the gentleman was equally insistent.

P.S. The kingdom produced about one billion (1,000,000,000) stalks of wheat (W) annually and the chessboard had 64 squares (S) to be filled.

Was the gentleman really being generous?

The program calculates the amount of wheat to be placed on each square in succession in R_2 and keeps track of the number of the square in R_1. If more than one billion stalks of wheat have to be supplied for a given square, the program halts and displays the number of the square at which it surpasses one billion. If the 64 chess squares can be filled without depleting the kingdom's supply, the program halts and displays the number of stalks of wheat that need to be paid.

Switch to W/PRGM mode, press [f] [PRGM] to clear program memory, and key in the following list of keys.

Keys	Comments
LBL	Beginning of program.
A	
1	Initalize R_1 and R_2 to take care of 1st square.
STO 1	
STO 2	
LBL	Beginning of repeat.
1	
2	Calculate amount.
STO	
×	
2	
RCL 1	Increment square number.
1	
+	
STO 1	

Keys	Comments
6	
4	Compare square number to 64.
g x=y	
RCL 2	If square number equals 64, display amount and stop.
RTN	
RCL 2	Otherwise, compare amount to 1 billion (1×10^9).
EEX	
9	
g x>y	
GTO	Branch if one billion is greater than amount.
1	
RCL 1	Otherwise, display square number and stop.
RTN	

Now switch back to RUN mode and run your program. After several seconds the calculator should display:

31.00

Over one billion stalks of wheat have to placed on the 31st square!

(*To find the exact amount on that square press* [RCL] [2].) To calculate the amount on the 64th square, press [2] [ENTER↑] [6] [4] [g] [y^x]. Needless to say, the generous gentleman was executed!

The second program calculates the arc sine of an input value *x* (*x must be within the limits of −1 and +1.*) The program tests the resulting angle, and if it is negative or zero, adds 360 degrees to it to make the angle positive.

Switch back to W/PRGM mode, press [f] [PRGM], and key in the program now.

Keys	Comments
LBL	Beginning of program.
D	
f^{-1}	Calculates the arc sine.
SIN	
0	Puts 0 in X. Exchanges 0 and arc sine.
g x≤y	

Keys	Comments
g x≤y	Test the angle.
3	These two steps are skipped if angle is positive.
6	
0	Adds 0 or 360 to assure than the angle is positive.
+	
RTN	

If the angle is positive, [3] and [6] are skipped and zero is added to the angle. Otherwise, 360 is added to the angle. Let's try a problem.

Example. Calculate the arc sine of .5 and —.5.

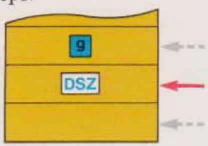
Press	See Displayed	
.5	.5	
B	30.00	(degrees)
.5 CHS	−.5	
B	330.00	(degrees)

Decrement and Skip on Zero

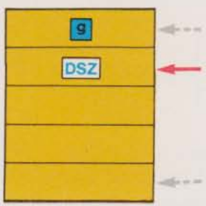
The DSZ (*decrement and skip on zero*) key subtracts 1 from the contents of R_8 and then tests for a non-zero value. The conditional can be stated like this:

Is the value in R_8 a number other than zero?

Once again, if the condition is met, program execution continues sequentially. If the condition is not met and the value in R_8 is zero after 1 has been subtracted, the program pointer skips two steps.



Condition Met.
If the value in $R_8 \neq 0$ continue execution sequentially.



Condition Not Met.
If the value in $R_8 = 0$ skip two steps before continuing execution.

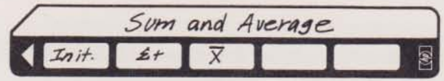
Continue here.

Naturally, since R_8 is used by DSZ, you will not want to use this register for other storage purposes when this test is a part of your program. DSZ does not work if the number stored in R_8 falls outside the range $-10^{10} = r_8 = 10^{10}$ and (*in general*) is not designed to work for non-integer values less than one.

DSZ can be used in many ways in your programs. It can be used as a counter, as a flag (*see page 91*) to repeat segments of your program, or to repeat your whole program.

Writing a Program Using DSZ. To use DSZ as a counter in your program, store zero in R_8 and include DSZ in the section of your program that repeats. As your program runs, R_8 keeps track of the number of repetitions (*although the number is negative*).

The following programs sum and average a group of numbers using DSZ in this way. The key art will give you a good idea as to how these programs work.



Switch to W/PRGM mode, press f PRGM, and key in these programs now.

Keys	Comments
LBL A	Beginning of initialization program.
f REG	Clears all registers.
RTN	
LBL B	Beginning of program that sums the data.
STO + 1	Each number is accumulated in R_1.

Keys	Comments
g DSZ	And the value in R_8 decreases by 1 each time.
RCL 1	Display running total.
RTN	
LBL C	Beginning of program that averages.
RCL 1 RCL 8 CHS ÷	Total is divided by the positive value of the number of repetitions.
RTN	

The first program, controlled by **A**, simply clears the registers. The second program, controlled by **B**, accumulates each number in R_1 and displays a running total. DSZ is used to count the number of repetitions in the group. Each time DSZ is encountered, 1 is subtracted from the value in R_8. Since that value starts at zero, the condition will always be met and the two steps will never be skipped. The third program, controlled by **C**, takes the average of the numbers by dividing the total by the number of repetitions. Since the value in R_8 is negative, its sign is changed before computing the average.

Now switch back to RUN mode and try the following example.

Example. Find the total and average of the following group of numbers:

65 78 908 345 23 98

Press	See Displayed	
A	0.00	Initialize program.
65 B	65.00	Running total.
78 B	143.00	Running total.
908 B	1051.00	Running total.
345 B	1396.00	Running total.
23 B	1419.00	Running total.
98 B	1517.00	Final total.
C	252.83	Average.

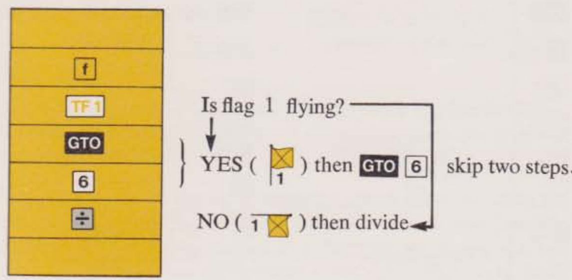
Flags

Your HP-65 also contains two flags which act as invisible switches. You can set each flag ON or OFF. You can also test each flag to see if it is ON (*flying*), or test it to see if it is OFF (*lowered*). The keystrokes to set and test these flags are listed below.

Keys	Result
f SF 1	Set flag one flying.
f^{-1} SF 1	Lower flag one.
f TF 1	Test flag one. Is it flying?
f^{-1} TF 1	Test flag one. Is it lowered?
f SF 2	Set flag two flying.
f^{-1} SF 2	Lower flag two.
f TF 2	Test flag two. Is it flying?
f^{-1} TF 2	Test flag two. Is it lowered?

These two pairs of instructions can be executed from the keyboard or from a program.

For each test made, if the answer is *YES*, program execution continues sequentially. And if the answer if *NO*, the program pointer skips two steps of memory before continuing.



Note that a flag retains its setting until an instruction to change it is executed.

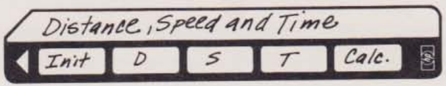
Writing a Program Using Flags. The relationships between speed, time, and distance for a moving body are given by the following formulas:

$d = s \cdot t$ Distance = speed × time

$s = \frac{d}{t}$ Speed = $\frac{\text{distance}}{\text{time}}$

$t = \frac{d}{s}$ Time = $\frac{\text{distance}}{\text{speed}}$

We'll write a program to calculate any one of the above when the other two values are given. The key art might look like this:



Now switch to W/PRGM mode, press f PRGM and key in the following list of keys.

Keys	Comments
LBL A	Beginning of initialization routine.
f SF 1	Set flag 1 flying.
f STK	Clear the stack.
DSP • 4	Change display setting.
RTN	
LBL B	Beginning of distance routine.
f TF 1	Test flag 1. Is it flying?

Keys	Comments
RCL 2 RCL 3 ×	Otherwise calculate distance.
RTN	
LBL C	Beginning of speed routine.
f TF 1	Test flag 1. Is it flying?
STO 2 RTN	If it is, store speed and stop.
RCL 1 RCL 3 ÷	Otherwise calculate speed.
RTN	

Keys	Comments
LBL D	Beginning of time routine.
f TF 1	Test flag 1. Is it flying?
GTO 1	If it is, branch to label 1.
RCL 1 RCL 2 ÷	Otherwise calculate time.
f →D.MS	Then convert it to hours, minutes, seconds.

Keys	Comments
RTN	
LBL 1	
f^{-1} →D.MS	Convert input time to decimal hours.
STO 3 RTN	Store in R_3, and stop.
LBL E	Beginning of calculation routine.
f^{-1} SF 1	Lower flag 1.
RTN	

The time program is set up so that the time is input in hours, minutes, and seconds, although for calculating purposes it will be converted to decimal hours.

To calculate one of the three variables, press A to initialize the routines by setting flag 1. Then input a variable and press its corresponding program control key. Because the flag is set, this variable is stored away and a value is not calculated. Next, input the second variable and press its corresponding program control key. Again, the second variable is stored away and no calculation is performed. The unknown value is calculated by pressing E (*which clears flag one*) and then pressing the corresponding program control key. Because the flag is not set, the unknown value is not stored but calculated. After each calculation press A to initialize the routines again. Try the following examples to see how this works.

Example. Calculate s when t = 5 hours and 30 minutes and d = 500.

Press	See Displayed	
A	0.0000	Runs initialization routine.
5.30 D	5.5000	Time is converted to decimal hours.
500 B	500.0000	Key in the distance.
E C	90.9091	Units per hour.

Example. Calculate t when s = 700 and d = 5000.

Press	See Displayed	
A	0.0000	Runs initialization routine.
700 C	700.0000	Key in the speed in units per hour.
5000 B	5000.0000	Key in the distance
E D	7.0834	7 hours, 8 minutes, and 34 seconds.

Example. Calculate d when s = 60 and t = 74 hours, 42 minutes, and 50 seconds.

Press	See Displayed	
A	0.0000	Runs initialization routine.
60 C	70.0000	Key in the speed in units per hour.
74.4250 D	74.7139	Time is converted to decimal hours.
E B	5229.9722	The answer.

Although flags require valuable memory for setting and unsetting them, they are still handy for program decision making that isn't the result of a direct comparison of the X- and Y-registers.

DSZ as a Flag

By setting the contents of R_s equal to 1, you create your own self-clearing flag using DSZ. When the program executes DSZ, it decrements the contents of R_s, which sets it to zero. Then it tests R_s and, because it is zero, skips two steps before continuing execution (*just as when testing a flag that is set*). The second time the program executes DSZ, the program pointer continues sequentially (*just as when testing a flag that is clear*) because the number in R_s is no longer zero.

Interrupting Your Program

R/S (run /stop) is a special program control key that operates differently from the keyboard than as a program step. As a program step R/S interrupts program execution at an intermediate point, allowing you to key in data, make additional calculations, etc. From the keyboard, R/S will start a program at the position of the active pointer or halt a running program. R/S, however, is also used to control programs differently from what you have learned thus far, so the following information should be studied with care.

To Enter Data

The primary use for R/S in a program (*or subroutine*) is to stop the program in order to allow you to key in data. When a R/S program step is encountered in a running program, the program halts, leaving the pointer at the R/S. By pressing R/S from the keyboard, program execution will continue.

Writing a Program Using R/S to Enter Data. To show you how this works, let's write a program to calculate the cumulative cost of various quantities of differently priced items at a 15% discount.

Switch to W/PRGM mode and press f PRGM . Now key in the following list of keys:

Keys	Comments
LBL	Beginning of program.
A	
f	Initialize routine.
STK	
LBL	Identify place to start repetition.
3	
R/S	Stop to key in quantity.
ENTER↑	Copy quantity to Y.
R/S	Stop to key in price.
×	Quantity × price.
•	Calculate discounted price.
8	
5	
×	
+	Add to previous total.
GTO	Repeat, starting at label 3.
3	

Notice in particular that there is no RTN needed at the end of this program. This is because the program is a never-ending loop. And it already stops each time through the loop to let you key in new data.

When the program stops the first time, you key in the quantity of the item and press R/S to start the program running again. R/S always starts a halted program at the current position of the activated pointer. When the program stops again, you key in the price of the item and again press R/S. It calculates the total and returns to label 3 where it stops to receive the next quantity. A running total is displayed. Switch back to RUN mode and try it now.

Example. Assume that you get a 15% discount on the following purchases:

Quantity	Price of Each
5	\$2.00
7	\$4.00
8	\$5.00
22	\$6.00

Calculate the cumulative cost.

Press	See Displayed	
A	0.00	The stack is cleared.
5 R/S	5.00	The first quantity.
2 R/S	8.50	Running total.
7 R/S	7.00	
4 R/S	32.30	Running total.
8 R/S	8.00	
5 R/S	66.30	Running total.
22 R/S	22.00	
6 R/S	178.50	Cumulative cost.

If a R/S in a program is immediately preceded by a numerical entry from the program, that number will be overwritten by an entry from the keyboard. This feature allows a program to display prompting information that will not be lifted in the stack. Except for this case, R/S does not affect the stack lift.

> **Note:** Digits occurring as program steps immediately following a R/S should be separated from the R/S by an ENTER↑ *.

Controlling Your Program with R/S. Up to this point, each program you have written has begun with a label and ended with a return. We have taught you to program this way because it was judged to be the most convenient for most people and the most used in practice. However, the great versatility of the HP-65 does not confine you to one method. R/S can be used to advantage to run initialization routines and even whole programs without using labels and saving valuable memory steps in the process.

The rule for using R/S is simple: Pair a R/S with a R/S. In other words, if you plan to initate execution of your program with R/S, a R/S must be used as a program step to halt the

* The reasons for this are discussed on p. 100 under SST Execution.

program. Similarly, if your program begins with a label (A *thru* E), and ends with a RTN you must start it with the program control key identified by the label.

Programs controlled by R/S should be at the top of memory so that they can be easily accessed by pressing RTN R/S.

> **Note:** Generally R/S should not be used to start a program beginning with a label.

Writing a Program Controlled by R/S. Switch to W/PRGM mode and press f PRGM. Then key in the following program which calculates $\sqrt{x^2 + y^2}$.

Keys	Comments	Keys	Comments
ENTER↑	Save copy of x just keyed in.	f^{-1}	Then calculate x^2.
R/S	Stop to key in y.	$\sqrt{x}$	
f^{-1}	Then calculate y^2.	+	
$\sqrt{x}$		f	
g x⇄y		$\sqrt{x}$	
		R/S	Stop the program.

Notice, in particular, that the program does not begin with LBL or end with RTN.

To run the program switch to RUN mode and press RTN to move the program pointer to the top of memory. Then key in a value for x and press R/S. When the program stops again, key in a value for y and again press R/S. The program then stops again to display the answer. Now switch to RUN mode and try the following example.

Example. Calculate $\sqrt{7^2 + 9^2}$.

Press	See Displayed	
RTN	0.00	
7 R/S	7.00	
9 R/S	11.40	The answer.

Program Stops

Blinking Display Stops. Errors that cause a blinking zero display, if executed in a program, also stop the program. Stop the blinking by pressing any key (CLX *is recommended*). You can then identify the reason for the stop by switching momentarily to W/PRGM mode to see the code of the offending operation. You can also use g LSTX to recover the last value in the display.

Normal Stops. To confirm that a program stops normally (i.e., *via a* RTN *or* R/S), switch momentarily to W/PRGM mode and observe the displayed code. It should be 24 or 84.

Accidental Stops. Remember, pressing any key will stop a program. Be careful to avoid pressing keys during program operation.

Cued Stops. If memory permits, it is sometimes helpful to put a familiar number into the X-register before stopping for data. Thus when the program stops, the displayed number identifies the desired input. For example, if your program requires eight stops for input, it is helpful to have the numbers 1 thru 8 appear so you know which input is needed.

If a cue number is created as a program step immediately preceding the R/S, it is not lifted into the stack and the number is overwritten by the data you key in. Cue numbers generated by other means (*recalled from a register, or calculated*) will be lifted.

Overflow Stops. If, during the course of a calculation, you exceed the dynamic range of the machine, a running program will be halted. The display will show 9.999999999 99.

Underflow Stops. If, during the course of a calculation, you calculate a number that is too small in magnitude ($<10^{-99}$) to be carried in a register, the register is set to zero and the program stops, if running.

Writing Programs to Solve Your Problems

In reading this manual, we hope that you have learned from the text and example programs how to program your HP-65. But you may be asking yourself: How do I write a program to solve *my* problem? It is the purpose of this section to briefly describe one approach you might use.

In general, program writing is composed of three major steps:

1. Define the problem.
2. Decide how the problem is to be solved.
3. Write down the keystrokes that need to be repeated.

You have already learned how to do step 3. Now for steps 1 and 2.

Example of program writing:

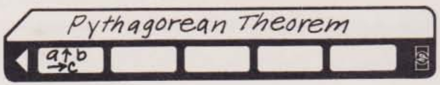
1. Define the problem: What is the program supposed to do? Write a program to solve the Pythagorean theorem:

$$c = \sqrt{a^2 + b^2}$$

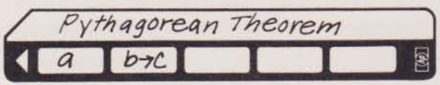
2. Decide how the problem is to be solved: What steps would you take to solve the problem on paper?

   For this you must decide what you want to solve for; what inputs will be required for that solution; what program control keys you will use and how they will be used.

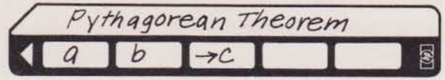
   These questions are most easily decided by drawing the key art for the magnetic cards. If you wanted to solve only for c your card might look like this:



   If you write your program this way, you will have to key in a value for a and ENTER↑ it, then key in a value for b and press A. However, you could solve the same problem with a card that looked like this:

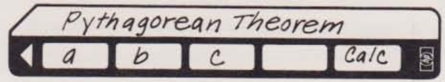


   You would key in a value for a and press A, then key in a value for b and press B. This way would allow you to store your a value so that you would not have to continually key it in for varying values of b. A third way to solve the same problem would have a card that looked like this:



   With this program, you would be able to store both your a and b values so that either could vary without having to key in the other again. You would key in a value for a and press A, key in a value for b and press B, and then press C to calculate c.

   Or you might decide that you would like to be able to solve for any variable given the other two. For this you would have a card that looks like this:



   This program would probably require setting a flag in an initialization routine by pressing RTN and R/S. Then you would key in a value for one variable and press the corresponding program control key, key in a value for the second variable and press its program control key, and finally solving for the third variable by pressing E and the corresponding program control key.

   So you can see that deciding how the problem is to be solved is a creative process. It depends heavily on your needs and the data to be processed. The way you approach your problem will largely determine how your program will be written.

3. Write down the steps for the calculator. Often on the first tries to write down keystrokes it is helpful to use the COMMENTS column of the program forms supplied with the calculator to keep track of the values in X, Y, Z, and T. Later, when you record your final documentation, you can replace those annotations with useful comments that will help you remember what various parts of your program do.

The following program was written using the third approach. Both a and b values are stored before c is calculated. Switch to W/PRGM mode, press f PRGM and key in the following list of keys to see how this program works.

Keys	Comments	Keys	Comments
LBL	Beginning of a storage routine.	RCL 1	Recall a.
A		ENTER↑	Calculate a^2.
STO 1	Store a.	×	
RTN	Stop.	RCL 2	Recall b.
LBL	Beginning of b storage routine.	ENTER↑	Calculate b^2.
B		×	
STO 2	Store b.	+	$a^2 + b^2$
RTN	Stop.	f	
LBL	Beginning of c calculation.	$\sqrt{x}$	The answer.
C		RTN	

Now switch to RUN mode and see if the program works.

Example. Calculate c for $a = 10$ and $b = 5$. For $a = 78$ and $b = 22$. For $a = 78$ and $b = 10$.

Press	See Displayed	
10 A	10.00	Key in a value.
5 B	5.00	Key in b value.
C	11.18	The answer.
78 A	78.00	Key in a value.
22 B	22.00	Key in b value.
C	81.04	The answer.
10 B	10.00	Key in new b value only.
C	78.64	The answer.

Switch again to W/PRGM mode and record the program on an unprotected magnetic card. Then mark the card as you had originally planned.

Flowcharting

One way to help you with step 2 (*deciding how the problem is to be solved*) is by means of a "flowchart." Flowcharts logically pictorialize the solution to a programming task. They are sometimes drawn long before the actual keystrokes are figured out. While flowcharting your problem, you might change or simplify your approach, see a flaw in your logic, etc. After several attempts (*even for experienced programmers*) you should have a workable flowchart and, once you do, your programming task is greatly reduced.

Any flowchart that you draw is useful, but a few basic flowcharting conventions are described briefly here. Terminal (*that is, starting or ending)* activities are represented by ovals. Arrows indicate the flow of operations between the terminals. Most calculator operations are represented by rectangles. A diamond represents a decision point. If the information within the diamond is computed as *"YES"* (*the condition is met*), the flow continues sequentially; if it is computed as *"NO"* (*the condition is not met*), the flow continues after skipping two steps.

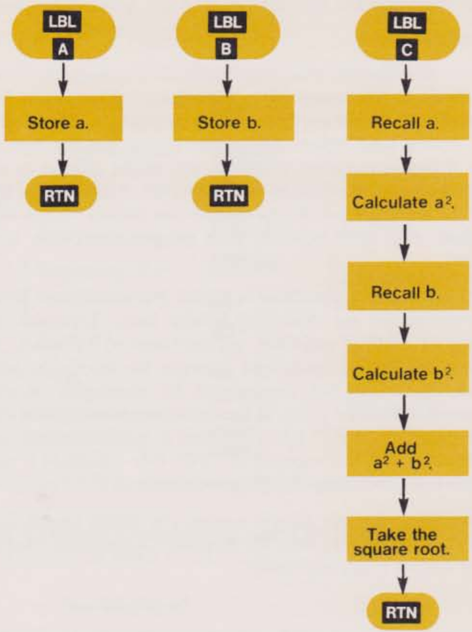
The flowchart for our simple program is shown on the following page. As you can see, once the flowchart is finalized, the program can be written relatively easily.

A complete discussion of flowcharting isn't possible here. It would take many volumes. If you want to learn more about it, you should consult a reference devoted to the subject.

Every program that you write, even the simplest, is written using our three steps, though you may find that with practice you can do much of the work in your head.

1. You must first define your problem.
2. Then you must decide how it is to be solved.
3. And finally you must write out the steps that the calculator will use to solve it.

Good luck!



Debugging Your Program

Even the most experienced programmer finds "bugs" in his programs. These bugs range from mistakes in his flowchart to mistakes in keying in the program. Wherever they occur, they need to be corrected and the HP-65 is designed to make this error-checking process as easy as possible.

SST Execution. In RUN mode, the SST key executes your program one step at a time. This allows you to observe the effect of your program in slow motion. If only a portion of your program seems to have bugs, move the program pointer to the nearest label

and use SST from there.

SST executes your program step by step. However, if the program step is a program control key (A *thru* E), pressing SST will activate the second pointer and calculate that subroutine in its entirety, finally returning to the step following the subroutine call and stopping there.

SST does not terminate data entries. Therefore, an ENTER↑ should be used to separate digits immediately following a R/S from the R/S itself. Otherwise the data entry from the keyboard will run together with the digits following the SST if followed by SST (*which may well happen in debugging a program*).

Cued Stops for Debugging. You have already read about cued stops on page 95. Where space permits, it is helpful to include additional cued stops to help you determine the position of the program pointer. This may be particularly useful to force a stop within a subroutine which otherwise would be exected in its entirety with one touch of SST. When the program is finally checked out, the unwanted stops can easily be deleted.

Common Mistakes

The most common mistakes you are likely to make with your HP-65 are listed here for your convenience.

Programming Errors

1. Having unwanted duplicate labels for program control keys because f PRGM was not pressed in W/PRGM mode before keying in a program.
2. Inadvertently erasing a program in memory by inserting a magnetic card when the W/PRGM-RUN switch was set to RUN.
3. Inadvertently erasing a program on a magnetic card by inserting an unprotected magnetic card when the W/PRGM-RUN switch was set to W/PRGM.
4. Keying unwanted operations into program memory because the W/PRGM-RUN switch was set to W/PRGM when the keys were pressed.

5. Seeing a dash in the display in W/PRGM mode for a program known to be less than 100 steps because the default programs were not first cleared by pressing f PRGM.
6. Failing to take account of a merged code and provide a g NOP as a filler in a two-step skip.
7. Mistakenly trying to use labels 0 thru 9 as subroutines. Only A thru E can be used to call a subroutine.
8. Forgetting to clear flags before using them.
9. Expecting LST X, the stack, or the registers to remain unchanged during the execution of a subroutine from the keyboard or from within a program.
10. Using DSZ in a program and forgetting to initialize R_8 to the proper value.
11. Forgetting R_9 does not have a merged code or that it is used to store intermediate results for trigonometric function, polar/rectangular conversions, and the numerical comparison tests.
12. Losing program and data by inadvertently switching the calculator OFF or by unplugging the battery charger.
13. Trying to call a subroutine from a subroutine.
14. Forgetting to delete both steps of a non-merged key sequence.

Calculation Errors

1. Failing to shift up to a gold function (f *or* f^{-1}) or down to a blue function (g) because the prefix key was omitted.
2. Losing the T-register contents because the entry of a new number or the recall of a new number lifted the stack.
3. Performing a trigonometric function in the wrong angular mode.
4. Trying to do an operation involving the X- and Y-registers with the numbers reversed because you did not press g x⇄y.

Appendix A

General Information

Accessories

Please check to see that all the standard accessories listed below have been included with your HP-65. Also, inspect the calculator for damage that may have occurred during shipment. If you find any damage or if any standard accessories listed are missing, you should file a claim with the carrier and contact the nearest Hewlett-Packard Sales or Service Office.

Standard Accessories

Your HP-65 comes complete with one each of the following standard accessories:

Accessory

Battery Pack
Battery Charger (115/230 Vac)
Travel Safety Case
Soft Case
HP-65 Owner's Handbook
HP-65 Quick Reference Guide
Standard Pac including:
- Instruction Book
- Blank Pocket Instruction Cards (20)
- Prerecorded Magnetic Cards (19)
- Head Cleaning Card
- Blank Magnetic Cards (20)

Programming Worksheet Pad

Optional Accessories

Other accessories, including software application pacs, are specified on the Accessory Order Form in the Important Information Envelope. Optional accessories include:

Accessory

Reserve Power Pack
Security Cradle
Field Case
Blank Magnetic Cards with Case (40)
Blank Magnetic Cards–bulk (100)
Programming Worksheet Pad
Blank Pocket Instruction Cards (20)

The HP 82004A Reserve Power Pack consists of a charging attachment and a spare battery pack so that one battery pack can charge while the other is in use.

Additional software pacs may be announced from time to time. Individual programs are available from the Users' Library. Please refer to the Users' Library Subscription Card shipped with your calculator (*U.S. only*).

Battery Operation

A rechargeable battery pack is provided with your calculator. Be sure to charge the battery pack before portable use of your calculator. A fully charged battery pack provides approximately 3 hours of continuous operation. By turning the power OFF when the calculator is not in use, the HP-65's battery pack should easily last throughout a normal working day. You can extend battery operation time by reducing the number of digits in the display. Press [•] between calculations and CLX prior to starting a new calculation if the wait between entries is extensive.

When 2 to 5 minutes of operating time remain in the battery pack, all decimal points in the display light. Even when all the decimal points are lit, the true decimal position is known because an entire digit position is allocated to it.

> **Note:** If you use your HP-65 extensively in field work or during travel, you may want to order the HP 82004A Reserve Power Pack, consisting of a battery charging attachment and spare battery pack. This enables you to charge one pack while using the other.

Recharging and AC Line Operation

To avoid any transient voltage from the charger, the HP-65 should be turned OFF before plugging it in. It can be turned ON again after the charger is plugged into the power outlet and used during the charging cycle.

A discharged battery will be fully charged after being connected to the charger for a period of 14 hours; overnight charging is recommended.

If desired, the HP-65 can be operated continuously from the ac line. The battery pack is in no danger of becoming overcharged. If a battery is fully discharged, it must be charged for at least 5 minutes before a card can be read or written. If the decimal points light during card feed and then go out, your battery needs recharging.

> **CAUTION**
>
> Running the HP-65 from the ac line with the battery pack *removed* may result in damage to your calculator.

The procedure for using the battery charger is as follows:

1. Make sure the line-voltage select switch on the battery charger is set to the proper voltage. The two line voltage ranges are 86 to 127 volts and 172 to 254 volts.

> **CAUTION**
>
> Your HP-65 may be damaged if it is connected to the charger when the charger is not set for the correct line voltage.

2. Set the HP-65 power switch to OFF.
3. Insert the battery charger plug into the rear connector of the HP-65 and insert the power plug into a live power outlet.
4. Set the power switch to ON. If the W/PRGM-RUN switch is set to RUN, you should see a display of 0.00.
5. Set the power switch to OFF if you don't want to use the calculator while it is charging.

6. At the end of the charging period, you may continue to use your HP-65 with ac power or proceed to the next step for battery-only operation.
7. With the power switch set to OFF, disconnect the battery charger from both the power receptable and the HP-65.

CAUTION

The use of a charger other than the HP 82002A Battery Charger *(or the equivalent number for operation outside the U.S.)* may result in damage to your calculator.

Maintenance

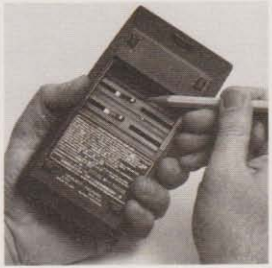
Battery Pack Replacement

To replace your battery pack use the following procedure:

1. Set the power switch to OFF and disconnect the battery charger.
2. Slide the two battery-door latches toward the bottom of the calculator.

3. Let the battery door and battery pack fall into the palm of your hand.

4. See if the battery connector springs have been inadvertently flattened inward. If so, bend them out and try the battery again.



5. Insert the new battery pack so that its contacts face the calculator and contact is made with the battery connectors.

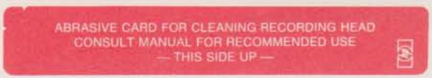
6. Insert the top of the battery door behind the retaining groove and close the door.

7. Secure the battery door by pressing it gently while sliding the two battery-door latches upward.

Magnetic Card Maintenance

Try to keep your cards as clean and free of oil, grease, and dirt as possible. Dirty cards can only degrade the performance of your card reader. Cards may be cleaned with alcohol and a soft cloth.

Minimize the exposure of your calculator to dusty, dirty environments by storing it in the soft carrying case when not in use. Each card pack contains one head cleaning card.



The magnetic recording head is similar to magnetic recording equipment. As such, any collection of dirt or other foreign matter on the head can prevent contact between the head and card, with consequent failure to read or write. The head cleaning card consists of an abrasive underlayer designed to remove such foreign matter. However, the use of the card without the presence of a foreign substance will remove a minute amount of the head itself. Thus, extensive use of the cleaning card can reduce the life of the card reader in your HP-65. If you suspect that the head is dirty, or if you have trouble reading or recording cards, by all means use the cleaning card; that's what it is for. If one to five passes of the cleaning card does not clear up the situation, refer to appendix C.

Appendix B

Additional Operating Information

Automatic Stack Lift

In order to remember when a number is lifted in the stack following a new number entry and when it is not, we would like to present a concept which, previous to this, has only been implied: number termination.

The keys on your calculator can generally be divided into two classes: the number building keys and the number terminating keys. The number building keys are:

[0 thru 9]
•
EEX
CHS

These keys are used to key in numbers.

Every other key is a number terminating key. What do we mean by number terminating? Whenever you build a number, you must somehow tell the calculator that you are through with the number—that the number is terminated. For example, if you key in the number 123, the calculator does not know if the number is terminated. If you key in the number 456, you would have the number 123456. And if you then press CHS, you would have the number −123456. However, if the first number had been terminated, it would have been lifted in the stack and you would have two numbers, 123 in the Y-register and −456 in the X-register.

This feature enables us to make a simple rule for the automatic stack lift:

If the number is terminated, the stack lifts it upon the entry of a new number.

There are only two number terminating keys which are exceptions to this rule, CLX and ENTER↑ .

CLX replaces the number in the displayed X-register with zero and prepares the X-register for a new number. The new number then writes over the zero in X.

ENTER↑ also prepares the X-register for a new number by terminating the old number and copying it into the Y-register. A new number then writes over the number in the X-register without lifting the stack.

Programming Tips

The following three programming tips should help the advanced programmer:

1. If you press A or GTO A or have GTO A in a running program and there is no corresponding LBL A, the calculator executes from the top of memory.
2. If R/S is pressed from the keyboard, the first RTN encountered will be ignored. The program will stop at the second RTN.
3. If a subroutine call does not have a corresponding label, the program will continue execution from the subroutine call, not from the top of memory. The next RTN encountered is ignored.

You can verify each of these tips with your calculator and make use of them in a number of ways.

Calculating Range

The HP-65 performs all calculations by using a 10-digit number and a power of 10. This abbreviated form of expressing numbers using powers of 10 is called scientific notation; i.e., $23712.45 = 2.371245 \times 10^4$ in scientific notation. All calculation results are rounded to 10 significant digits.

Underflow

If a result develops that is too small in magnitude to be carried in a register ($0 <$ result $< 10^{-99}$), the register is set to zero and a running program stops.

Overflow

If a computation develops a magnitude that exceeds the capacity of a register ($> 9.999999999 \times 10^{99}$), the register is set to all 9's (*with appropriate sign*), the largest magnitude expressible in a register, and a running program stops.

Temperature Range

The operating temperature range for the HP-65, including charging, is 10° to 40°C (50° to 104°F).

Calculator Service

CAUTION
Calculator can be damaged by *strong static charge.*

Blank Display

If the display blanks out, turn the HP-65 OFF, set the W/PRGM-RUN switch to RUN, and turn the HP-65 back ON. If 0.00 does not appear on the display, check the following:

1. Check the battery pack to see if it is discharged and whether it is making proper contact with the calculator.
2. If the display is still blank, try operating the HP-65 from the ac line.
3. With the battery charger connected to the HP-65, make sure the charger is plugged into a live ac outlet.
4. If the display is still blank, the HP-65 is defective. *(Refer to the warranty information that follows.)*

Low Power

All decimal points light to warn you that you have 2 to 5 minutes of operating time left on battery power. You *must* either:

1. Operate from ac power.
2. Charge the battery pack.
3. Insert a fully charged battery pack.

Improper Card Reader/Writer Operation

If your calculator appears to be operating properly except for the reading or writing of program cards, check the following:

1. Make sure that the W/PRGM-RUN switch is in the correct position for desired operation: RUN position for reading cards, W/PRGM for recording cards.
2. If the drive motor does not start when a card is inserted, make sure the battery pack is making proper contact and has ample charge. Remember that the battery charger alone does not deliver enough current to operate the drive motor.

A charger must be used in conjunction with a partially charged battery in order to drive the card reader motor. If the battery has been completely discharged, plug in the charger and wait 5 minutes before attempting to operate the card reader/writer.

3. If the card drive mechanism functions correctly, but your HP-65 will not read or write program cards, the trouble may be due to dirty record/playback heads. Use the head-cleaning card as directed. Then, test the calculator using the two diagnostic program cards furnished with it, following the instructions provided. If difficulty persists your HP-65 should be taken or sent to an authorized Hewlett-Packard customer service facility.
4. Cards must move *freely* past the record/playback heads. Holding a card back or bumping a card after the card drive mechanism engages could cause a card to be misread.

CAUTION

Cards can be accidentally erased if subjected to strong magnetic fields. *(Magnetometers at airports are in the safe range.)*

5. Check the condition of your magnetic cards. Cards that are dirty or that have deep scratches will oftentimes not read properly.
6. If you are trying to operate the calculator outside the recommended temperature range, you may experience problems with the card reader. Low temperatures slow the card reader down and often cause the drive rollers to slip.

Battery Failure

Temporary degradation, peculiar to nickel-cadmium batteries, may cause a decrease in the operating period of the battery pack. Should this happen, turn the HP-65 ON for at least 5 hours to completely discharge the battery pack. Then, put it on charge for at least 14 hours. This procedure should correct the temporary degradation.

If the battery won't hold a charge, it may be defective. If the warranty is in effect, return the pack to Hewlett-Packard according to the shipping instructions that follow. If the battery pack is out of warranty, use the Accessory Order Form provided with your HP-65 to order a replacement.

Warranty

The HP-65 is automatically warranted against defects in materials and workmanship for one (1) year from date of delivery to original purchaser. During the warranty period, Hewlett-Packard will repair or, at its option, replace components that prove to be defective, when the calculator is returned, shipped prepaid, to a Hewlett-Packard Customer Service Facility. *(Refer to Shipping Instructions.)*

This warranty does not apply if the calculator has been damaged by accident or through misuse or as a result of service or modification by any person other than at an authorized Hewlett-Packard Customer Service Facility.

No other warranty is expressed or implied. Hewlett-Packard is not liable for consequential damages.

Beyond the one-year warranty period, your HP-65 will be repaired for a moderate charge. Return the HP-65 along with battery pack, recharger and travel case *(Refer to Shipping Instructions.)* If only the battery pack is defective, simply order a replacement on the Accessory Order Form provided.

Shipping Instructions

Malfunctions traced to the calculator or battery charger require that you return:

1. Your HP-65 with battery pack, recharger and travel case.
2. A completed Service Card (*from the back cover of this handbook*).

If a battery pack is defective and within warranty, return:

1. Only the defective battery pack.
2. A completed Service Card (*from the back cover of this handbook*).

Send items to be returned to the address nearest you shown on the Service Card, after packaging them safely. Should other problems or questions arise regarding service, please call the applicable service telephone number on the Service Card, or, if inside the U.S.A., call Advanced Products Division, Customer Service Department, at (408) 996-0100.

Index

B

C

D

E

F

G

P

R

S