HEWLETT-PACKARD

HP-65

Owner's Handbook

HEWLETT PACKARD

HP-65 Owner's Handbook

January 1974

00065-90200

PRINTED IN U. S. A.

Table of Contents

Introduction (4-17)

1. General Instructions (18-26)

2. Registers (27-35)

3. Functions (35-47)

4. Programming (48-78)

Appendices (79-96)

List of Illustrations

Introduction

Three Ways to Use the HP-65

Congratulations on purchasing your **HP-65 Programmable Pocket Calculator.** In addition to all the computational capabilities that have made the earlier HP-35 and HP-45 models so popular with professional people, your new HP-65 offers a feature that no other pocket calculator can provide: true programmability.

Simply defined, programmability is the HP-65's ability to learn, remember, and automatically execute the keystroke sequence required to solve a particular type of problem. The value of this feature becomes clearer when we consider that most of us who routinely work with numbers spend a great deal of time doing the same types of calculations over and over again. No matter whether we're preparing flight plans, surveying construction sites, calculating returns-on-investment, or designing power supplies, we can all identify repetitive, time-consuming problems which diminish our productivity and frustrate our goals.

Although programmable computers and desk-top calculators have been available for some time, their expense, complexity and non-portability have made them inappropriate or impractical for many tasks. The real significance of the HP-65 is that it overcomes these limitations and lets almost anyone enjoy the advantages—speed, accuracy, convenience—of a programmable calculating device.

You can use this powerful device in three ways:

1. To Calculate Manually

You control every step of the calculation by pressing keys in the actual order of execution: you enter data, perform functions, store results, control display, etc., by pressing keys.

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 highly complex calculations with minimal effort or study of the calculator itself. You load a card into the calculator and let the stored program handle the busy part of the calculation. Typically, you just key in the data and start the program running. The program stops when it needs more data or when it displays a result.

3. To Create, Record, and Execute Your Own Programs

No prior programming experience is necessary to program the HP-65. You can easily define the five *top row keys* to calculate functions of your own creation for use alone or with other programs. You plan your problem in terms of the keystrokes needed for calculation and the additional keystrokes needed to control your program. You set the mode switch to W/PRGM position and key the keystroke sequence into memory. You may then record your program for future re-entry by merely passing a magnetic card through the calculator. Upon switching back to RUN mode, you can execute your stored program.

In this introduction, we will briefly demonstrate these three methods. We suggest that you do the examples to confirm that your calculator works properly and to become familiar with it.

1. Calculating Manually

Getting Started

Your HP-65 Pocket Calculator is shipped fully assembled including a battery. Before using the calculator for portable use, charge it for 14 hours as described in **Appendix C.** You may run the calculator on battery power alone or you may connect the battery charger and run while the battery is charging. To get started:

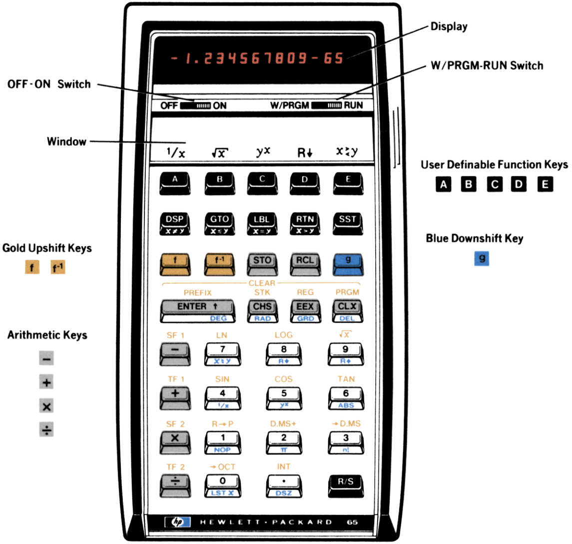
- Set W/PRGM—RUN switch to **RUN** position.
- Turn OFF—ON switch to **ON** position.

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

Keying In Numbers

Key in the number and include the decimal point if it is a part of the number. **For example,** try keying in 314.32 which would be done by merely pressing:

[3] [1] [4] [•] [3] [2]


Display
W/PRGM-RUN Switch
OFF-ON Switch
Window
User Definable Function Keys
A B C D E
Blue Downshift Key
g
Gold Upshift Keys
f f-1
Arithmetic Keys
- + × ÷
-1.234567809-65
OFF ON W/PRGM RUN
HEWLETT·PACKARD 65

If you make a mistake when keying in a number, clear the entire number by pressing **CLX** (clear X); then key in the number correctly.

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**

Performing Simple Arithmetic

In the HP-65 arithmetic answers are calculated by pressing **+** , **−** , **×** , or **÷** . For any problem having two numbers and one arithmetic operator, you key in the first number and save it by pressing **ENTER↑** ; then you key in the second number and follow it by the arithmetic operator. For example, add 12 and 3 by pressing:

12 **ENTER↑** 3 **+**

The calculator uses the last number saved and the last number keyed in: it adds the latter to the number saved; it subtracts the latter from the number saved; it multiplies the latter by the number saved; or it divides the latter into the number saved. For example, you can subtract 3 from 12 by pressing:

12 **ENTER↑** 3 **−**

To divide 12 by 3, press:

12 **ENTER↑** 3 **÷**

Nonarithmetic Functions

A **blue** symbol on the inclined lower key surface denotes the function of the key when preceded by the downshift **g** key. A **gold** symbol above the key denotes the function of the key when preceded by the upshift **f** key; the same gold symbol above a key denotes the inverse (or complement) of the function of the key when preceded by the **f⁻¹** key. To use a blue or gold function, press the appropriate shift key (**g**, **f**, or **f⁻¹**) immediately before pressing the selected key. For example, you

Compute	By Pressing	See Displayed
sin (90°)=1	90 **f** SIN	1.00
arc sin (.5)=30°	.5 **f⁻¹** SIN	30.00
1/5=.2	5 **g** 1/x	0.20

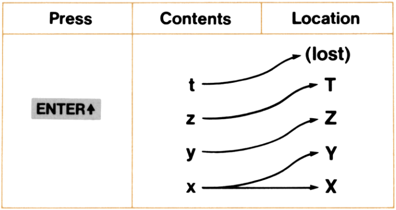
The Operational Stack

There are four working registers in the HP-65 called **X**, **Y**, **Z**, and **T**. They are arranged in a 'stack' with **X** on the bottom (*see below*).

Contents	Location
t	T
z	Z
y	Y
x	X

To avoid confusion between the name of a register and its contents, the register is designated in this handbook by a capital letter and the contents by a small letter. Thus **x, y, z,** and **t** are the contents of the **X, Y, Z,** and **T** registers.

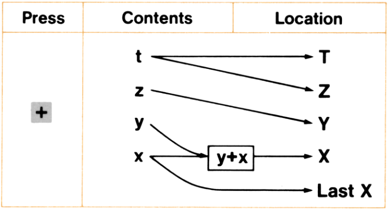
When you key in a number, it goes into **X**, the displayed register. When you press **ENTER↑**, this number is also reproduced in **Y**. At the same time **y** is transferred to **Z**, **z** is transferred to **T**, and **t** is lost (*see below*):



The HP-65 can save a number in each of the four registers.

Most problems can be solved by keying in the numbers in the same order as they appear in the original expression, that is, from left-to-right. To work a problem, key in the first number. If there is an operation you can perform at this point, do it. If there is not, press ENTER↑ . Now key in the next number. Perform any operation that can be done (+,—,×,÷, etc.). If there is no operation you can perform, ENTER↑ this number and repeat the procedure, keying in the next number. The following examples illustrate this procedure.

Arithmetic and the Stack. When you press the addition key the contents of **X** and **Y** are added together. The stack then drops, with **t** reproduced in **T** and **Z**, **z** transferred to **Y**, (y+x) transferred to **X**, and **x** transferred to **Last X.** (*Last X is described in* **Section 1.)**



The same dropping action takes place with any arithmetic operator (+,—,×, or÷); the result is placed in **X**.

Combined Arithmetic Operations. Anytime a new number is entered after an operation, the HP-65 performs an automatic ENTER↑ on the result of that operation. This feature allows you to work serial calculations as well as chain and mixed chain calculations. Notice that we implicitly used this in the following:

Sample Case:

[(4×5) / (2+3)] −6 = −2

Press	See Displayed
4 ENTER↑	4.00
5 ×	20.00
2 ENTER↑	2.00
3 +	5.00
÷	4.00
6 −	−2.00

Notice that the numbers are entered in the same order as they appear in the problem. Now consider the stack contents as we do the same example.

Stack Register

T											
Z						20	20				
Y		4	4		20	2	2	20		4	
X	4	4	5	20	2	2	3	5	4	6	-2
Keys	4	↑	5	×	2	↑	3	+	÷	6	−

Note: ENTER↑ is here abbreviated as ↑ .

Sample Case: $(12 \times 5) + (11 \times 4) + (10 \times 3) = ?$

Press	See Displayed
12 ENTER↑ 5 ×	60.00
11 ENTER↑ 4 × +	104.00
10 ENTER↑ 3 × +	134.00

More Computing Power

The calculator also has nine addressable registers so that the calculator can hold intermediate results or frequently used constants. This means that calculations of considerable complexity can be performed without reentering data or intermediate results. You now have some practice in calculating manually. We will consider these registers and further capabilities for manual calculation in the body of this handbook. In the meantime, let us move on to the question of running a prerecorded program.

2. Running a Prerecorded Program

A built-in magnetic card reader/writer allows a program to be permanently preserved on magnetic cards for future use. By reading such a card, your general purpose calculator gains a highly specific capability in a matter of seconds. Some users may wish to use professionally programmed cards without themselves doing any programming.

Irrespective of your major interests, we think that you may find a use for the Personal Investment Program, the first program in the Standard Pac shipped with your calculator. You will find the prerecorded magnetic card for this program in the card case, along with 18 additional programs, a head cleaning card, and 20 blank cards for recording your own programs.

The programs vary from general to specialized. Some programs were selected from other pacs available through HP. For example, the Pi Network Matching Program is from the **EE Pac I,** the Mean and Standard Deviation program is from the **Stat Pac,** etc. As leisure permits, you may wish to familiarize yourself with them all and work the numerical examples. The Personal Investment Program, however, is from no other pac. It was created for you, to allow you to calculate the growth of a regular monthly savings plan. Information about this as well as any of the pac's prerecorded programs is in the **Standard Pac Instruction Book:** what a program does, how to use it, etc. For our present purposes, we merely load the program and execute it, using sample data.

Loading the Program

1. **Select** the Personal Investment Card from the 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. If the card does not read properly, the display will blink and program memory will be cleared; press **R/S** and re-insert 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:

Sample Case: Growth of a Savings Plan

Starting on January 1, 1974, you add $100 per month to your savings of $1000 invested at 12% per annum, compounded monthly. How much will you have saved on September 1, 1975?

To solve the problem, just follow the instructions given in standard format in Figure 0-1. You read the "instructions," line by line, keying in the required "input," pressing the indicated "key(s)," and observing the displayed "output." The amount saved is displayed after the future date is entered via the **E** key.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
1	Enter program (Personal				
	Investment Program) as				
	shown above				
2	Key in start date (Jan., 1974)	1.1974	A		0.00
3	Key in present savings ($1000)	1000	↑		1000.00
4	Key in monthly savings (100)	100	B		1000.00
5	Key in annual interest rate (12)	12	C		1000.00
6	Key in future date (Sept., 1975)	9.1975	E		3444.11
					ANSWER
	NOTE: **ENTER↑** here is				
	abbreviated as **↑**				

Figure 0-1. *Instructions for Running Personal Investment Program*

A pad of program worksheet in this format is included with your HP-65 for use with the blank cards when you write your own programs.

3. Creating Your Own Program

Programmability and Definable Keys

Highly sophisticated calculations can be achieved by sequences of keystrokes. Since the calculator is truly programmable, including both branching and testing capability, it is quite possible to set a program to iterate all night. Programs can consist of up to 100 memory locations.

We have seen how the top row key functions can be defined to a particular use by loading an appropriately prerecorded magnetic card. Using a very simple example, we will now define the **A** key. We first plan the function, key it into memory, and then test it. If it tests satisfactorily, we will record it on a magnetic card for future use.

Planning the Function

The following key sequence computes x^3 (*the cube of whatever value k is in the* **X** *register*).

T				
Z		k		
Y	k	k	k	
X	k	k	k^2	k^3
Key	↑	↑	×	×

Note: ENTER↑ is here abbreviated as ↑.

To adapt the sequence to be a function that is callable by the **A** key, we precede the sequence by **LBL** **A** (*to identify the function*) and conclude the sequence by **RTN** (*to return control to the keyboard*).

Putting the Function in Memory

1. **Set** W/PRGM-RUN switch to W/PRGM and press f PRGM to clear the program memory.
2. **Press** the keys in the order shown:

Key(s)	Comment
LBL A	Defines beginning of function A.
ENTER↑ ENTER↑ × ×	Calculates x^3.
RTN	Defines the end of function A.

If (for now) you make a mistake, clear the program and start over. In section 1 you will learn how to correct mistakes and the meaning of the numbers in the display. The calculator has now "learned" to calculate x^3 when you press A in RUN mode.

Testing the Function

- Switch W/PRGM-RUN switch to **RUN**.
- Key in a number and press A. You should see the cube of the number.

Press	See Displayed	Comment
2 A	8.00	2^3
3 A	27.00	3^3
4 A	64.00	4^3
5 CHS A	–125.00	$(-5)^3$

Recording the Function

To record the program

1. Select an unprotected (unclipped) magnetic card.

2. Switch to W/PRGM. (*A reminder:* ***W****/PRGM stands for* **Write.**)
3. Pass the card through the right lower slot exactly as in loading a program (*above*). Providing the card is unprotected, it now contains your program.

   In the above example, we left keys **B** . . . **E** undefined.

   We could have keyed in definitions for them, also.

We have just shown how you can write a program for a simple function and identify it with one of the five user-definable keys. Your HP-65 can also be programmed without any reference to the top keys. You will see an example of this when you turn again to the subject of programming later in this handbook. It is easy to create simple functions. With very little additional effort, you can create functions or other programs of considerable complexity.

Onward

We hope that you have enjoyed your introduction to the HP-65 Pocket Calculator. The rest of the handbook presents the aspects of the calculator not covered thus far. In regard to the reference information (*enclosed in blocks, to distinguish it from the narrative*), it is probably sufficient to study this material casually on first reading, postponing a more thorough reading until completion of the entire book. You can quickly become familiar with the keyboard and gain assurance by merely keying in the numerous sample cases and making sure that you understand each of them.

Sections 1 through **3** elaborate on the subject of calculating manually. **Section 1** tells how to set the display and how to enter data in scientific format. **Section 2** tells how to use the addressable registers and to manipulate the stack, while **Section 3** tells how to use the built-in functions.

Section 4 covers the subject of programming, telling how to revise (edit) your programs, how to have them repeat themselves, to stop themselves, to make decisions, etc. The programming section also illustrates the use of the program worksheets shipped with your calculator.

The HP-65 Quick Reference Guide

The guide summarizes the more important procedures given in this handbook and explains the key functions, arranged in order of keyboard symbol for ease of reference. Use the guide to check details. It can be carried in the soft case.

If we have not answered all of your questions, contact your nearest HP office, or, if you are in the U.S., dial **(408) 996-0100,** *collect, and ask for* **Customer Service.** *We want you to be completely satisfied with your HP-65.*

Section 1

General Operations

In this section we will describe how to ▪ perform the clear operations ▪ control the display ▪ enter numbers in scientific notation ▪ recover from wrong keystrokes using the Last X feature. In addition, a reference block is devoted to the operation of the Stack Lift (*automatic enter*).

Clear Operations

Four separate clearing operations are possible with the 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 following keys (*not yet explained*) are also prefix keys:

STO RCL DSP GTO LBL

Clearing Stack Registers

f STK clears all four registers (**X**, **Y**, **Z**, and **T**) of the operational stack. To clear only the **X**-register, press CLX .

Clearing Addressable Registers

f REG clears all nine addressable registers. (*These will be described in a later section.*)

Clearing Entire Calculator

The entire calculator can be completely cleared by turning the power switch off, then on. When the power comes on, however, programs for the default functions corresponding to the window legends above the top row keys ([1/x] , [√x] , [yˣ], [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, program in execution, and program steps. Additionally, 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 the* **Programming** *section*).

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 setting selected. 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 ENTER↑	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	Comment
(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

Now return to eight decimal places in fixed notation.

Press	See Displayed	Comment
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 operation are attempted. Depressing any key stops the blinking without otherwise performing the key function. CLX is the recommended blink stopper. Figure 1-1 lists these improper operations.

Illegible Display

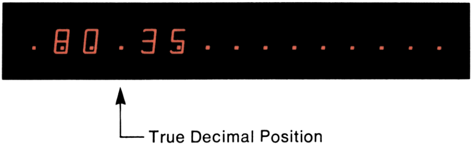
During execution of a stored program, the display continuously changes and is purposely illegible to indicate that the program is running. When the program stops, the display is steady.

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 the true decimal position.

Example:



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

Figure 1-1. *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 turn **off** the calculator before connecting the recharger to the calculator.

Also, be sure to start with at least partially charged batteries before using the card reader/writer.

Keying in Large and Small Numbers

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

Press	See Displayed	Comment
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

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

To see your answer in scientific notation with 6 decimal places,

Press	See Displayed
DSP 6	1.923077 04

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

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. sec. — and multiply it by 50.

Press	See Displayed
6.625 EEX	6.625 00
27	6.625 27
CHS	6.625 −27
ENTER↑	6.625000 −27
50 ×	3.312500 −25

If you return to DSP • 2 the result is rounded to zero.

Last X

Last X is the name of the register reserved for storing the latest value of **x** (*the number you see in the display*) just after an operation using it has been specified and prior to its use in a calculation. Initially set to zero when the power comes on, **Last X** remains unchanged until a calculation of **x** or **x** and **y** is attempted; at such a time, **x** is first saved in **Last X** as an automatic prelude to the calculation. The saved value is recallable to **X** (*repeatedly, if desired*) by the g LST X operation. Last X is 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 substract 3 from 12 and divided instead, you could compensate as follows:

Press	See Displayed	Comments
12 ENTER↑ 3 ÷	4.00	**Oops** — you wanted to subtract.
g LST X	3.00	Retrieves last number preceding division operation.

×	12.00	Reverses division operation: you are back where you started.
g LST X	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	Comments
12 ENTER↑ 3.157		You wanted to divide by **2.157**, not **3.157**.
÷	3.80	
g LST X	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.

Another, and possibly more important, use for Last X is in functions where **x** appears more than once. Without going into details since we have not yet discussed functions, examples might be:

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

In each case **x** is saved in Last X after the first operation is performed.

> The following operations (*including inverses*) save **x** 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 does not affect the Last X register.

Stack Lift Enable/Disable

We saw, in the Introduction, that when you key in a new number after a calculation, the calculated result is automatically lifted in the stack, relieving you of the need to save the result (*by pressing* ENTER↑) before keying in the number. The same lifting action occurs if you recall a value to **X** from a storage register, from the Last X register, or if you recall the permanently stored value of π. You may have observed that certain other operations also enable the Stack Lift while CLX and ENTER↑ disable the lift. You will generally be quite unaware of the lift status because the operation is so natural for most calculations. For reference, the keys affecting lift status are tabulated below. Notice that many operations have no effect on the Stack Lift. Most of the operations are yet to be presented in this handbook.

Operations that disable the Stack Lift: R/S within a program if the program puts a number into **X** from program memory just before executing the R/S , and CLX or ENTER↑ at any time.

Operations that enable the Stack Lift: *All number entry keys:* 0 . . . 9 , • , EEX , π , *but not* CHS .

- *All calculating keys:* − , + , × , ÷ , ABS , COS , →D.MS , D.MS+ , INT , LN , LOG , LST X , n! , →OCT , R→P , SIN , TAN , 1/x , √x , y^x
- *Stack manipulating keys:* R↑ , R↓ , x⇄y , but *not* ENTER↑
- *Storage register keys:* STO , RCL .

Operations not affecting Stack Lift status: All other keys have no effect on the lift status. They include: all programming keys, angular mode keys, display control keys, clear keys (except CLX), and CHS .

Section 2

Registers

In this section we will describe the use of the nine addressable storage registers and the manipulation of information in the stack. Also, to illustrate a use of the stack registers we will present a manual solution to a problem of compound interest.

Addressable Registers

Registers R_1, R_2, . . . , R_9 constitute the addressable registers. Their respective contents are referred to as r_1, r_2, . . . , 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 store in any register an arithmetic sum, difference, product, or quotient of the contents of the given register and the **X**-register. **For example,** if R_5 contains 100 and if **X** contains 70, you can store the difference ($100 - 70 = 30$) in R_5 simply by pressing STO - 5.

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 through 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 through 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 the lift is disabled.

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 $5,000. How much will the customer pay for each of the three items? What is the total cost?

Solution:

Press	See Displayed	Comment
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 item.
+ +	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	Comment
DSP • 4		
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 as to which registers you use.

R_8 is the special object of the *Decrement and Skip on Zero* (g DSZ) operation (*presented in* **Section 4**), which uses it as a descending counter (*index*) in program applications. If this use is contemplated, R_8 should be avoided for other uses. Otherwise, it may be freely used.

R_9 is subject to alteration by the trigonometric functions (*including the rectangular/polar conversions*) and the relational tests (*used in programs*). These functions use R_9 for intermediate calculations (*scratch*). At other times R_9 is available for your use.

The following operations destroy R_9:

SIN COS TAN R→P (trigonometric functions and their inverses)

x≠y x≤y x=y x>y (relational tests)

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 **x** is unchanged.

Subtraction.	To subtract x from r_n, **press**	STO − n
Addition.	To add x to r_n, **press**	STO + n
Multiplication.	To multiply x and r_n, **press**	STO × n
Division.	To divide x into r_n, **press**	STO ÷ n

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

Press	See Displayed	Comment
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.

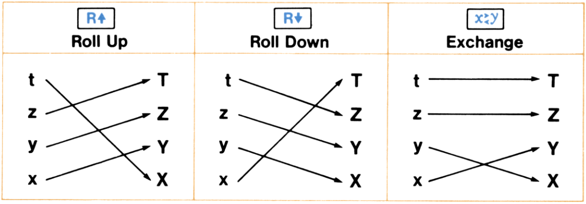
5 STO − 1	5.00	
RCL 1	3.00	Confirms that r_1 has been reduced to 3.

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.

Additional Stack Operations

You already know that, except for the case of the storage register arithmetic just described, all calculations are done in the stack. You simply put the problem numbers into place and press the appropriate function key. You already know of the possibility of keeping intermediate results in the stack above those being calculated. To lift information in the stack, you have used **ENTER↑** and the automatic lift. To drop the stack, you have used the arithmetic functions. We will now consider the three remaining operations to move information in the stack.



You use these operations to verify (*display*) the contents of stack registers other than **X** and to move information into place for calculation.

Notice that

- [R↑] allows you to conveniently see **t**.
- [R↓] saves **x** in the **T**-Register.
- [x⇄y] allows you to look at **y** without altering **t** or **z**.
- [R↑], [R↓] and [x⇄y] are functions appearing on the [9], [8], and [7] keys.

You may notice that [R↓] and [x⇄y] are also available on the **D** and **E** keys when the power is first turned **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, as in this case. They each permit single keystroke operation of functions that otherwise would require two keystrokes. When the **A**, . . . , **E** keys are redefined by a program, the window functions are still available by two keystrokes.

Sample Case: Turn the calculator **off,** then **on,** put the values 4, 3, 2, and 1 in the **T**, **Z**, **Y**, and **X**-registers, respectively, and review the stack using [x⇄y], [R↓], [E], [R↑], and [D].

T						4	4
Z				4	4	3	3
Y		4	4	3	3	2	2
X	4	4	3	3	2	2	1
Key	[4]	[↑]	[3]	[↑]	[2]	[↑]	[1]

Note: [ENTER↑] **is here abbreviated as** [↑].

T	4	4	4	4
Z	3	3	3	3
Y	1	2	1	2
X	2	1	2	1
Key	[g] [x⇄y]	[g] [x⇄y]	[E]	[E]
	$\underbrace{\hspace{6em}}$ **Using** [g] [x⇄y]		$\underbrace{\hspace{3em}}$ **Using** [E]	

Notice that using [g] [x⇄y] or [E] twice leaves the stack in its original condition.

T	1	2	3	4	1	2	3	4
Z	4	1	2	3	4	1	2	3
Y	3	4	1	2	3	4	1	2
X	2	3	4	1	2	3	4	1
Key	[g] [R↓]	[g] [R↓]	[g] [R↓]	[g] [R↓]	[D]	[D]	[D]	[D]
	Using [g] [R↓]				**Using** [D]			

Notice that using [g] [R↓] or [D] four times leaves the stack in its original condition.

T	3	2	1	4
Z	2	1	4	3
Y	1	4	3	2
X	4	3	2	1
Key	g R↑	g R↑	g R↑	g R↑

Notice that using R↑ four times leaves the stack in its original condition.

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	Comment
g π	3.14	Recall π to X
3 ENTER↑ ×	9.00	Calculate 3×3
×	28.27	Answer $9 \times \pi$

Innovative Use of Calculator — A Compound Growth Schedule

There is considerable room for innovation and creativity in using HP pocket calculators. Our customers have amazed us by solving problems using highly original manipulations on the calculator. Here is such a routine to calculate a geometric series, showing the compound growth of invested capital.

A geometric series is a set of numbers in which each term is calculated by multiplying the previous term by some factor. **For example:** 4, 8, 16, 32, etc. In this case, the factor is 2. In the practical world, 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. Our customer's solution to generating the series was:

Press	See Displayed	Comment
1.10 **ENTER↑** **ENTER↑** **ENTER↑** 1000	1000.00	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
etc.		

In other words, you put the growth factor (1.10) in the **Y**, **Z**, and **T** stack 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 1000 × 1.10 (i.e., x×y). The result (1100.00) is displayed in the **X**-register and a new copy of the growth factor (1.10) drops into the **Y**-register. Since a new copy of the growth factor is generated each time the stack drops, you never have to reenter it.

Contents		Register	
t (1.10)	→	**T** (1.10)	Stack contents before and
z (1.10)	→	**Z** (1.10)	after first multiplication
y (1.10)	→	**Y** (1.10)	
x (1000)	→ (x × y) →	**X** (1100)	

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

Taken as a whole, the functions are both powerful and important. While, conceivably, you might not use them directly, you will almost surely use them indirectly when you use preprogrammed cards from the Standard Pac and other pacs. **For example,** you already used the exponential function (y^x) and the integer function (INT), when you used the Personal Investment program presented in the Introduction. Without y^x , the program would have to use more laborious arithmetic methods, repeatedly.

The functions are both easy to learn and easy to use. In the **Introduction** you found that to do a function, you press a prefix key (f , f^{-1} , or g) and follow it by 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.

Using this rule and common sense, you may have already calculated several functions effectively. Missing from the presentation thus far, has been a systematic review of just which functions are available, and the respective conditions that apply to them individually. To meet this need, all essentials are included in Figures 3-1, 3-2, and 3-3.

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/from degrees, minutes, and seconds.

Angular Mode

Operations involving angles 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.

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 (|x| ≤ 1)	Principal value of arc cosine (x) in X (0° ≤ result ≤ 180°)**
f SIN	Sine	Angle*	Sine (x) in X
f^{-1} SIN	Arc sine	x not be greater than 1 or less than −1 (|x| ≤ 1)	Principal value of arc sine (x) in X (−90° ≤ result ≤ 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° ≤ result ≤ +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 (x + y) 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 99999.59599 in DDDDD.MMSS format.

Figure 3-1. *Functions Involving Angles*

Degrees, Minutes, Seconds

You can convert from the decimal form of an angle to degrees, minutes, seconds. You can also do the inverse. 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	Comment
DSP • 4		Set display to four fixed places.
g π 7 ÷	0.4488	$\pi/7$
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	Comment
g GRD	25.4251	Set grad mode.
f^{-1} →D.MS	28.5713	Answer in grads.

Sample Case Part 3. Now convert to decimal degrees.

Press	See Displayed	Comment
g LST X	25.4251	25° 42′ 51″.
g DEG	25.4251	Set degree mode.
f^{-1} →D.MS	25.7142	Answer in decimal degrees.
DSP • 2	25.71	Reset display.

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

Press	See Displayed	Comment
DSP · 4	0.0000	
45.1050	45.1050	
ENTER↑	45.1050	
44.4910	44.4910	
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	Comment
DSP · 4	0.0000	
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	Comment
g DEG 60	60.	
f COS	0.50	Answer.

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

Press	See Displayed	Comment
g RAD 1 CHS	−1	
f^{-1} COS	3.14	Answer in radians.

Compute *sine* 30°.

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

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

Press	See Displayed	Comment
g RAD 1	1.	
f^{-1} SIN	1.57	Answer in radians.

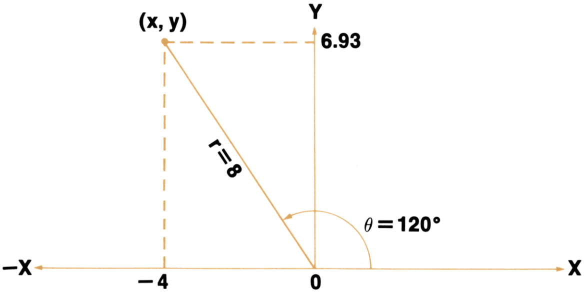
Compute *tangent* 45°.

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

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

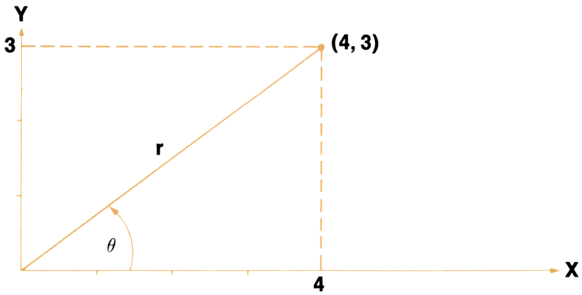
Press	See Displayed	Comment
g RAD 39.4	39.4	
f^{-1} TAN	1.55	Answer in radians.

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



Press	See Displayed	Comment
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 angle expressed in degrees:



Press	See Displayed	Comment
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).

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 is negative, −x in X; 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 and leave the result there. **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	Comment
512 [f] [→OCT]	1000.00	Octal representation of 512_{10}.

Convert the octal number 2000 to its decimal equivalent:

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

Sample Case: *Truncating at Decimal Point*. The Personal Investment Program (*presented in the Introduction*) expects you to key in the dates using the format *mm.yyyy*. The program separates *mm* from *yyyy* using the truncation functions. Do the same for the date 12.1980.

Press	See Displayed	Comment
[DSP] [•] [4]		
12.1980	12.1980	
[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	Comment
3 [g] [ABS]	3.00	$\lvert +3 \rvert$
[CHS]	−3.00	
[g] [ABS]	3.00	$\lvert -3 \rvert$

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

These functions are listed in Figure 3-3. They all leave the result in the **X**-register. 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	Comment
10 [ENTER↑]	10.00	Save value 10.
2 [f] [LOG]	.30	Log 2.
[×]	3.01	Answer.

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 and unrestricted x $(y>0)$	y^x in X; stack drops.

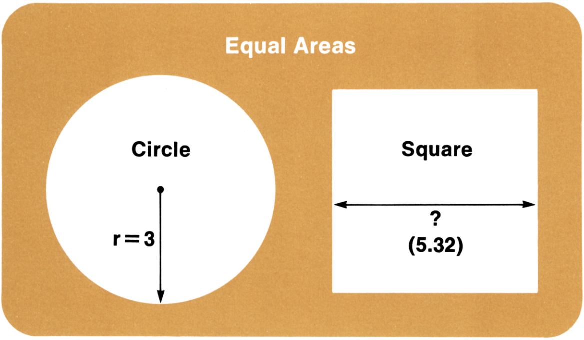
Figure 3-3. *Functions of x and the Exponential Function* (y^x)

Sample Case: *Natural Antilogarithm*. Display the constant e to nine places ($e = e^1 =$ *natural antilog 1*).

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

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	Comment
g π	3.14	π
3 f⁻¹ √x	9.00	3^2
×	28.27	Area of circle.
f √x	5.32	Size of square.

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

Press	See Displayed	Comment
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. We need only

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

Section 4

Programming

In the **Introduction** you learned to run a prerecorded program and to create a simple program. You have since explored the preprogrammed abilities of the calculator. This section provides the information for creating more ambitious programs, comparable in scope to the recorded programs in the *Standard Pac* and other pacs. You are encouraged to create and use your own programs even though you otherwise take full advantage of your calculator's power by merely using prerecorded programs. We think you will find it as exciting as it has been for us.

To create a program, you need to:

1. **Define** the problem.
2. **Work** out the keystroke sequence that solves the problem.
3. **Add** control operations for automatic execution.
4. **Key** the keystroke sequence, including control operations, into program memory.
5. **Edit, verify,** and **record** the sequence for later use.
6. **Run** the sequence, automatically, with your data.

To **key** a program into the machine, **press** the successive keys with the switch in **W/PRGM** position. Then, by passing an unprotected magnetic card through the right lower slot of the calculator, you can save the program (*contents of the 100-step program memory*) for future use.

The subject is discussed under four major headings: ■ **Looking at a Program.** ■ The **Control Operations** needed in programs to start, to repeat, and to stop. ■ The **Editing Operations** that allow you to correct and change programs in memory. ■ **Test Operations** that allow your program to make decisions.

Looking at a Program

Quite obviously, it is not possible to see the entire program at once; you see one step at a time as determined by the program pointer (*defined in Figure 4-1*). Recall that program memory consists of 100 locations. Above the first location is the top of memory, which displays as:

Top of Memory Display

Whenever you see this display, you know that the program pointer is at the top.

To see this, turn the calculator **OFF,** then **ON,** and switch to **W/PRGM.** This clears any programs and replaces them with the default programs.

To move the pointer to display the next (*first*) location:

Press	**See Displayed**
SST	23

Except for the digit keys, the display indicates the row and column of the key that the display represents.

Thus 23 is read as *row two, column three* or LBL.

Column 3

Row 2

For ease of recognition, the digit keys 0, . . . , 9 are displayed simply as 00, . . . , 09. You can now read the remaining contents of memory.

Press	See Displayed	Comment
	23	Represents LBL.
SST	11	Represents A.
SST	35	Represents g.
SST	04	Represents 1/x*.
SST	24	Represents RTN.
Etc.	Etc.	

Thus, starting at the top, you can now see that the first five codes (*23, 11, 35, 04, 24*) represent the default function defined for the top row A key. This function, LBL A g 1/x RTN, does nothing more than compute the reciprocal of x.

The SST key is discussed in Figure 4–1. **Note** that SST is used also in **RUN** mode to execute a program, one step at a time.

Also, note that to conserve memory, the most frequently used prefix-suffix pairs are each merged into single codes. Thus, g LST X is encoded and displayed as:

35 00

Merged Code Display

Control Operations

In creating a program, you take into account how it is to be started and how it is to stop. You may recall that in the function created in the Introduction, you put a label (LBL A) ahead of the actual calculating steps so that the beginning of the program could be found. You also put a return (RTN) at the end so that the program could stop itself:

LBL A ENTER↑ ENTER↑ × × RTN

* Notice that 35, representing the g prefix, determines that the next code 04 be interpreted as 1/x, the blue alternate function on the 4 key.

Thus programmed, when you pressed the **A** key (*in RUN mode*), the calculator searched the program memory for the corresponding label (**LBL** **A**). Upon finding the label, the calculator executed the steps following the label, one after the other in sequential order, until the return (**RTN**) told the calculator to stop.

Figure 4-2 summarizes the above operations used to define functions.

Program Memory: Program memory contains the user's stored program.

- Capacity: 100 locations
- *Top* is above first location.
- *Bottom* is last location.

Pointer: The pointer is an internal part of the calculator. It determines which memory location is executed or displayed.

Codes: In W/PRGM mode, keystrokes are stored in memory as codes: Top of memory code is 00 00. The codes for keys 0-9 are 00-09. For other keys, a code denotes a row and a column. **Example:** Code for **R/S** (row 8, column 4) is 84.

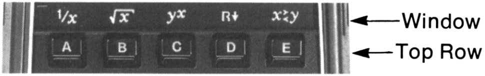
Merged Codes: Program codes for the following are merged with their respective prefix codes: LST X, NOP, x⇄y, R↓, R↑, x≠y, x≤y, x=y, x>y, and 1, . . . , 8 when prefixed by STO or RCL. ■ **Example:** g LST X in program mode is merged and displayed as 35 00; STO 5 as 33 05, etc. Note that STO 9 and RCL 9 are not merged.

Single Step: ■ In W/PRGM mode, **SST** advances the program pointer to the next memory location, displaying the code. Repeated use of the key enables you to review a program and to position the pointer for editing. ■ In RUN mode **SST** executes the program step denoted by the program pointer. In the case of single stepping a call to a user defined function, (**A**, . . . , **E**, the entire function executes (as one step) before returning control to the keyboard.

Figure 4-1. *Memory, Codes, and the Single Step Key*

LBL **Label** (prefix). **LBL** identifies its suffix (a digit [0], . . . , [9], or top row key, **A**, . . . , **E**) as a label in a stored program. A branch to the labeled part of the program can then be done by executing **GTO** followed by the same suffix. For user-defined functions, the label suffix must be a top row key (**A**, . . . , **E**).

A, **B**, **C**, **D**, **E** **Top Row Keys.** These keys can be redefined by inserting a program into memory. When used without a prefix, a top row key finds and executes the corresponding user defined function.



Default Functions. When the calculator is turned on, default functions as defined in the window above the top row are automatically inserted into memory. **f** [PRGM] clears these functions so that alternate functions may be keyed in.

RTN **Return.** ■ If executed manually from the keyboard, **RTN** merely resets the program pointer to the top of memory. ■ In a stored program, **RTN** is the logical end of a user defined function.

■ If a function is executed from the keyboard, **RTN** *stops* the program.

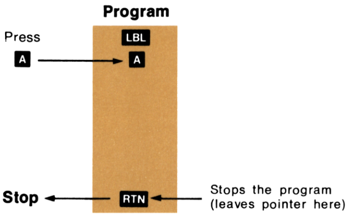
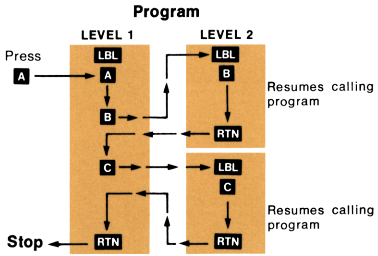
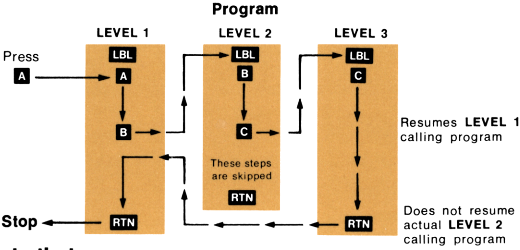


Figure 4-2. *Control Operations for Defining Functions*

- If a function is executed in a program, execution of **RTN** *resumes* the program.



- A function executed from the keyboard or a nonfunction program can execute another function. The latter, however, cannot properly execute yet another function.



Note that:

1. **LBL** can prefix a digit as well as a top row key, thus allowing up to 15 unique labels. We shall presently use a digit label in a program that repeats.
2. **RTN** can be used if pressed manually to move the program pointer to the **top of memory.** We shall use this to start execution at the very first step in program memory.
3. If a program calls another programmed function, the **RTN** terminating the function does not cause a stop, but rather resumes the sequential execution of the calling program.

Figure 4-2. *Control Operations for Defining Functions (cont.)*

Sample Case: Repeating and Stopping

Assume that you get a 15% discount (*i.e., you pay 85%*) on the following purchases:

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

Create a program at the top of memory to display the cumulative cost.

Only two unfamiliar operations (GTO and R/S) are needed. They are summarized in Figure 4-3.

Method. After clearing the stack once at the start, let the program stop to accept data and to display any previous accumulation (zero, at first). Then key in:

Quantity ENTER↑ Price R/S

Have the program then calculate the discounted cost ($5 \times 2 \times .85$, *the first time*), adding the cost to the previously accumulated total. Repeat this for the second, third, and fourth pairs which are to be entered after the respective stops. The flowchart (p. 55), shows the process.

Writing the Program

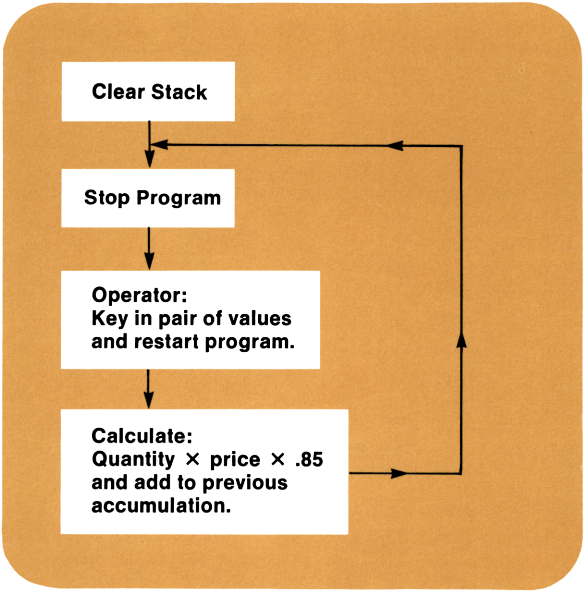
1. To **key** in the program set the mode switch to **W/PRGM.**
2. **Press** f PRGM to clear program memory.
3. **Key** in the program.

Again, for now, if you make a mistake, clear memory and start over.

Press	Comment
f STK	**Clear** the stack.
LBL 9	**Identify** place to start repetition.
R/S	**Stop** the calculator. During the stop, **key** in a pair of values and **press** R/S to **restart** the program.

✕ • 8 5 ✕ +	Calculate quantity × price × .85 for one pair and add product to previous accumulation.
GTO 9	Repeat, starting at LBL 9.

Notice that we arbitrarily chose LBL 9 to mark the part to repeat.



R/S RUN STOP. ■ If R/S is keyed in and a stored program is not executing, the stored program starts executing at the step denoted by the program pointer. ■ If executed in a stored program, R/S stops the program, displaying the **X**-register and allowing keystrokes from the operator. ■ If a R/S in a program is immediately preceded by a numerical entry from the program, the automatic lift is disabled upon return to the keyboard. This allows a program to display prompting information that will not be lifted in the stack if you enter a number from the keyboard. Except for this case, R/S does not affect the stack lift.

GTO Go to (prefix). When followed by a digit (0, . . . , 9) or a letter (A, . . . , E), GTO advances the program pointer to the first occurrence of the corresponding program label (i.e., LBL followed by the same digit or letter).

Figure 4-3. *Control Operations for Stopping and Branching*

The function programming sequence of LBL, . . . , RTN was not used. The method shown conserves function labels, using GTO instead. By this method you could still define all five functions, keeping all six programs in memory simultaneously (*within the 100 step limit*).

Running the Program

Switch to **RUN** mode and

Press	See Displayed	Comment
RTN R/S	0.00	Start program at top of memory.
5 ENTER↑ 2 R/S	8.50	5 × 2 × .85
7 ENTER↑ 4 R/S	32.30	7 × 4 × .85 + 8.50
8 ENTER↑ 5 R/S	66.30	8 × 5 × .85 + 32.30
22 ENTER↑ 6 R/S	178.50	22 × 6 × .85 + 66.30

Editing Operations

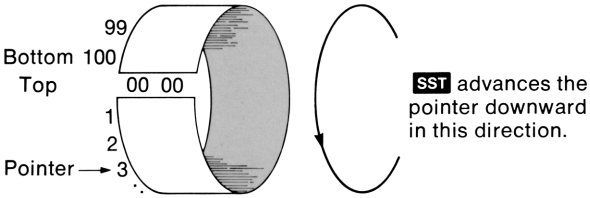
You edit with the mode switch set to W/PRGM. The edit operations make it easier to key in programs because, in case of mistakes, you can correct a wrong step by (1) stepping through your program using SST until the wrong step is in the display, (2) pressing g DEL (delete program step) and by reentering the step correctly.

To insert an operation following the one on display, just key it in. Now you will be able to step through your program and insert a step *between* any two steps.

Revising a Program

Sample Case. To demonstrate the editing process, let us revise the default A function to calculate the factorial function instead of the reciprocal function. The desired function will be LBL A g n! RTN, which is identical to what is already in memory except for the fourth key. We need only delete the 1/x and insert n!.

Editing Procedure: Before we can delete 1/x, we must move the pointer to 1/x (*code 04*). Conceivably we could repeatedly press SST in W/PRGM mode. This would advance the pointer to the **bottom of memory,** through the **top,** and to the first location, etc. in circular fashion.



It is easier, however, to switch back to RUN and to move the pointer to the top of memory (*by pressing* **RTN**) or to the label (*by pressing* **GTO** **A**). Let us do the latter:

Switch the calculator **OFF,** then **ON.**
Switch to **RUN.**
Press **GTO** **A**.
Switch to **W/PRGM.**

Press	See Displayed	Comment
	11	A
SST	35	g
SST	04	1/x

To delete the 1/x :

Press	See Displayed	Comment
g DEL	35	04 has been deleted.

Notice that the pointer has backed up to the g (code 35). To insert the n! , just

Press	See Displayed	Comment
n!	03	03 has been inserted.

Testing the Revision. To verify that the **A** key is redefined to calculate the factorial, switch to RUN and

Press	See Displayed	Comment
5 A	120.00	5!=120

Figure 4-4 summarizes the editing process. You can tell when memory is full by observing the display:

23 -

One "–" sign indicates full memory.

Note that if memory is full, and if you try to delete a step, you will also delete the bottom code; be sure to reinsert it.

You can tell when the pointer is at the bottom:

- 23 -

Two "–" signs indicate pointer is at bottom.

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. Deleting the bottom step also deletes the step ahead of it. For a critical case like this, be sure to reinsert the last step.

Test Operations

To complete the discussion of programming the HP-65, we will consider the test operations summarized in Figure 4-5. The test operations are particularly valuable for performing iterative calculations.

Using the Flags for Programmed Decisions

The calculator has two flags (*called flag 1 and flag 2*) available for your use. A flag is an invisible piece of information with just two possible conditions: **on** or **off.** The flag operations are given in Figure 4-5, p. 63. You can set a flag **on** or **off** by using the Set Flag operations. These operations can be executed from the keyboard or from a program. The reason for setting a flag is so that a program can later make a decision based on the condition of the flag (*using the test flag operations*).

Sample Case: *Flags.* Create a function A that computes $(1/x)^2$ if flag 1 in **on** and computes (x^2) if flag 1 is **off.** Assume that the desired condition of flag 1 has been previously set.

- **Switch** to **W/PRGM**
- **Clear** memory by pressing f PRGM .
- **Key** in the following steps.

Keys	**Comment**
LBL A	
f TF 1	Test flag 1 for on.
g 1/x *	If on, calculate (1/x).
f^{-1} √x	Square x or 1/x.
RTN	Stop.

* If flag 1 is **on,** these steps are not skipped. If **off,** they are skipped and x is **not** replaced by 1/x.

Positioning the Pointer. ■ To move the pointer to the top of memory, press RTN in RUN mode. ■ To move the pointer to a label, press in RUN mode, GTO n, where n is the same digit or top row key as in LBL n (the label). ■ To step through your program, use SST in W/PRGM mode. You will see the successive program codes in the display.

Insert: ■ Pressing a key in W/PRGM mode stores the instruction code in program memory between the displayed code and the following instruction code and moves the pointer to display the code just inserted. The bottom location drops off. ■ **Insert is not performed:** (1) For PRGM, DEL, SST. (2) For the second key of a merged code. (3) When the pointer is at the bottom.

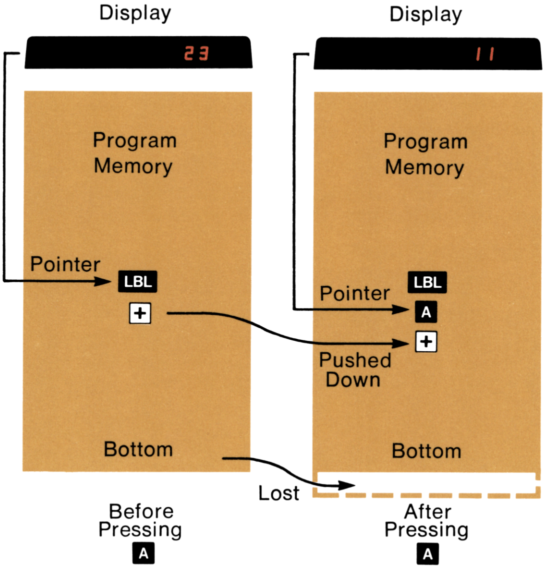
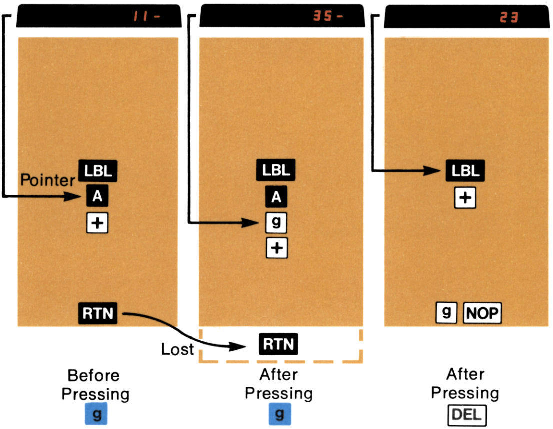


Figure 4-4. *Editing Operations*

DEL **Delete Program Step.** ■ With the switch set to W/PRGM, g DEL (1) deletes the program step denoted by the program pointer, (2) moves all the following steps up one step, and (3) inserts a g NOP code in the vacated bottom position of memory. ■ When switched to RUN, g DEL acts only as CLX. ■ g DEL can be used to back up the pointer (after which you reinsert the deleted codes). ■ If *memory is full* (i.e., a minus sign shows in display), g DEL causes the bottom memory step to be lost. ■ If *the program pointer is at the bottom* (i.e., two minus signs show in display), g DEL deletes two locations.



PRGM **Clear Program Memory.** In program mode, f PRGM or f^{-1} PRGM clears the entire program memory to "no operation" codes (35 01), leaving the *pointer* at the *top of the memory*. ■ In run mode, f PRGM is equivalent to CLX.

Figure 4-4. *Editing Operations (cont.)*

Running the Sample Case.

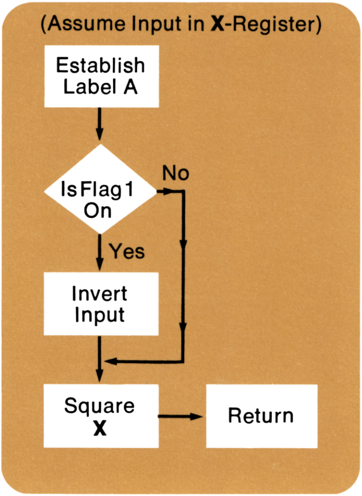
Switch to **RUN.**

Press f SF 1 to turn flag 1 on so that the square of the reciprocal will be computed.

Press	See Displayed	Comment
2 A	0.25	$(1/2)^2$
.5 A	4.00	$(1/.5)^2$

Press f⁻¹ SF 1 to turn flag 1 **off** so that the square will be computed.

Press	See Displayed	Comment
2 A	4.00	(2^2)
.5 A	0.25	$(.5^2)$



DSZ . **Decrement and Skip on Zero.** Subtracts 1 from an integer in register R_8, then skips two program memory locations if R_8 contains zero. ■ The decrement operation is suppressed outside the limits: $1 \leq |r_8| \leq 10^{10}$. ■ Useful for looping.

x≠y , x≤y , x=y , x>y . **Relational tests of x and y.** Each test compares the values in the X and Y registers, and skips two memory locations if the test condition is not met. The tests use R_9 and alter the contents.

NOP **No Operation.** Useful as a filler in tests ■ f PRGM in W/PRGM mode clears the entire memory to g NOP (merged code 35 01).

SF 1 , SF 2 . **Set Flag 1, Set Flag 2.** ■ f SF 1 sets flag 1 *on* while f⁻¹ SF 1 sets it *off.* ■ f SF 2 performs similarly, but using flag 2. Initially turned off when the calculator is turned on, flags retain their settings until changed by these operations.

TF 1 , TF 2 . **Test Flag 1, Test Flag 2.** ■ f TF 1 tests flag 1, skipping 2 memory locations if flag 1 is *off*, while f⁻¹ TF 1 skips if flag 1 is *on*. ■ f TF 2 performs similarly, but uses flag 2.

Figure 4-5. *Operations Used in Programmed Decisions*

Relational Test Operations

The four relational tests allow you to program a decision based on the relationship of x to y. This can be valuable in iterative calculations or in simpler applications such as the following.

Sample Case: *Relational Test*. Create a function B to calculate the arc sine of an input value x (*x must be within the limits of —1 and +1*). If the resulting angle in decimal degrees is negative or zero, add 360 degrees to it.

Switch to **W/PRGM.**

Clear memory by pressing f PRGM .

Key in following sequence.

Keys	Comment
LBL B	
f⁻¹ SIN	Compute arc sine.
0	0 to X lifts arc sine to Y.
g x⇄y	Put 0 in Y and arc sine in X.
g x≤y	
3	Test and skip these steps if angle is positive.
6	
0	Add 0 or 360.*
+	
RTN	Stop.

Running the Function

Switch to **RUN**

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

Looping N Times

The DSZ operation is useful in repeating a labelled segment of a program a given number of times. The repeated segment is called a loop. **Rule:** To execute a labeled program segment n times, preset n in R_8 and end the segment with a DSZ to determine whether or not to repeat the segment.

* If the angle is negative, 3 6 0 are put into the X register as 360.
If the angle is positive, 3 6 are skipped and 0 puts zero into X.
Thus, either 0 or 360 are added to the angle.
Step saving techniques like this will undoubtedly occur to you in writing your own programs.

Sample Case: *Sum of First n Digits*. Calculate the sum of 1 + 2 + 3 + . . . + n where n is in the x-register at the beginning.

Switch to **W/PRGM.**

Press f PRGM to clear memory.

Key in the following steps.

Keys	Comments
LBL A	
STO 8	Preset n in R_8.
f STK	Clear stack.
LBL 1	Beginning of repeat segment.
RCL 8	Recall r_8 to X.
+	Add to previous sum.
g DSZ	Decrement r_8 (r_8 — 1→ R_8) and test for zero.
GTO 1	Repeat if r_8 is not zero.
RTN	Stop after nth iteration.

Calculating the Function

Switch to **RUN.**

Press	See Displayed	Comment
5 A	15.00	Sum of 1 + 2 + . . . + 5.
20 A	210.00	Sum of 1 + 2 + . . . + 20.
25 A	325.00	Sum of 1 + 2 + . . . + 25.

A Complete Problem

Thus far we have covered all of the operations possible in programs, illustrating them with small, individual examples. In the real world, things are not quite so simple. Consequently, we present a real problem and will take you through the programming using the following aids provided with your HP-65.

- HP-65 Program Form
- HP-65 User Instruction Form
- HP-65 Blank Magnetic Cards
- HP-65 Pocket Instruction Card

Assume that you need $4000 to make a downpayment on a house and your current savings are $2000. Assume further that you are able to add the amount of $185 to your savings monthly and that your total savings earn 1% per month. By delaying purchase of the house, the required downpayment increases through inflation by 0.5% per month. Given these assumptions, answers to the following questions are desired:

- What is the new required downpayment each and every month?
- What are your total savings each month?
- What is the difference between savings and required downpayment each month?
- When will you be able to purchase the house (assuming it is still on the market)?
- What is the difference in a particular month (*possibly to see if a projected bonus could make it possible to purchase earlier)?*

To start, we will break this seemingly complex problem into manageable pieces and, following the steps outlined at the beginning of this section, create a program to give us the desired answers. Conveniently, there are five answers desired. One way to look at the problem is to have keys **A** , . . . , **E** each defined to provide one answer. We shall do this.

Label A: Given $4000 required downpayment increasing at 0.5% per month. What is the monthly total downpayment? You may recall that this problem is that of calculating a monthly compound growth schedule with a growth (*inflation*) factor of 1.005.

Label B. Given \$2000 initially plus \$185 per month invested at 1% per month. What is the monthly total savings? This also is a compound growth schedule with a growth (*interest*) factor of 1.01 but with an addition of \$185 monthly to the amount being compounded.

Label C: This is simply the difference between required downpayment and savings (*label A* **minus** *label B*).

Label D: What is the number of months required for the savings to overtake the required downpayment?

Label E: What is the difference between savings and required downpayment at a specified month—say 8 months hence?

Figure 4-6 shows the monthly answers for labels **A, B,** and **C** for the first 12 months.

Month	Label A Required Downpayment (4000)	Label B* Savings (2000)	Label C Difference
1	4020.00	2205.00	−1815.00
2	4040.10	2412.05	−1628.05
3	4060.30	2621.17	−1439.13
4	4080.60	2832.38	−1248.22
5	4101.01	3045.71	−1055.30
6	4121.51	3261.16	− 860.35
7	4142.12	3478.77	− 663.34
8	4162.83	3698.56	− 464.27
9	4183.64	3920.55	− 263.09
10	4204.56	4144.75	− 59.81
11	4225.58	4371.20	145.62
12	4246.71	4599.91	353.20

*Note that the calculation assumes the monthly amount to be deposited at the end of the month; this is an **ordinary annuity** in business parlance. The Personal Investment Program in the **Standard Pac,** on the other hand, assumes the monthly amount to be deposited at the beginning of the month; this is an **annuity due** in business parlance.

Figure 4-6. *Calculation Results*

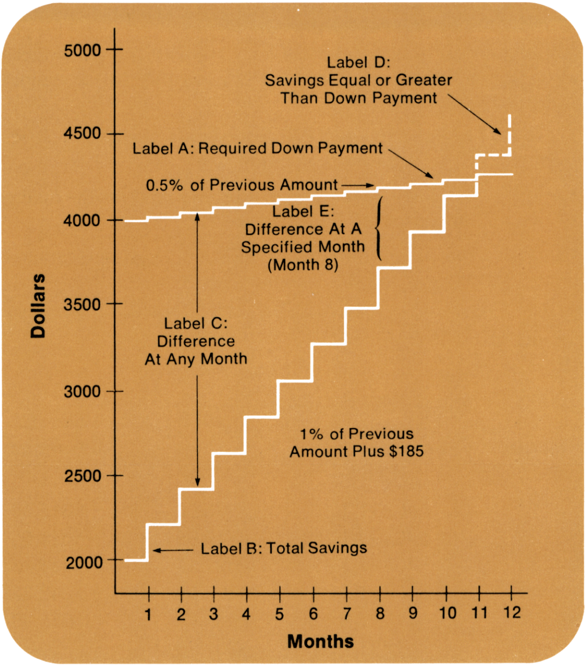


Figure 4-7. *Graphic Representation of Program Tasks*

Figure 4-7 is a graphic presentation of the problem with the specified values.

A good programming technique is to permit variation of the input values (*you might only be able to save $110 per month after carefully considering all of your actual expenses and in-*

come). We shall accomplish this by manually storing all of the values in registers prior to performing the calculations using the program. Assume that the following registers are used:

Initial downpayment	($4000)	STO 1
		STO 2
Monthly inflation factor	(1.005)	STO 3
Initial savings	($2000)	STO 4
		STO 5
Monthly savings	($185)	STO 6
Monthly interest factor	(1.01)	STO 7

Note that the initial downpayment is stored in both R_1 and R_2. This is because in our program r_1 will remain the same as the initial value but R_2 will contain the downpayment in successive months. We can at will reset R_2 to the initial value saved in R_1 and start over again. The same reasoning holds for R_4 and R_5 containing the initial savings.

One final word before we actually start our program. Normal procedure is to write the entire program sequence, enter it, and then test it. For the sake of greater understanding, we will write one label at a time and obtain the answers for that segment of the problem as we go along.

Programming Label A : Monthly Total Downpayment

1. **Switch to W/PRGM.**
2. **Press** f PRGM to clear memory.
3. **Key** in the sequence of steps below.

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
A	11	
RCL 2	34 02	Dnpmt
RCL 3	34 03	Inflation factor
x	71	
STO 2	33 02	New dnpmt
RTN	24	

Dnpmt × 1.005 = New Dnpmt

If you make any errors, use the editing procedures to make corrections.

To operate:

4. **Switch** to **RUN.**
5. If you have not already done so, store your input variables in R_1 through R_7. Follow the user instruction below:

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
	To see dnpmt for months 1, 2, 3, etc.		A		4020.00
	repeatedly press		A		4040.10
			A		4060.30
			ETC.		ETC.

Programming Label B : Monthly Total Savings

1. **Switch** to **W/PRGM.** You should see 24, the code for RTN. Since in LBL A there are no instructions following RTN, you can immediately key in the sequence below. **Do not press** f PRGM.

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
B	12	
RCL 5	34 05	Savings
RCL 7	34 07	interest factor
×	71	
RCL 6	34 06	monthly savings
+	61	
STO 5	33 05	New savings
RTN	24	

(Savings × 1.01) + $185 = New Savings

To operate:

2. **Switch** to **RUN.**
3. Follow the instructions below.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
	To see savings for months 1, 2, 3, etc.		B		2205.00
	repeatedly press		B		2412.05
			B		2621.17
			ETC.		ETC.

Programming Label C : Difference Between Savings and Down-payment.

Here we are looking for successive values of the difference between the label A and label B calculations for each month.

The program has to start by recalling the initial values for savings and downpayment so that the difference for month 1, 2, 3, . . . etc. can be calculated. This means that we cannot press C and get successive values for the difference. Therefore, a slightly different technique using looping and R/S is used in writing the program.

1. **Switch** to **W/PRGM.** Do not press f PRGM.
2. As before, you should see 24, the code for RTN at the end of LBL B. You are ready to enter the sequence below.

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
C	13	
RCL 1	34 01	Reset savings and
STO 2	33 02	dnpmt to
RCL 4	34 04	initial conditions
STO 5	33 05	
LBL	23	
0	00	Return point
B	12	Calculate savings
A	11	Calculate dnpmt
−	51	difference
R/S	84	
GTO	22	
0	00	

Savings − Dnpmt = Difference

Note that Label C first calls B to calculate the savings and then calls A to calculate the downpayment before subtracting and stopping.

To operate:

3. **Switch** to **RUN.**
4. Follow the instructions below.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
	To see difference for month 1,				
	press [C]. To see successive		C		−1815.00
	differences, repeatedly press		R/S		−1628.05
			R/S		−1439.13
			ETC.		ETC.

Programming Label **D** : Months for Savings to Overtake Downpayment

Instead of calculating monthly a sequence like before, the task here is to calculate a specific answer—the month at which savings are great enough to permit purchase of the house. We will set up a counter for the number of months in R_8 and use a logical test ($x>y$, downpayment > savings) to decide if we should continue for another month or if the increased savings has overtaken the required downpayment.

1. **Switch** to **W/PRGM.** Do not press **f** **PRGM**.

2. See 84, the code for **R/S** . Since **R/S** was not the last instruction in **LBL** **C** , press **SST** , see 22, the code for **GTO** .

3. Now press **SST** , see 00, the code for **0** . You are now at the end of the previous segment of the program and are ready to key in the sequence below.

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
D	14	
RCL 1	34 01	
STO 2	33 02	Reset savings and
RCL 4	34 04	dnpmt
STO 5	33 05	
0	00	Start month zero
STO 8	33 08	
LBL	23	
1	01	Return point
1	01	
STO	33	Add one to month
+	61	counter
8	08	
B	12	Calculate savings
A	11	Calculate dnpmt
g x>y	35 24	If savings less
GTO	22	than dnpmt return
1	01	to LBL 1, otherwise
RCL 8	34 08	recall month
RTN	24	

To operate:

4. **Switch** to **RUN.**
5. Follow the instructions below.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
	To see month at which savings			
	overtake dnpmt, press		D	11.00

Programming Label E: Difference Between Savings and Downpayment at a Specified Month.

Here again, a specific answer is required—the difference between downpayment and savings at a specified month. We will use DSZ to cycle through the requisite number of times.

1. **Switch** to **W/PRGM.**
2. See 24, the code for RTN at the end of LBL D. You are ready to key in the sequence below:

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
E	15	
STO 8	33 08	Store desired month
RCL 1	34 01	Reset savings and
STO 2	33 02	dnpmt
RCL 4	34 04	
STO 5	33 05	
LBL	23	Return point
2	02	
B	12	Calculate savings
A	11	Calculate dnpmt
9	35	Repeat until
DSZ	83	contents of register
GTO	22	8 is zero
2	02	Then calculate
−	51	difference
RTN	24	

R_8 Savings − Dnpmt = Difference

To operate:

3. **Switch** to **RUN.**
4. Follow the instruction below.

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
		8	E	−464.27
		3	E	−1439.13
			ETC.	ETC.

HP-65 Program Form

SWITCH TO W/PRGM PRESS f PRGM TO CLEAR MEMORY

Page 1 of 1

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	
A	11	
RCL 2	34 02	Dnpmt
RCL 3	34 03	Inflation factor
x	71	
STO 2	33 02	New dnpmt
RTN	24	
LBL	23	
B	12	
RCL 5	34 05	Savings
RCL 7	34 07	Interest factor
x	71	
RCL	34 06	monthly savings
+	61	
STO 5	33 05	New savings
RTN	24	
LBL	23	
C	13	
RCL 1	34 01	Reset savings and
STO 2	33 02	dnpmt
RCL 4	34 04	to initial
STO 5	33 05	conditions
LBL	23	
0	00	Return point
B	12	Calculate savings
A	11	Calculate dnpmt
−	51	difference
R/S	84	
GTO	22	
0	00	
LBL	23	
D	14	
RCL 1	34 01	
STO 2	33 02	Reset savings
RCL 4	34 04	and dnpmt
STO 5	33 05	
0	00	Start month zero
STO 8	33 08	
LBL	23	
1	01	Return point
1	01	Add one to month
STO	33	counter
+	61	
8	08	
B	12	Calculate savings
A	11	Calculate dnpmt
g x>y	35 24	If savings less than
GTO	22	dnpmt return to
1	01	LBL 1, otherwise
RCL 8	34 08	recall month

KEY ENTRY	CODE SHOWN	COMMENTS
RTN	24	
LBL	23	
E	15	
STO 8	33 08	Store desired month
RCL 1	34 01	
STO 2	33 02	Reset savings and
RCL 4	34 04	dnpmt
STO 5	33 05	
LBL	23	Return point
2	02	
B	12	Calculate savings
A	11	Calculate dnpmt
g	35	Repeat until
DSZ	83	contents of register
GTO	22	8 is zero
2	02	then calculate
−	51	difference
RTN	24	

REGISTERS
R_1 Initial dnpmt
R_2 Dnpmt any month
R_3 Inflation factor
R_4 Initial savings
R_5 Savings any month
R_6 Monthly addition to savings
R_7 monthly interest factor
R_8 month counter
R_9

LABELS	
A	
B	
C	
D	
E	
0	used
1	used
2	used
3	
4	
5	
6	
7	
8	
9	

FLAGS	
1	
2	

TO RECORD PROGRAM INSERT MAGNETIC CARD WITH SWITCH SET AT W/PRGM

HP-65 User Instructions

Page ____ of ____

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
1.	Load program				
2.	Key in initial dnpmt	dp_0	STO	1	dp_0
			STO	2	dp
3.	Key in inflation factor (monthly)	infl. fact	STO	3	infl. fact
4.	Key in initial savings	sav_0	STO	4	sav_0
			STO	5	sav_0
5.	Key in monthly savings	m sav.	STO	6	m sav.
6.	Key in interest factor (monthly)	int. fact	STO	7	int. fact
7.	To see dnpmt for month 1, 2, 3, etc.				
	repeatedly press		A		dp 1, 2,...
8.	To see savings for month 1, 2, 3, etc.				
	repeatedly press		B		sav 1, 2, ...
9.	To see difference for month 1, press		C		$sav_1 - dp_1$
	to get months 2, 3, etc. repeatedly press		R/S		diff 2, 3..
10.	To see month at which savings				
	overtake dnpmt, press		D		mo (diff+)
11.	To see difference at any month, enter	mo.	E		diff

The entire program requires 68 memory locations as shown on the following filled-out program form.

To save the program, just pass a blank card through the calculator, switched to W/PRGM. Write the new definitions of keys **A**, . . . , **E** on the magnetic card together with the program title. Then fill out the HP-65 User Instruction Form to remind you how to run the program at a later time. Finally, if you wish, you can write your instructions on the HP-65 Pocket Instruction Card. You may carry the magnetic program card in it.

What we have presented is one possible way to write a program which gives us the required answers. Doubtless you can think of other ways. You may wish to build your variable entry capability into the program instead of storing them manually. You may wish to have the labels calculate differently than we did. Perhaps you can find a way to show what month you are calculating for in labels **A** and **B**. We urge you to try your own way.

Miscellaneous Program Topics

Program Debugging

Including Temporary Stops. Where space permits it is advisable to include additional **R/S** operations in long programs to display intermediate results while writing and checking the program. When the program is finally checked out, the unwanted stops can be deleted.

Single Stepping. When switched to RUN mode, **SST** executes the program, one step at a time. You can observe the effect of your program in slow motion. To single step through a function, first press **GTO** followed by **A**, . . . , **E** or [0], . . . , [9], then press **SST** , repeatedly. However, the **RTN** at the end of the function will be ignored.

Numerical Examples. It is usually absolutely necessary to work out a sample problem by independent means and then to do the same calculation using your program.

Magnetic Cards

Prerecorded Cards. Now that you have seen how programs work you can understand that the answers to the following question depends on the program:

> "During a program stop, can I use the stack, the other registers, the flags, etc?"

It is safest to follow the printed procedures when using the prerecorded programs. Departures should not be made without studying the program listing in detail.

Read/Write Operations. Reading or writing a card does not change the contents of the registers. A program can utilize data developed by a prior program. Whenever a magnetic card is written or read, all 100 steps are transferred. 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.

Identifying Stops

Blinking Display Stops. Errors that cause a blinking zero display, if executed in a program, also stop the program. You can identify the stop by switching momentarily to W/PRGM to see the code of the offending operation.

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

Accidental Stops. Remember, that pressing any key will stop a program. Be careful to avoid pressure on the keyboard during program operation.

Cued Stops. If memory space 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 8 stops for input, it is very helpful to have the numbers 1, . . . , 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 will be lifted.*)

Concerning W/PRGM Mode Display

Another feature of the W/PRGM display is that it allows you to see the last key pressed in a series of manual operations (*except program operations*). **For example,** in RUN mode you intend to key in

4.032 **+**

when the phone rings. After talking on the phone you can't remember whether or not you pressed the **+** key. Switch to W/PRGM, you will see 61 if you pressed **+**.

If you have been calculating manually, and then wish to display a program, pressing **SST** will resume the program memory display.

Programming Is a Creative Process

As you may have observed, programming the HP-65 is surprisingly easy to learn. Once you learn the operations, and follow the few rules, a whole world of possibilities is open to you. We have purposely presented the calculator with minimal use of formulas in case some of your mathematical training has dimmed by lack of use. You can adapt what you have learned here to your own purposes. In case you write a program that exceeds memory, remember that most programs can be shortened upon reexamination. Impromptu programs prepared for special purposes are usually faster to program and debug. Larger general programs can take considerable persistence. Good luck!

Appendix A

Operating Limits

Accuracy

The accuracy of the HP-65 depends upon the operation being performed. Also, in the case of transcendental functions, it is impractical to predict the performance for all arguments alike. Thus, the accuracy statement is not to be interpreted strictly, but rather as a general guide to the calculator's performance. The accuracy limits are presented here as a guide which, to the best of our knowledge, defines the maximum error for the respective functions.

The elementary operations [1/x], [f] [√x], [f⁻¹] [→D.MS] have a maximum error of ±1 count in the 10th (least significant) digit. Errors in these elementary operations are caused by rounding answers to the 10th digit.

An example of roundoff error is seen when evaluating $(\sqrt{5})^2$. Rounding $\sqrt{5}$ to 10 significant digits gives 2.236067977. Squaring this number gives the 19-digit product 4.999999997764872-529. Rounding the square to 10 digits gives 4.999999998. If the next largest approximation (*2.236067978*) is squared, the result is 5.000000002237008484. Rounding this number to 10 significant digits gives 5.000000002. *There simply is no 10-digit number whose square when rounded to 10-digits is 5.000000000.*

When subtracting numbers having 10 significant digits, the answer is correct to ±1 count in the 10th (least significant) digit of the algebraically larger operand.

Factorial function ([g] [n!]) is accurate to ±1 count in the ninth digit. Values converted to degrees-minutes-seconds [f] [→D.MS] are correct to ±1 second, as are the results of [f] [D.MS+] and [f⁻¹] [D.MS+].

The accuracy of the remaining operations (*trigonometric, logarithmic, and exponential*) depends upon the argument. The answer that is displayed will be the correct answer for an input argument having a value that is within ±N counts (*see table below*) in the 10th (*least significant*) digit of the **actual** input argument. **For example,** 1.609437912 is given as the natural log of 5 when calculated on the HP-65. However, this is an approximation because the result displayed (1.609437912) is actually the natural log of a number **between** 4.999999998 and 5.000000002, which is ±2 counts (*N=2 for logarithms*) in the 10th (*least significant*) digit of the **actual** input argument.

OPERATION	VALUES FOR N
f LOG, f LN, f^{-1} LN	2*
trigonometric	3**
g y^x	4 for y, and 10 for x
f^{-1} LOG	7
f R→P, f^{-1} R→P	4

*An additional error in the 10th least significant digit of the *displayed result* is ± 1 count for f LN and ± 3 counts for f LOG.

**Trigonometric operations have an additional accuracy limitation of $\pm 1 \times 10^{-9}$ in the *displayed answer*.

Calculating Range

To ensure greater accuracy, the HP-65 performs all calculations by using a 10-digit number and a power of 10. This abbreviated form of expressing number is called **scientific notation**; i.e., $23712.45 = 2.371245 \times 10^4$ in scientific notation.

Underflow

If a result develops 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.

Overflow

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

Temperature Range

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

Appendix B

Accessories

To order additional standard or optional accessories for your HP-65, fill out the Accessory Order Form in the Important Information Envelope and return it with check, money order, or company purchase order to:

HEWLETT-PACKARD, Advanced Products Division
19310-19320 Pruneridge Avenue, Cupertino, Calif. 95014

If outside the U.S., please contact the Hewlett-Packard Sales Office nearest you.

Standard Accessories

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

Accessory	*Model/Part No.*
Battery Pack	**82001A**
Battery Charger (115/230 Vac)	**82002A**
Travel Safety Case	**82018A**
Soft Case	**82017A**
HP-65 Owner's Handbook	**00065-90200**
Personalized Labels (4)	**7120-2946**
HP-65 Quick Reference Guide	**00065-90203**
Standard Application Pac including:	***00065-67008***
■ *Instruction Book*	
■ *Blank Pocket Instruction Cards (20)*	
■ *Prerecorded Magnetic Cards (19)*	
■ *Head Cleaning Card*	
■ *Blank Magnetic Cards (20)*	
Programming Worksheet Pad	**9320-0616**

Optional Accessories

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

Accessory	*Model/Part No.*
Battery Holder and Pack	**82004A**
Security Cradle	**82015A**
Field Case	**82016A**
Blank Cards (40)	**00065-67010**
Programming Worksheet Pad	**9320-0616**
Blank Pocket Instruction Cards (20)	**9320-0613**

The HP 82004A Battery Holder and 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).

Appendix C

Service and Maintenance

Calculator Checkout Procedure

> **Note**
> Charge battery pack before portable use.

A rechargeable battery pack is provided with your calculator. The calculator will operate while charging; however, be sure to fully charge the battery pack for 14 hours before portable use. Charge in either **ON** or **OFF** position.

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

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** then either:

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

Blank Display

If the display blanks out, turn the HP-65 **off,** place the W/PRGM-RUN switch in **RUN** position, and turn the HP-65 back **on.** If 0.00 does not appear on the display, check the following:

1. Examine 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 warranty information below).

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, shipping 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.

Out of Warranty

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 booklet*).

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 booklet*).

Send items to be returned to the address nearest you shown on the Service Card, after packaging them safely. Under normal conditions, your calculator will be repaired and shipped to you within 5 days of receipt at this address. Should other problems or questions arise regarding service, please call the applicable service telephone number on the Service Card, or call **Advanced Products Division, Customer Service Department,** at **(408) 996-0100.**

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. Shorter charge periods will reduce battery operating time.

If desired, the HP-65 can be operated continuously from the ac line. The battery pack is in no danger of becoming overcharged. Since you cannot operate the card reader/writer without a charged battery, even with the charger plugged in, the battery should not be removed while running from the ac line. 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. Turn 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. Slide the power switch to **ON.** If the W/PRGM-RUN switch is set to **RUN,** you should see a display of 0.00.
5. Slide 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 turned **OFF,** disconnect the battery charger from both the power receptacle and the HP-65.

CAUTION

The use of a charger other than the HP 82002A Battery Charger supplied with the calculator may result in damage to your calculator.

Battery Operation

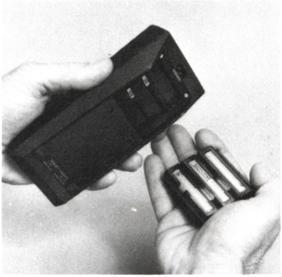
Use only the HP 82001A Battery Pack. A fully charged battery pack provides approximately three hours of continuous operation. By turning the power **off** when the calculator is not in use, the HP-65's battery pack should easily last throughout a normal working day.

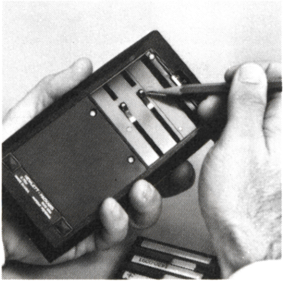
All decimal points in the display light when 2 to 5 minutes of operation time remain in the battery pack. Even when all the decimal points are lit, the true decimal position is known because an entire digit position is allocated to it.

Battery Pack Replacement

To replace your battery pack use the following procedure:

1. Turn 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.

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

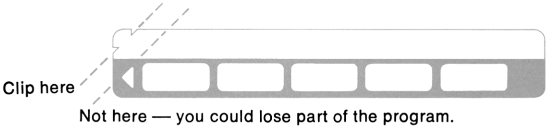
Temporary degradation, peculiar to nickel-cadmium batteries, may cause a decrease in the operating period of the battery pack. Should this happen, turn the HP-65 on for at least 5 hours to completely discharge the battery pack. Then, put it on charge for at least 14 hours. This procedure should correct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

If the battery pack 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 previously discussed. If the battery pack is **out** of warranty, use the Accessory Order Form provided with your HP-65 to order a replacement.

Magnetic Cards

Protecting a Card

To protect a card containing a stored program, clip the notch already there with scissors as shown below.



Care and 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/or with a soft damp 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, 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 writing programs, by all means use the cleaning card; that's what it is for. However, if one to five passes of the cleaning card does not clear up the situation, send your calculator in for servicing.

Annotating 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 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.

Using Alternate Track

It is possible to store a program on the opposite edge of a card (*and to later read it*) by inserting the other end (*opposite to the arrowhead*), face up. Thus, a card can hold two 100-step programs. However, we recommended that you use only one track since:

1. Second program cannot easily be labelled.
2. Extreme care must be taken in protecting 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, it may not read properly.

Improper Card Reader/Writer Operation

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

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 still not read or write program cards, the trouble may be due to dirty record/playback heads. Use the head-cleaning card once 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.

CAUTION

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

Appendix D

Common Errors

1. Having unwanted duplicate names (*labels*) for user defined functions in memory 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 W/PRGM-RUN switch is set to **RUN.**
3. Inadvertently **erasing** a program on a magnetic card by inserting an unprotected magnetic card when W/PRGM-RUN switch is set to **W/PRGM.**
4. Keying unwanted operations into program memory because the W/PRGM-RUN switch is set to **W/PRGM** when keys are pressed.
5. **Failing** to shift **up** to a gold function (**f** or **f⁻¹**) or **down** to a blue function (**g**) because prefix key was omitted.
6. **Losing** the T-register contents because the entry of a new number or the recall of a register lifts the stack.
7. **Destroying** the contents of register R_9 because a trigonometric function, a relational test, or the rectangular-polar conversion was done.
8. **Failing** to take account of a merged code and to provide a NOP as filler in a two-step skip.
9. Performing a trigonometric function in the **wrong** angular mode.
10. **Mistakenly** trying to call user defined functions labelled 0, . . . , 9. Only **A**, . . . , **E** can be used to label or call defined functions.
11. **Forgetting** to **clear** flags before using them.
12. Expecting LST X, stack, or registers to remain unchanged after calling a user defined function from the keyboard *(letters* **A** *, . . . ,* **E** *) or within a program.*

13. Using DSZ in a program and **forgetting** to initialize register R_8 to the proper value.

14. Calculating f(x,y) with the operands **reversed.**

15. **Losing** program and data by inadvertently switching the calculator **off,** by **unplugging** the battery charger, or by **plugging** in the battery charger.

16. Causing **improper** return from a defined function because multi-level nesting was attempted. Proper usage is as follows: a program can **call** a function but that function must **return** control to the caller **before** another function can be executed. See description of RTN in section 4.

17. Using CLX to put zero into **X** only to have it **overwritten** by a subsequent operation, because CLX disables the stack lift. Use 0 to put zero into **X.**

Index

A

B

C

E

F

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

HEWLETT PACKARD

Sales, service and support in 172 centers in 65 countries
19310-19320 Pruneridge Ave., 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

Printed in U.S.A.