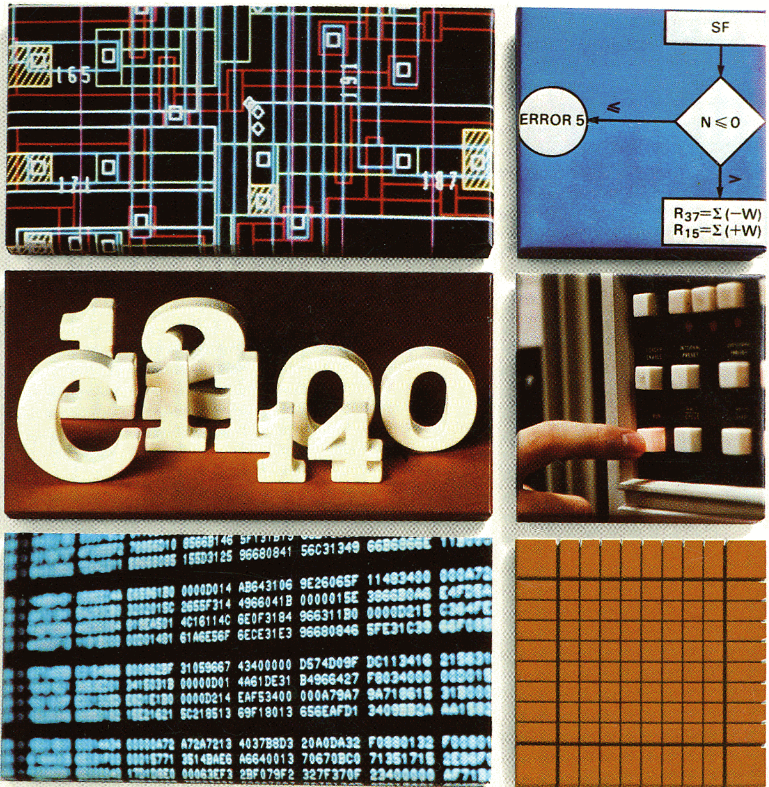
HEWLETT-PACKARD

HP-16C Computer Scientist

OWNER'S HANDBOOK



HEWLETT
PACKARD

HP-16C
Computer Scientist
Owner's Handbook

April 1982

00016-90001

Introduction

Welcome to the world of the Hewlett-Packard Computer Scientist! You're in good company with HP—the calculator of choice for astronauts in the space shuttle, climbers on Mt. Everest, yachtsmen in the America's Cup, and engineers, scientists, and students the world over.

The HP-16C is a versatile and unique calculator, especially designed for the many professionals and students who work with computers and microprocessors—whether as programmers or designers. The HP-16C specialized design provides:

- Integer arithmetic in four number bases (hexadecimal, decimal, octal, and binary), operating in 1's or 2's Complement or Unsigned mode.
- A variable word size, selected by the user, up to a maximum of 64 bits.
- Logical operators and bit manipulations.
- 203 bytes of user memory, providing up to 203 program lines.
- Floating-point decimal arithmetic.
- Continuous Memory, retaining data and program instructions indefinitely.
- Extremely low power consumption and long battery life.

This handbook is written with the professional in mind—someone already familiar with the principles of computer organization and binary operations. The handbook accommodates a wide range of expertise, however. For a quick overview of and reference to the calculator's operations, use the Function Summary and Index—the blue-edged pages just in front of the Subject Index.

Part I of the handbook, HP-16C Fundamentals, covers the specific operations of the HP-16C, as well as its RPN (Reverse Polish Notation) logic system. Part II, HP-16C Programming, is dedicated to keystroke programming methods and capabilities. Each programming section is structured to give first a general explanation of operations, then examples, then a more detailed look at certain features. This makes it easy for you to get a quick picture of how the system operates, if this is all you need.

The functions discussed in sections 1 and 2 of part I and in part II (HP-16C Programming) are similar to those used in certain other HP calculators, while the features unique to the HP-16C are concentrated in sections 3 through 6.

Before starting these sections, you can get a brief introduction to the capabilities of the HP-16C by working through the HP-16C: A Quick Look, starting on page 10.

Finally, the appendices include details on error and flag conditions, lists of operations with special characteristics, and warranty and service information. The Function Summary and Index at the back of the handbook provides short descriptions of every key's function, with page references to more comprehensive material within the handbook. It makes it easy to get the most from your HP-16C!

Contents

The HP-16C: A Quick Look

The HP-16C Computer Scientist is a powerful problem solver, operating in either Integer mode or Floating-Point Decimal mode. In Integer mode you can perform integer binary arithmetic, number base conversion, bit manipulations, and logical operations. In Floating-Point Decimal mode you can work out extensive floating-point calculations. Programming can be done in both modes. The HP-16C Continuous Memory retains data and program instructions indefinitely until you choose to reset it.

An important feature of the HP-16C is its extremely low power consumption. This efficiency eliminates the need for a recharger and provides a lightweight, compact design. Power consumption in the HP-16C is so low that the average battery life in normal use is 6 to 12 months. In addition, the low-power indicator gives you plenty of warning before the calculator stops functioning.

The HP-16C also conserves power by automatically shutting its display off if it is left inactive for a few minutes.

Keyboard Operation

Your Hewlett-Packard calculator uses Reverse Polish Notation (RPN), an operating logic that involves the use of the [ENTER] key. The use of [ENTER] eliminates the need for parentheses in calculations; instead calculations are performed using a memory stack. For example, let's look at the arithmetic functions.

With the calculator on (press [ON] if necessary), select a number base (hexadecimal, decimal, octal, or binary) in Integer mode by pressing the key marked [HEX], [DEC], [OCT], or [BIN]. This establishes the *number base mode* for the display, and is indicated by the presence of an **h**, **o**, **d**, or **b** at the right of the display. The default mode (at initial turn-on or Continuous Memory reset) is Hexadecimal (Integer). You can clear the display to zero by pressing [g] [CLx] (a blue-printed function).*

* If you have not used an HP calculator before, notice that most keys have three labels. To use the main function—the one printed in white on top of the key—just press that key. For those printed in gold or blue, press the gold [f] key or the blue [g] key first.

To perform arithmetic, key in the two operands—separated by [ENTER]—and then the operator. The function is executed when its key is pressed, and the result immediately appears. If you enter a digit incorrectly, press [BSP] to undo the mistake, then key in the correct digit.

Integer Calculations

When you set one of the number base modes, the calculator operates in Integer mode.

To Compute*	Press	Display
(in base 2)	[g] [CLx] [BIN]	**0 b**
1111 − 1	1111 [ENTER] 1 [−]	**1110 b**
1111 × 11	1111 [ENTER] 11 [×]	**101101 b**

You can also calculate using a value already in the display:

To Compute	Press	Display
101101 ÷ 10	10 [÷]	**10110 b**
10110 AND 1111	1111 [f] [AND]	**110 b**

(The [÷] operation results in the display of a **C** annunciator, signifying that the *carry* flag has been set. Flags are explained on page 36. Press [g] [CF] 4 to clear the flag and the annunciator.)

Notice that in the four examples:

- Both numbers are keyed in before you press the operator key.
- [ENTER] is used only to separate two numbers that are *keyed in* one after the other.
- Pressing a function key—in this case [−], [×], [÷], or [AND]—executes the function immediately and displays the results.

Floating-Point Calculations

The HP-16C can perform floating-point decimal arithmetic when it is in Floating-Point Decimal mode. The [FLOAT] function will convert the calculator from Integer mode into Floating Point mode and display the specified number of decimal places.

*The calculator should display **2–16–0000** when [f] [STATUS] is pressed. If it does not, refer to page 37.

To Compute	Keystrokes	Display
(floating-point decimal)	[f] [FLOAT] 4 [BSP]	0.0000
$-4.9 \div 6$	4.9 [CHS] [ENTER] 6 [÷]	-0.8167
$\sqrt{60}$	60 [g] [√x]	7.7460

Programmed Solutions

Writing a Program. The HP-16C is keystroke-programmable: you can program it simply by recording the same keystrokes you use to evaluate a problem manually.

Example: Write an iterative program that adds 1 continually to a given number.

Keystrokes	Display*	
[g] [P/R]	000-	Sets calculator to Program mode (**PRGM** annunciator on). Line 000.
[f] CLEAR [PRGM]	000-	Clears program memory.
[g] [LBL] A	001-43,22, A	Assigns this program label "A".
1	002- 1	Line 002: 1.
[+]	003- 40	Line 003: adds 1 to whatever is in display when program is run.
[f] SHOW [BIN]	004- 42 26	Momentarily pauses and displays binary result.
[GTO] A	005- 22 A	Continues execution in a loop.

*The display includes line numbers and keycodes. Keycodes are two-digit numbers that indicate the row and column position of the key(s) pressed.

Keystrokes	Display	
g P/R		Returns calculator to Run mode; no **PRGM** annunciator. Display will show the result of the last calculation performed.

Running the Program. Key the starting number (for example, zero) into the display. You do not need to use ENTER since starting the program will separate the two numbers to be added. The program above adds 1 to whatever number you key in.

Keystrokes	Display	
DEC		Converts to Integer mode, base 10. (You can start in any number base; the program will display the numbers in binary.)
16 f WSIZE		Sets word size to 16.
0	**0 d**	Initial number: 0.
GSB A	**1 b** **10 b** **11 b** **100 b** ⋮	Addresses and starts a program with label "A". The momentary displays are binary.
R/S	**22 d**	Since this is an endless loop, stop program execution with R/S (*run/stop*). The display shows the decimal equivalent of the binary value at the particular moment you press R/S.

This introduction to the HP-16C should give you a feel for its operation. It is only a glimpse, however; for a look at the dozens of other powerful HP-16C functions, turn the page and explore Part I, HP-16C Fundamentals.

Part I
HP-16C Fundamentals

Section 1

Getting Started

This section provides a detailed orientation to general use of the HP-16C: digit entry, display clearing, the use of [ENTER] and RPN (Reverse Polish Notation), and Continuous Memory. Although the examples use Integer mode, all features operate identically in Integer and Floating-Point Decimal modes unless otherwise indicated. This material has been written primarily for those unfamiliar with these features of current Hewlett-Packard calculators.

Power On and Off

The [ON] key turns the HP-16C on and off.* To conserve power, the calculator automatically turns itself off after a few minutes of inactivity.

Keyboard Operation

Primary and Alternate Functions

Most keys on your HP-16C perform one primary and two alternate ("shifted") functions.† To select the primary function printed on the face of a key, press only that key. For example: [÷]. To select the alternate function printed in gold above the key or in blue below the key, press the like-colored prefix key ([f] or [g]) followed by the function key. For example: [f] [XOR] and [g] [DBL÷].

* Note the [ON] key is lower than the other keys to prevent its being inadvertently pressed.

† Throughout this handbook, we will observe certain conventions in referring to functions. References to the *function itself* will appear as just the function name in a box, such as "the [MEM] function." References to *using the key* will include the prefix key, such as "press [g] [MEM]." References to the functions printed in gold under the brackets labeled "CLEAR," "SET COMPL," or "SHOW" will be preceded by the word "CLEAR," "SET COMPL," or "SHOW," such as "the CLEAR [REG] function" or "press [f] SHOW [DEC]."

When a prefix key can be followed by any of several keys, the reference will specify the possible keys in braces. For example, "press [f] [WINDOW] {0 to 7}".

Notice that when you press the [f] or [g] prefix key, an **f** or **g** annunciator appears and remains in the display until another key is pressed.

Clearing Prefix Keys

A prefix key is any key that must be followed by one or more additional keys to complete the key sequence for a function. There is a list of all prefix keys in appendix B.

When any prefix key (such as [STO] or [f]) has been pressed, pressing [f] CLEAR [PREFIX] will clear that prefix key, leaving the calculator ready for a new keystroke. If you have mistakenly pressed [f] instead of [g] or vice-versa, you can correct it merely by pressing the other key.

The "CLEAR" Keys

The "CLEAR" operations are listed below. *Clearing* a register means to replace its contents with zero.

Clearing Sequence	Effect
[f] CLEAR [PRGM]	
In Run mode:	Repositions program memory to line 000.
In Program mode:	Clears entire program memory.
[f] CLEAR [REG]	Clears all data storage registers.
[f] CLEAR [PREFIX]	Cancels any prefix from a partially entered key sequence.

Display Clearing: [CLx] and [BSP]

The HP-16C has two types of display clearing operations: [CLx] (*clear X*) and [BSP] (*back space*).

In Run mode:

- [CLx] clears the display to zero.
- [BSP] deletes only the last digit in the display if digit entry has not been terminated. ([ENTER] and most other functions terminate digit entry so that the next digit keyed in becomes part of a new number.) You can then key in (a) new digit(s) to replace the one(s) deleted. If digit entry *has* been terminated, then [BSP] acts like [CLx].

Keystrokes	Display	
[HEX]		Hex mode. Display shows last result.*
1234	**1234 h**	Digit entry not terminated.
[BSP]	**123 h**	Clears only the last digit.
1	**1231 h**	
[ENTER]	**1231 h**	Terminates digit entry.
[BSP]	**0 h**	Clears all digits to zero.
12	**12 h**	
[g] [CLx]	**0 h**	Clears display whether or not digit entry has been terminated.

In Program mode:

- [CLx] is programmable: it is *stored* as a programmed instruction, and will not delete the currently displayed instruction.
- [BSP] is *not* programmable. It is used instead to delete program instructions.

One-Number Functions

A one-number function performs an operation using only the number in the display (X-register). To use any one-number function, press the function key *after* the number has been placed in the display.

Keystrokes	Display
0	**0 h**
[f] [NOT]	**FFFF h**

Two-Number Functions and [ENTER]

A two-number function must have two numbers present in the calculator before executing the function. [+], [-], [×], and [÷] are examples of two-number functions.

*The calculator should display **2-16-0000** when [f] [STATUS] is pressed. If it does not, refer to page 37.

Terminating Digit Entry. When *keying in* two numbers to perform an operation, the calculator needs a signal that *digit entry has been terminated* for the first number. This is done by pressing [ENTER] to separate the two numbers. If, however, one of the numbers is already in the calculator as the result of a previous operation, you do not need to use the [ENTER] key. All functions except the digit entry keys themselves have the effect of *terminating digit entry*.*

Chain Calculations. Long calculations do not require the use of parentheses. [ENTER] is used to separate two numbers sequentially keyed into the stack.

Example: Calculate $(6 + 7) \times (9 - 3)$ in base 10.

Keystrokes	Display	
[DEC]		Decimal mode. Display shows last result.
6 [ENTER]	**6 d**	Digit entry terminated.
7 [+]	**13 d**	The number 13 is stored as an intermediate result.
9 [ENTER]	**9 d**	
3 [−]	**6 d**	Six is also stored as an intermediate result.
[×]	**78 d**	$(13 \times 6) = 78$.

Continuous Memory

What Is Retained

The Continuous Memory feature of the HP-16C retains the following information, even while the calculator is off:

- Number base or operating mode (Hexadecimal, Decimal, Octal, Binary, or Floating-Point Decimal).
- Arithmetic mode (1's Complement, 2's Complement, Unsigned).
- Word size.

*The digit entry keys are the digit keys and [BSP]. Also—in Floating-Point Decimal mode only—[·], [EEX], and [CHS].

- All stored numbers.
- All stored programs.
- Position of the calculator in program memory.
- Any pending subroutine returns.
- Flag settings.
- Scrolling of the display.
- Type of digit separators.

When the calculator is turned on, it always "wakes up" in Run (not Program) mode.

Continuous Memory can be preserved for a short period while the batteries are removed. (The calculator must be off.) Refer to appendix C for instructions on changing batteries.

Resetting Continuous Memory

To reset (entirely clear) Continuous Memory:*

1. Turn the calculator off.
2. Press and hold ON, then press and hold −.
3. Release ON, then −. (Steps 2 and 3 are represented in this manual as ON/−.)

Error Display. When Continuous Memory is reset, **Pr Error** (*power error*) is displayed. Press any one key to clear the display. Appendix A contains a list of error messages and the conditions that cause them.

Default Conditions. When the calculator is *initially* turned on or Continuous Memory is reset, the following conditions are set by default:

- Number base: Hexadecimal (Integer mode).
- 2's Complement mode.
- Word size: 16 bits.
- All flags cleared.
- Program memory and all registers cleared.

*If the calculator is dropped or otherwise mishandled, Continuous Memory may be reset.

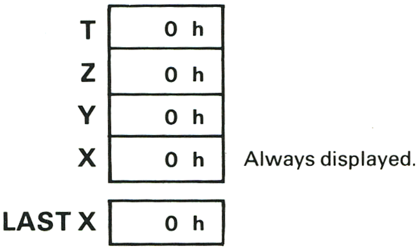
Section 2

The Automatic Memory Stack

The Memory Stack and Stack Manipulation

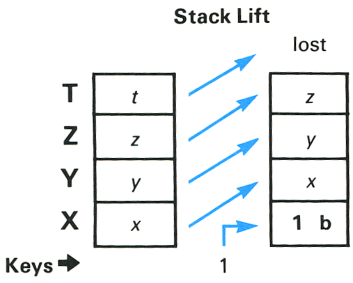
The HP-16C uses Reverse Polish Notation (RPN) to solve complicated calculations without parentheses and with a minimum of keystrokes. Using the memory stack and the ENTER key, the calculator automatically retains and returns intermediate results. This section discusses the operation of the calculator stack, which is fundamental to the use of the HP-16C in all modes, including programming.

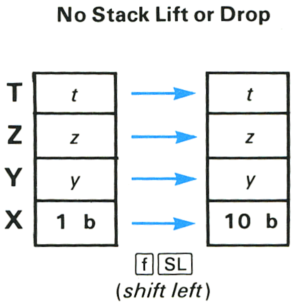
The Automatic Stack Registers

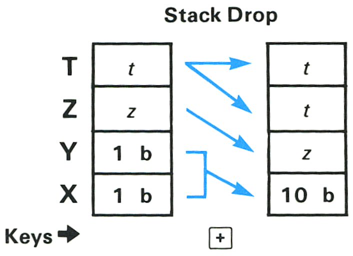


The number that appears in the display is the number in the X-register—unless the calculator is in Program mode (**PRGM** annunciator displayed).

Numbers in the stack are stored on a last-in, first-out basis. The three stacks drawn below illustrate the three types of stack movement. Assume that *x, y, z,* and *t* represent any numbers which may be in the stack, and that the calculator is in Binary mode.





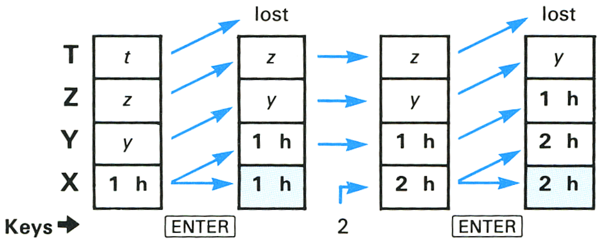


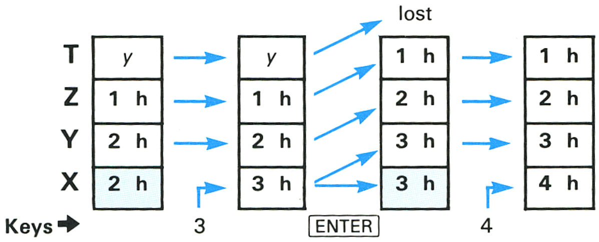
Typically, one-number functions (as defined in the previous section) result in no stack movement, while two-number functions *usually* result in a stack drop.

Notice the number in the T-register is regenerated when the stack drops, allowing this number to be used repetitively as an automatic constant.

Stack Manipulation Functions

Pressing ENTER separates two numbers keyed in one after the other. It does so by lifting the stack and copying the number in the display (X-register) into the Y-register. The number entered next then writes over the value in the X-register; there is no stack lift. The example below shows what happens as the stack is filled with the hexadecimal numbers 1, 2, 3, 4. (The shading indicates that the contents of that register will be written over when the next number is keyed in or recalled.)





In addition to ENTER, there are three other functions that rearrange the stack:

- R↓ (*roll down*) rolls the contents of the stack registers down one register. The number in the X-register rolls around into the T-register.
- R↑ (*roll up*) rolls the stack contents up one register. The T-register contents roll around into the X-register.
- x⋛y (*X exchange Y*) exchanges the numbers in the X- and Y-registers.

The LAST X Register

The LAST X register, another memory register, preserves the number that was last in the display before execution of a numeric operation.* Pressing g LSTx (*LAST X*) places a copy of the contents of the LAST X register into the display (X-register).

*For a complete list of operations which save *x* in the LAST X register, refer to appendix B.

The [LSTx] feature allows you to reuse a constant value without re-entering it (as shown under Calculations with Constants, page 26). It can also assist you in error correction by recovering the number that was in the calculator before the last numeric operation.

For example, suppose you mistakenly entered the wrong addend (10 instead of 11) in a chain calculation:

Keystrokes	Display	
[BIN]		Binary mode. Display shows the last result.
1010 [ENTER]	1010 b	
10 [+]	1100 b	Oops! The wrong number was keyed in.
[g] [LSTx]	10 b	Retrieves from LAST X the last entry to the X-register (the incorrect addend) before [+] was executed.
[-]	1010 b	Reverses the function that produced the wrong answer.
11 [+]	1101 b	The correct answer.

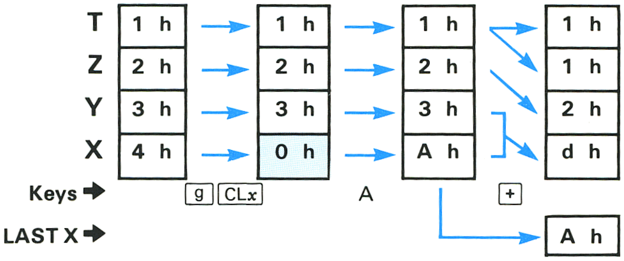
Numeric Functions and the Stack

Stack Movement

When you want to key in two numbers, one after the other, you must press [ENTER] between entries of the numbers. However, when you want to key in a number immediately following any function (including stack manipulations such as [R↓]), you do not need to use [ENTER]. Executing most HP-16C functions has this additional effect:

- The automatic memory stack is lift-*enabled*; that is, the stack will lift automatically when the *next* number is keyed in or recalled from storage into the display.
- Digit entry is terminated, so the next number starts a new entry.

There are two functions, [ENTER] and [CLx], that *disable* stack lift—that is, they do *not* provide for the lifting of the stack when the *next* number is keyed in or recalled.* Following the execution of either of these functions, a new number will simply write over the currently displayed number instead of causing the stack to lift. (Although the stack lifts when [ENTER] is pressed, it will *not* lift when the *next* number is keyed in or recalled.)



As you can see, when an arithmetic operation is performed with operands (A_6 and 3_{16}) in the X- and Y-registers, the stack drops and the result (D_{16}) is left in the X-register.

For a complete listing of how functions affect stack lift (enabling, disabling, and neutral) and digit entry, refer to appendix B.

Nested Calculations

The automatic stack lift and drop make it possible to do nested calculations without using parentheses. Intermediate results are automatically saved in the stack and used as needed. A nested calculation is solved simply as a series of one- and two-number operations. If you begin your calculation at the innermost number or pair of parentheses and work outward (as you would when working with pencil and paper), you will rarely need to store intermediate results in a storage register.

* [BSP] will also disable stack lift and clear the display (just as [CLx] does) if digit entry has been terminated. Otherwise, it is *neutral* to stack lift—that is, it neither enables nor disables stack lift.

For example, consider the (integer decimal) calculation

$$3[4+5(6+7)].$$

Keystrokes	Display	
DEC		Display shows last result.
6 ENTER 7 +	**13 d**	Intermediate result.
5 ×	**65 d**	Intermediate result.
4 +	**69 d**	Intermediate result.
3 ×	**207 d**	Final result.

This example shows that the stack automatically drops after each two-number calculation and lifts when a new number is subsequently keyed in.

Calculations With Constants

There are two ways (without using a storage register) to perform repeated calculations with a constant:

- Use the LAST X register.
- Load the stack with the constant prior to doing the computations.

Example: Remove the upper four bits and preserve the lower four bits from the following 8-bit binary numbers: 10001001, 10101111, and 11110101. The constant value 1111 will be used as a mask.

Using the LAST X Register. Make sure to calculate with the constant in the X-register (rather than the Y-register) so that it will always be saved in the LAST X register. Retrieve the constant by pressing g LSTx.

Keystrokes	Display	
BIN		Binary mode. Display shows previous rsult.
10001001 ENTER	**10001001 b**	First number.
1111	**1111 b**	The mask (the constant).
f AND	**1001 b**	Lower four bits.

Keystrokes	Display	
10101111	10101111 b	Second number.
[g] [LSTx]	1111 b	Retrieves the constant.
[f] [AND]	1111 b	Lower four bits.
11110101 [g] [LSTx]	1111 b	
[f] [AND]	101 b	Lower four bits.

Using the Stack. Load the stack with a constant by keying it in and pressing [ENTER] three times. After each operation (here: [AND]), the stack drops (making the constant available in the Y-register) and the constant value is regenerated in the T-register. By using [BSP] or [CLx] to disable the stack, the new, variable numbers entered will "write over" the previous result, thus preserving only the constant in the stack.

Keystrokes	Display	
1111 [ENTER]	1111 b	The mask (the constant).
[ENTER] [ENTER]	1111 b	Fills the stack with 1111.
10001001 [f] [AND]	1001 b	Lower four bits of first number.
[BSP]	0 b	Stack lift disabled.
10101111 [f] [AND]	1111 b	Lower four bits of second number.
[BSP]	0 b	Stack lift disabled.
11110101 [f] [AND]	101 b	Lower four bits of third number.

Section 3

Number and Display Control

Number representation in the HP-16C is much more versatile than in other calculators. This section will discuss the different aspects of *integer* number use and display: number bases, word size, complements, number ranges, and the resulting displays. (Floating-point format is described in section 5, Floating-Point Numbers.) The formats you specify are preserved by Continuous Memory.

Integer Mode

The number base modes (HEX, DEC, OCT, and BIN) operate strictly in Integer mode (that is, using integers only). Fractional decimal numbers can be used in Floating-Point Decimal mode, described in section 5. Pressing any of the four number base keys establishes Integer mode.

Number Base Modes

There are four number base modes used by the HP-16C in Integer mode for purposes of display and digit entry: Hexadecimal (base 16), Decimal (base 10), Octal (base 8), and Binary (base 2). An **h**, **d**, **o**, or **b** to the right of the eight-digit display indicates the present number base mode. The calculator defaults to Hexadecimal mode when first turned on or when Continuous Memory is reset.

Pressing HEX, DEC, OCT, or BIN converts the display to that number base in a right-justified, integer format. Digit keys pressed are interpreted accordingly: the calculator will not respond if you attempt to enter an inappropriate digit (such as a "3" in Binary mode). In addition to the digit keys 0 to 9, Hexadecimal mode uses the keys A to F, appearing in the display as **A**, **b**, **C**, **d**, **E**, and **F**.

Note: Regardless of the current number base mode, *the internal representation of numbers is always binary.* Switching between number modes changes the display only, not the calculator's internal representation of the value.*

Temporary Display ("SHOW")

To *temporarily* view the displayed value in another base, press [f] SHOW {[HEX], [DEC], [OCT], [BIN]}. The converted form of the number will be shown as long as you hold down the number base key.

Keystrokes	Display
[HEX] F	F h
[BIN]	1111 b
[f] SHOW [OCT] (hold)	17 o
(release)	1111 b

Complement Modes and Unsigned Mode

The HP-16C provides three conventions for representing numbers: *1's Complement* mode, *2's Complement* mode, and *Unsigned* mode. The 2's Complement mode is the default mode when the calculator is *first* turned on or after Continuous Memory is reset. Once a mode is set, it remains in effect until you change it or until Continuous Memory is reset. (All examples in this handbook use 2's Complement unless otherwise indicated.)

In the binary representation of a signed number, the leftmost or most significant bit with respect to word size serves as the sign bit: 0 for plus and 1 for minus. In Decimal mode, a negative number is displayed with a minus sign.

*The keystroke sequence on page 35 shows how number base mode, word size, and complement mode affect the display without affecting the calculator's internal binary representation of a number in Integer mode.

1's Complement Mode

Pressing [f] SET COMPL [1's] will set 1's Complement mode. When you press [CHS] (*change sign*) in 1's Complement mode, the 1's complement of the number in the X-register is formed by complementing all bits.

One's Complement accommodates an equal number of positive and negative numbers, but has two representations for zero: 0 and -0.

2's Complement Mode

Pressing [f] SET COMPL [2's] will set 2's Complement mode. The [CHS] function will take the 2's complement of the number in the display (that is, it complements all the bits in the X-register and adds 1).

In 2's Complement there is just one representation for zero, but there is always one more negative number than positive number represented.

Unsigned Mode

Pressing [f] SET COMPL [UNSGN] will set Unsigned mode, which uses no sign bit. The most significant bit adds magnitude, not sign, so the largest value respresented by an 8-bit word is 255_{10} instead of 127_{10}.

Changing signs in Unsigned mode has no meaning. If you press [CHS] in Unsigned mode, the result will be the 2's complement of the number in the X-register. Flag 5 (signified by the **G** annunciator) is set as a reminder that the true result is a negative number, which is outside the range of Unsigned mode.

The following table summarizes how the complement modes affect the decimal interpretation of all possible 4-bit patterns (word size 4).

Decimal Interpretation of 4-Bit Binary

Binary	1's Complement Mode	2's Complement Mode	Unsigned Mode
0111	7	7	7
0110	6	6	6
0101	5	5	5
0100	4	4	4
0011	3	3	3
0010	2	2	2
0001	1	1	1
0000	0	0	0
1111	−0	−1	15
1110	−1	−2	14
1101	−2	−3	13
1100	−3	−4	12
1011	−4	−5	11
1010	−5	−6	10
1001	−6	−7	9
1000	−7	−8	8

Word Size and Window Display

The HP-16C will work with *words* (data units) up to 64 bits long. The default *word size* when you first turn on the calculator or reset Continuous Memory is 16 bits. The display window shows eight digits at a time; leading zeros are not displayed.* A period is placed on the left and/or right side of the **h**, **d**, **o**, or **b** to indicate the presence of more, undisplayed digits to the left or right of the currently displayed portion of a number.

* Setting flag 3, as explained later in this section (page 36), will cause all leading zeros to be included in the display.

Word Size

To specify a word size, first place the desired word size (1_{10} to 64_{10}) in the X-register, then press [f] [WSIZE]. The absolute value of the number is used; a zero is interpreted as 64. After [WSIZE] is executed the stack drops.

A current word size smaller than 8 will limit the size of the number you can enter to stipulate a new word size; but you can always enter 0 [f] [WSIZE] to set a word size of 64. (You can then set any word size.) **Error 2** results if you attempt to specify a word size larger than 64.*

Keystrokes	Display	
[DEC] 16 [f] [WSIZE]		Base 10; word size 16.
[f] SET COMPL [2's]		Sets 2's Compl. mode.
32767 [ENTER]	**32767 d**	Largest positive 2's complement number with a word size of 16.
8 [f] [WSIZE]	**-1 d**	Number changes from $01111111\ 11111111_2$ (16 bits) to 11111111_2 (eight bits).
16 [f] [WSIZE]	**255 d**	Number changes from 11111111_2 to $00000000\ 11111111_2$.

Note: A change in word size might not preserve numerically equivalent values stored *in the memory stack.* Going to a smaller word size will truncate a word, leaving the least significant bits. Going to a larger word size will not preserve the sign bit of a negative number. If the original word size is restored, the original stack contents are not restored. (The effect on storage registers is different and is discussed on page 67.)

* It is possible (in 1's or 2's Complement mode) to obtain a negative number if you try entering a number larger than the largest positive number that can be represented within the current word size. This occurs when the most significant bit (the sign bit) becomes 1 (negative), as shown at the end of the keystroke sequence on page 36.

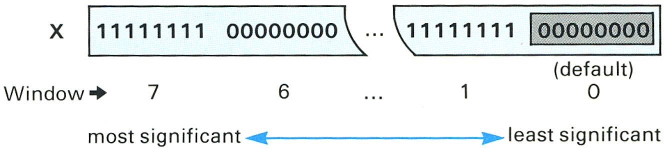
If the word size is 3 or less, attempting to initially enter a digit that is legal in the current number base mode *but is too large for the given word size* will result in the entry of a zero.

Windows

The display can be considered a *window* showing up to eight digits of the number in the X-register. The X-register—like all *registers*—can hold up to 64 binary digits, depending on the word size. What you normally see is window 0, the eight least significant digits of the number in the X-register. As you key in more than eight digits, the most significant digits move off the left end of the display and into window 1.

Pressing [f] [WINDOW] {0 to 7} will display different eight-digit portions of the word in the X-register. The display returns to window 0, the eight least significant digits of the word, with each new entry into the X-register. The highest window number is 7 since the maximum word size is 64. (With smaller word sizes or smaller numbers, the higher windows will be blank.) **Error 1** results if you specify a window number greater than 7.

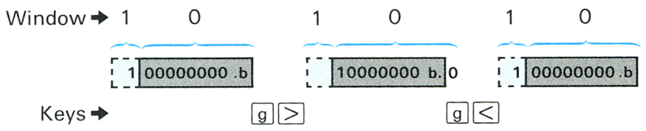
Example: The 16-digit hexadecimal value FF00 FF00 FF00 FF00 has a 64-digit binary representation (eight 1's alternating with eight 0's). In Binary mode, you can view the entire number by executing [f] [WINDOW] 0 through [f] [WINDOW] 7.



Scrolling

Scrolling with the [<] and [>] keys allows you to move different parts of a number into the display, one digit at a time. This does not change the number itself, only what part of the number you see.

The location of the period tells you where to look for the rest of the number in the X-register. For instance, if the period is on the *left* of the base indicator (.**b**), then there are more digits to the *left* of the current display. Pressing [g] [>] will scroll the number to the *right*, bringing these "hidden" digits into view. A period can appear both on the left and right sides of the base indicator if the current window is not at one end of the number.



Example: The following scrolling and WINDOW functions can be used to view the entire X-register contents. The word size used is 16 bits.

Keystrokes	Display	
BIN		Sets Binary mode.
11111111 ENTER	**11111111 b**	Display filled (eight digits).
1 +	**00000000 .b**	Period on left side, so number continues to the left.
g >	**10000000 b.**	Scrolls number one digit to right (period shows number now continues to the right).
f WINDOW 1	**1 b.**	The contents of window 1: the most significant digit.
f WINDOW 0	**00000000 .b**	Window 0: the least significant digits.

Scrolling is "reset"—that is, the display is reset to window 0—when a bit manipulation or mathematical function is executed. A complete list of functions that do *not* reset the display to window 0 is included in appendix B.

The Display and Internal Representation

The following keystrokes illustrate how various functions (number base, word size, complement mode) alter the calculator's display in relation to the internal binary representation.

Keystrokes	Display	Internal Binary Representation
HEX		
8 f WSIZE		
BSP	**0 h**	00000000
62	**62 h**	01100010
OCT	**142 o**	01100010
BIN	**1100010 b**	01100010
DEC	**98 d**	01100010
62	**62 d**	00111110
OCT	**76 o**	00111110
62	**62 o**	00110010
HEX	**32 h**	00110010
f SET COMPL 2's	**32 h**	00110010
CHS	**CE h**	11001110 Negative number in 2's Compl., word size 8.
OCT	**316 o**	11001110
BIN	**11001110 b**	11001110
DEC	**–50 d**	11001110
f SET COMPL 1's	**–49 d**	11001110 Internal representation does not change.
f SET COMPL UNSGN	**206 d**	11001110
f SET COMPL 1's	**–49 d**	11001110 In 1's Compl. this is interpreted as a negative number.
CHS	**49 d**	00110001

Keystrokes	Display	Internal Binary Representation
2	**2 d**	00000010
5	**25 d**	00011001
4	**-001 d**	11111110 (Corresponds to -1_{10} in 1's Compl. Zeros are placeholders for purposes of digit entry correction.)
f SET COMPL UNSGN	**254 d**	11111110

Flags

The HP-16C has three *user* flags (0, 1, and 2), which can be used to control program execution, and three *system* flags (3, 4, and 5), which are used to indicate system status.

The use of flags in programming is discussed in section 9, Program Branching and Controls.

Flags 3, 4, and 5 are simply status indicators and have no effect on calculator operation (unless you choose to use them to control program execution):

- Flag 3 controls the display of leading zeros. When it is set, zeros to the left of the highest nonzero digit are displayed. When it is clear (the default condition), the display of leading zeros is suppressed. (Note that leading zeros are always suppressed in Decimal and Floating-Point Decimal modes.)
- Flag 4 is set (and the **C** annunciator appears) when a **carry** has occurred.
- Flag 5 is set (and the **G** annunciator appears) when the returned value is **out-of-range** (Greater than the largest representable number or not representable in the current mode).

Section 4 includes a discussion of how the carry condition and out-of-range condition are generated.

All flags can be set, cleared, and tested as follows:*

- [g] [SF] *n* will *set flag* number *n* (0 to 5);
- [g] [CF] *n* will *clear flag* number *n*; and
- [g] [F?] *n* will check if flag *n* is set.

A flag's status and associated annunciator, if any, are retained until changed by:

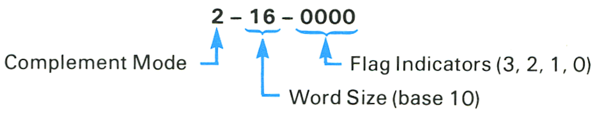
- Resetting Continuous Memory.
- Executing a function which affects that flag (flags 4 and 5 only).
- Clearing the flag with [CF] or setting it with [SF].

In programming, flags are generally used to record the result of a test for future use. Section 9 describes the use of flags in conditional branching.

Machine Status ([STATUS])

Pressing [f] [STATUS] will temporarily show (1) the current complement mode, (2) the current word size, and (3) which flags are set *other than* flags 4 and 5 (which display annunciators **C** and **G** when set). The display remains as long as you hold down the [STATUS] key. To alter machine status, refer to page 30 (complement modes), page 32 (word size), or page 36 (flags).

Initial [STATUS] Display †



* Press the decimal representation for the flag number. Note that the flag number does not enter the stack.

† When the calculator is first turned on or after Continuous Memory is reset.

Complement Status. The Complement mode is **0** (Unsigned), **1** (1's), or **2** (2's).

Flag Status. The display portion for flag status shows four places held by zeros or ones. The flags are numbered from the right from zero to three; a place occupied by **1** represents a set flag. For example, consider the following flag portions of STATUS displays:

# 3 2 1 0 **-0100**	Flag 2 set.	# 3 2 1 0 **-1101**	Flags 0, 2, and 3 set.

Special Displays

Annunciators

The HP-16C display contains six annunciators that indicate the status of the calculator for various operations. The meaning and use of these annunciators are discussed on the following pages:

*	Low-power indication, page 38.
f and **g**	Prefixes for alternate functions, page 17.
C	Flag 4 (carry) set, page 39.
G	Flag 5 (out-of-range) set, page 40.
PRGM	Program mode, page 72.

Error Display

If you attempt an improper operation—such as specifying a word size larger than 64—an error message will appear. For a complete listing of error messages and their causes, refer to appendix A.

To clear the **Error** display and restore the calculator to its prior condition, press any key. You can then resume normal operation.

Low-Power Indication

A flashing asterisk in the lower left-hand corner of the display indicates low battery power. At this point, however, you still have operating time remaining: at least 15 minutes if you run a program continuously, or at least an hour if you do calculations manually. (Certain batteries provide more time.) Refer to appendix C (page 102) for information on replacing the batteries.

Section 4

Arithmetic and Bit Manipulation Functions

Integer arithmetic operations and bit manipulation functions can only be performed in Integer mode. Since these functions are subject to carry and out-of-range conditions, an explanation of these conditions precedes the discussion of the functions themselves.*

Floating-point decimal arithmetic and other capabilities of Floating-Point Decimal mode are discussed in section 5, Floating-Point Numbers.

Carry and Out-of-Range Conditions

The execution of certain arithmetic and bit manipulation operations can result in a *carry* and/or an *out-of-range* condition. These conditions set flags (that may be tested) and display annunciators (for visual indication). The definitions for "carry" and "out-of-range" depend on the particular function executed.

Section 3, under Flags (page 36), explains how to manually set and clear these (and other) flags.

Flag 4: Carry (C)

The shifting, rotating, and arithmetic operations listed below will set *or* clear flag 4 and the **C** annunciator whenever they are performed in Integer mode. Flag 4, the carry flag, is set if the carry bit is 1, and cleared if the carry bit is 0.

SL	RL	RLn	+ (carry)
SR	RLC	RLCn	− (borrow)
ASR	RR	RRn	÷ (remainder $\neq 0$)
	RRC	RRCn	DBL÷ (remainder $\neq 0$)
			$\sqrt{x}$ (remainder $\neq 0$)

(These functions are described later in this section.)

* Appendix A includes a table of the relevant functions and how they affect the carry and out-of-range flags.

Example: The following simple additions set and then clear the carry flag (4).

Keystrokes	**Display**	(STATUS : 2–16–0000)*
HEX FFFF ENTER	**FFFF h**	Hex mode.
1 +	**0 h**	**C** annunciator: carry occurred and flag 4 set.
1 +	**1 h**	Carry flag cleared because no carry occurred.

Flag 5: Out-of-Range (G)

Flag 5 and the **G** annunciator are set if the correct result of an operation cannot be represented in the current word size and complement mode. For the + and − operations, this corresponds to the "overflow" condition on most computers.

The functions below either set or clear flag 5 and the **G** annunciator whenever they are performed in Integer mode:

+	−	×	÷	ABS	CHS
DBLX†	DBL÷†				

In addition, the arithmetic operators +, −, ×, and ÷ will affect flag 5 in Floating-Point Decimal mode. The FLOAT function also affects flag 5. Refer to section 5 for details.

When a result is out-of-range, the lower bits (as many as fit in the given word size) of the full answer will be returned. If the operation was × or ÷ in 1's or 2's Complement mode, the most significant bit (sign bit) returned will match the sign bit of the full answer.

* Throughout this manual, this status display is used to indicate what the machine status (as explained on page 37) must be for the examples to work as shown.

† Always clears flag 5.

Keystrokes	Display	($\boxed{\text{STATUS}}$: 2–16–0000)
$\boxed{\text{DEC}}$ 32767 $\boxed{\text{ENTER}}$	**32767 d**	
2 $\boxed{\times}$	**32766 d**	**G** annunciator displayed and flag 5 set; overflow. Leading binary digit is zero; number is positive.
$\boxed{\text{g}}$ $\boxed{\text{CF}}$ 5	**32766 d**	Clears flag 5.

Flag 5 can also be set in the course of a running program; this will not halt program execution.

Arithmetic Functions

Addition, Subtraction, Multiplication, and Division

The arithmetic operations $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, and $\boxed{\div}$ can be performed using integers in any of the four number bases. The operands, which can be entered in different bases, must be in the Y- and X-registers. After the operation is performed, the stack drops and the result is placed in the X-register.

In Integer mode, $\boxed{\div}$ performs an integer division. The fractional part of the quotient is truncated.

All the arithmetic operators except $\boxed{\times}$ will set or clear flag 4 and flag 5 whenever executed. $\boxed{\times}$ affects flag 5 only.

Example: Find $(5A0_{16}) \div (177764_8)$.

Keystrokes	Display	($\boxed{\text{STATUS}}$: 2–16–0000)
$\boxed{\text{HEX}}$ 5A0 $\boxed{\text{ENTER}}$	**5A0 h**	Enters first number.
$\boxed{\text{OCT}}$ 177764	**177764 o**	Changes to Octal; keys in second number.
$\boxed{\div}$	**177610 o**	Result in base 8. Since a carry was not generated, the result is exact.
$\boxed{\text{HEX}}$	**FF88 h**	Converts to base 16.

Addition and Subtraction in 1's Complement Mode. In 2's Complement and Unsigned modes, the result of an addition or subtraction is simply the sum or difference of the two bit patterns in the X- and Y-registers. In 1's Complement mode, however, the result of an addition is affected by the occurrence of a carry, and the result of a subtraction is affected by the occurrence of a borrow. If a carry out of the most significant bit occurs, 1 is added to the result. If a borrow into the most significant bit occurs, 1 is subtracted from the result. Both cases set flag 4.

($\boxed{\text{STATUS}}$: 1–04–1000)

Carry

$$\begin{array}{r} -1 \\ +(-1) \\ \hline -2_{10} \end{array} \qquad \begin{array}{r} {}^{1\,1\,1} \\ 1110 \\ +1110 \\ \hline 1100 \\ +\quad\;\; 1 \\ \hline 1101_2 \end{array}$$

No Carry

$$\begin{array}{r} -3 \\ +\;\;3 \\ \hline -0_{10} \end{array} \qquad \begin{array}{r} 1100 \\ +0011 \\ \hline 1111_2 \end{array}$$

Borrow

$$\begin{array}{r} 3 \\ -4 \\ \hline -1_{10} \end{array} \qquad \begin{array}{r} \overset{1}{\cancel{0}}{}^{1}011 \\ -0100 \\ \hline 1111 \\ -\quad\;\; 1 \\ \hline 1110_2 \end{array}$$

No Borrow

$$\begin{array}{r} 6 \\ -5 \\ \hline 1_{10} \end{array} \qquad \begin{array}{r} 01\overset{0}{\cancel{1}}{}^{1}0 \\ -0101 \\ \hline 0001_2 \end{array}$$

The Carry Flag During Addition. The carry flag (flag 4, **C** annunciator) will be set whenever a binary addition results in a carry "out of" the most significant bit. If an addition does not result in such a carry, the carry flag is cleared. This is the same for all complement modes.

($\boxed{\text{STATUS}}$: 2–04–1000)

Carry Set

$$\begin{array}{r} -6 \\ +(-4) \\ \hline 6_{10} \end{array} \qquad \begin{array}{r} {}^{1}1010 \\ +1100 \\ \hline 0110_2 \end{array}$$

(incorrect, so out-of-range flag set also)

Carry Cleared

$$\begin{array}{r} 6 \\ +\;1 \\ \hline 7_{10} \end{array} \qquad \begin{array}{r} 0110 \\ +0001 \\ \hline 0111_2 \end{array}$$

The Carry Flag During Subtraction. The carry flag (flag 4, **C** annunciator) will be set whenever a binary subtraction results in a borrow into the most significant bit. Otherwise, the carry flag is cleared. This is the same for all complement modes. (Note that subtraction in the HP-16C is *not* computed as the addition of a negative number; this affects how carry generation occurs.)

(STATUS : **2–04–1000**)

Carry Set		**Carry Cleared**	
-6	$\overset{0_1}{\cancel{1}}010$	6	$01\overset{0_1}{\cancel{1}}0$
$-(-4)$	-1100	-1	-0001
-2_{10}	1110_2	5_{10}	0101_2

The Out-of-Range Flag. Arithmetic results that cannot be shown in the current word size and complement mode set the out-of-range flag. For ÷, this occurs only in 2's Complement mode when the largest possible negative number is divided by −1.

Example: With a word size of 4 bits, calculate (7 + 6) in base 2 and observe the effect on flags 4 and 5.

Keystrokes	**Display**	(STATUS : **2–04–0000**)
BIN		Binary mode.
111 ENTER	**111 b**	7.
110	**110 b**	6.
+	**1101 b**	-3. Flag 5 (out-of-range) set; flag 4 (carry) cleared.

Remainder After Division and RMD

In division, only the integer portion of the result is returned to the X-register. If the remainder is not zero, flag 4 (carry) and the **C** annunciator are set. If the remainder is zero, flag 4 is cleared.

To obtain the remainder instead of the quotient, press f RMD instead of ÷. This calculates $|y|$ MOD $|x|$. The sign of the result matches the sign of the dividend (that is, the sign of y).

Keystrokes	Display	([STATUS]: 2–16–0000)
[HEX] 66 [ENTER]	**66 h**	Hexadecimal mode.
7 [÷]	**E h**	**C** annunciator; flag 4 set. 66/7 leaves a remainder.
2 [÷]	**7 h**	No annunciator; flag 4 cleared. E/2 leaves no remainder.
4 [f] [RMD]	**3 h**	Remainder of 7/4.

Square Root

The [$\sqrt{x}$] function calculates the square root of the number in the X-register. The fractional part of the square root is truncated. If this fraction is not zero, flag 4 (carry) is set; otherwise, flag 4 is cleared.

Negative Numbers and Complementing

Changing Signs. The [CHS] function (*change sign*) will change the sign, forming the complement (1's or 2's) of the number in the X-register. If the X-register holds the largest possible negative number in 2's Complement mode, the only effect of pressing [CHS] will be to set flag 5 (out-of-range).

In Unsigned mode, [CHS] forms a 2's complement and sets flag 5 (**G** annunciator) as a reminder that a negative number is outside the range of Unsigned mode.

To key in a negative number, press [CHS] *after* its digits have been keyed in. [CHS] terminates digit entry in Integer mode.

Absolute Value. Pressing [g] [ABS] converts the number in the X-register to its absolute value, forming the 1's or 2's complement of a negative number. There is no change if the calculator is in Unsigned mode or if the number is positive.

If the X-register holds the largest possible negative number in 2's Complement mode, the only effect of [ABS] will be to set flag 5 (out-of-range).

Logical Operations

The logical (Boolean) operations NOT, OR, AND, and EXCLUSIVE OR return the results of a bit-by-bit analysis of a binary number. The functions [OR], [AND], and [XOR] operate on the bits in corresponding positions of the words in the X- and Y-registers; the stack then drops and the result is placed in the X-register. The operator [NOT] acts only upon the word in the X-register; the stack does not drop.

NOT

The [NOT] function inverts the values of all bits in the binary number in the X-register. It is equivalent to forming the 1's complement, that is, using [CHS] in 1's Complement mode. Only the X-register is affected.

Keystrokes	Display	([STATUS]: 2–16–1000)
[BIN] 11111111	**11111111 b**	Binary mode.
[f] [NOT]	**00000000.b**	1's complement of
[f] [WINDOW] 1	**11111111 b.**	00000000 11111111_2 is 11111111 00000000_2.

AND

The [AND] function (the *logical product*) compares each corresponding bit in two words. Each resulting bit is 1 only if both corresponding operand bits are 1; otherwise, it is 0.

The use of [AND] is illustrated under Masking, page 51.

OR

The [OR] function (the *logical sum*) compares each corresponding bit in two words. Each resulting bit is 0 only if both operand bits are 0's.

Example: Perform a logical OR to determine which bits are zero in *both* 10101_2 and 10011_2.

Keystrokes	Display	([STATUS]: 2–16–0000)
10101 [ENTER]	**10101 b**	
10011	**10011 b**	
[f] [OR]	**10111 b**	Bit 3 (represented by the zero) and all bits to the left of bit 4 are zero in both of the given words.

EXCLUSIVE OR

The [XOR] function (the *logical difference*) compares the corresponding bits in two words and yields a 1 only if two corresponding bits are different.

Example: Use the [XOR] function to determine if two binary quantities (01010101_2 and 01011101_2) are the same. A 1 in the result signifies that the two quantities are different at those bit position(s).

Keystrokes	Display	([STATUS]: 2-16-0000)
1010101 [ENTER]	1010101 b	
1011101	1011101 b	
[f] [XOR]	1000 b	The two numbers differ in the fourth bit from the right.

Shifting and Rotating Bits

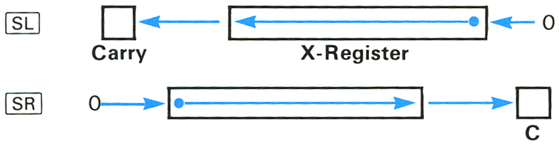
Shifting and rotating operations cause the bits of a word to be moved left or right. The fate of the bit moved off the end of the word, and the value of the bit entering the vacated position, depend upon the type of shift or rotate performed.

Flag 4 (carry) is set or cleared by any shift or rotate function, except [LJ] (*left-justify*), as shown in the diagrams below.

Shifting Bits

The HP-16C can perform two types of shifts on the contents of the X-register: a *logical shift* or an *arithmetic shift.* The latter preserves the sign bit. The HP-16C can also *left-justify* the contents of the X-register.

Logical Shifts. Pressing [f] [SL] (*shift left*) or [f] [SR] (*shift right*) moves all the bits of the word in the X-register one bit to the left or right. Bits shifted out of the word are shifted into the carry bit, and the previous state of the carry bit is lost. The new bits generated at the opposite end of the word are always zeros.

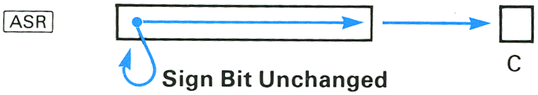


Left-Justify. To left-justify a bit pattern within its word size, press [g] [LJ]. The stack will lift, placing the left-justified word in the Y-register and the count (number of bit-shifts necessary to left-justify the word) in the X-register. The carry flag is not affected by [LJ].

Example: Left-justify the binary value 1111 in a word size of eight.

Keystrokes	Display	([STATUS]: 2-08-0000)
1111	1111 b	
[g] [LJ]	100 b	The count: four bit-shifts to left-justify the word.
[R↓]	11110000 b	Left-justified word.

Arithmetic Shift Right. Pressing [g] [ASR] (*arithmetic shift right*) will move the contents of the word in the X-register one bit to the right, as does [SR]. However, instead of placing a zero into the new place at the left of the word, *the sign bit is regenerated.* (In Unsigned mode, which has no sign bit, [ASR] operates like [SR].) The carry bit is set if a 1 is shifted out of the X-register and cleared if 0 is shifted out.



Example: Shifting a positive binary number to the right n places is equivalent to dividing it by 2^n. Since it regenerates the sign bit, an arithmetic shift also can be used to divide an even negative integer by 2.* Divide 01111111 (word size 8) by 2^3, then divide 10000000 by 2^3.

Keystrokes	Display	([STATUS]: 2-08-0000)
[g] [SF] 3		Allows display of leading zeros.
1111111	01111111 b	

* For *odd* negative integers in 2's Complement mode, [ASR] gives a result one less than division by 2.

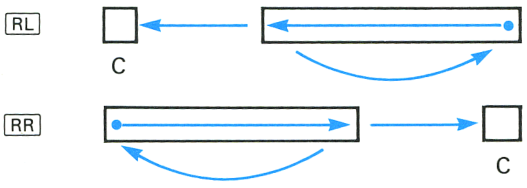
Keystrokes	Display	([STATUS]: 2–08–0000)
[f] SHOW [DEC]	127 d	([ENTER] is not needed because this function terminates digit entry.)
[f] [SR] [f] [SR] [f] [SR]	00001111 b	Each shift performs an *integer* divide by 2 and sets flag 4 because a 1 is shifted into the carry bit.
[f] SHOW [DEC] (hold)	15 d	
(release)	00001111 b	
10000000	10000000 b	
[f] SHOW [DEC]	–128 d	
[g] [ASR] [g] [ASR]	11100000 b	
[g] [ASR]	11110000 b	Sign bit is regenerated and carry flag is cleared with each shift.
[f] SHOW [DEC]	–16 d	
(release)	11110000 b	

Rotating Bits

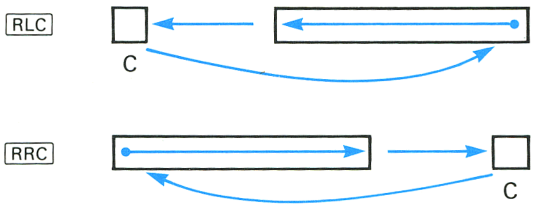
There are three general types of rotate functions on the HP-16C, encompassing eight different functions.

- Rotate left and right ([RL], [RR]).
- Rotate left and right "through the carry bit" ([RLC], [RRC]).
- Rotate *n* places ([RLn], [RRn], [RLCn], [RRCn]).

Rotation. Pressing [f] [RL] (*rotate left*) or [f] [RR] (*rotate right*) causes the contents of the X-register to rotate (or "circularly shift") one bit to the left or right. Bits shifted out of the word re-enter it at the other end. The carry flag is set if a 1 bit is rotated around the end, and is cleared if a zero is rotated around the end.



Rotation Through the Carry Bit. The $\boxed{\text{RLC}}$ and $\boxed{\text{RRC}}$ (*rotate left through carry* and *rotate right through carry*) functions respectively load the leftmost or rightmost bit of a word into the carry bit, and move the carry bit into the other end of the word.



Rotating More Than One Bit at a Time. Given a bit pattern in the Y-register and n in the X-register, pressing $\boxed{\text{f}}$ $\boxed{\text{RLn}}$, $\boxed{\text{f}}$ $\boxed{\text{RRn}}$, $\boxed{\text{g}}$ $\boxed{\text{RLCn}}$, or $\boxed{\text{g}}$ $\boxed{\text{RRCn}}$ will rotate the pattern $|n|$ bits. The stack drops, placing the result in the X-register.

The status of the carry flag (flag 4) is the same as if $\boxed{\text{RL}}$, $\boxed{\text{RR}}$, $\boxed{\text{RLC}}$, or $\boxed{\text{RRC}}$ were performed $|n|$ times. For instance, executing $\boxed{\text{RRn}}$ with $n = 3$ will set the carry flag only if the third bit from the right (bit 2) is 1.

Example: Develop a keystroke sequence that will serve to rotate left *as one word* a 16-bit word *divided into two separate 8-bit words* held in two separate registers. For instance, with a word size of eight bits, rotate the word 00011100 11100111.

Keystrokes	**Display**	($\boxed{\text{STATUS}}$: **2–08–1000**)
11100	**00011100 b**	High order portion of 16-bit word.
$\boxed{\text{f}}$ $\boxed{\text{SL}}$	**00111000 b**	Moves most significant bit into carry bit.
$\boxed{\text{g}}$ $\boxed{\text{LSTx}}$	**00011100 b**	Recovers high-order portion.
11100111	**11100111 b**	Low-order portion of 16-bit word.

Keystrokes	Display	([STATUS]: 2–08–1000)
[g] [RLC]	11001110 b	Carry bit (most significant bit of high-order portion) moves into least significant bit position of low-order portion.
[x≷y]	00011100 b	Switches X- and Y-registers.
[g] [RLC]	00111001 b	C cleared: carry bit moved into first part of word and zero moved into carry bit.
[x≷y]	11001110 b	New word is 00111001 11001110.
[g] [CF] 3	11001110 b	Suppresses leading zeros.

Setting, Clearing, and Testing Bits

Individual bits in a word can be set to 1 or cleared to 0 using the [SB] (*set bit*) and [CB] (*clear bit*) functions. In a manner analogous to flag-testing, you can also test for the presence of a set bit with [B?]. If executed in a program, the result affects program execution.

To set, clear, or test a specific bit in a word:

- The word containing the specific bit must be in the Y-register.
- The magnitude of the number in the X-register specifies the number of the bit to be set, cleared, or tested.

When the key ([SB] or [CB]) is pressed, the stack drops and the word affected returns to the X-register.

Bits are numbered from zero to one less than the word size, with the least significant bit as bit number 0.

Keystrokes	Display	([STATUS]: 2–16–0000)
11111111 [ENTER]	11111111 b	Enters quantity and copies it into the Y-register.
11	11 b	Bit number 3.
[f] [CB]	11110111 b	Stack drops; resulting word is in X-register.

Testing for the presence of a given bit (f B?) is a *conditional test* useful in programming: a decision for program execution can be based on the bit pattern of a number. (The X- and Y-registers must contain the proper parameters, as noted above.) Section 9 describes conditional branching.

Masking

The MASKL (*mask left*) and MASKR (*mask right*) functions create left- or right-justified strings of 1 bits. The magnitude of the number in the X-register is used to determine how many 1's will comprise the mask. Upon execution, the mask pattern is placed in the X-register (the stack does not move).

You can create a mask as large as the word size. To place a mask in the middle of the field of a number, use a shift function in conjunction with MASKL or MASKR.

Example: The ASCII representation of a two-digit number occupies 16 bits—eight bits per digit. Given an ASCII "65" (0011 0110 0011 0101), extract the high-order digit (6), thereby converting half of this ASCII code to binary coded decimal.

```
     0011 0110 0011 0101  ASCII "65" ("3", "6", "3", "5").
AND  0000 1111 0000 0000  Mask.
     -------------------
     0000 0110 0000 0000  The extracted, high-order digit ("6").
```

You can save keystrokes in this example by shifting the digits into position before masking.

Keystrokes	**Display**	(STATUS: 2–16–0000)
HEX 3635 ENTER	**3635 h**	
8 f RRn	**3536 h**	Rotates word eight bits to the right to right-justify the desired hex digit (6).
4 f MASKR	**F h**	Right-justifies a mask of four 1 bits (1111) in the 16-bit word.
f AND	**6 h**	Extracts the rightmost four bits (6).

Bit Summation

Pressing [g] [#B] (*number of bits*) sums the bits in the X-register and returns that value to the X-register. The bit pattern is saved in the LAST X register. No stack lift occurs. (In word sizes 1 and 2, the result must be interpreted in Unsigned mode.)

"Double" Functions

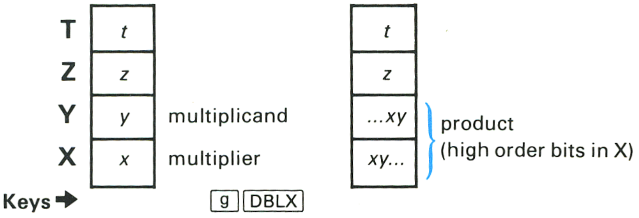
The HP-16C provides several "double" functions: [DBLX] (*double multiply*), [DBL÷] (*double divide*), and [DBLR] (*double remainder*). These functions allow the *exact* calculation of a product double the given word size and the *exact* calculation of a quotient and remainder from a dividend of double word size.

To obtain meaningful double numbers as results in Hexadecimal and Octal modes, the word boundary (which is based on the numbers of *bits*) must not "split" a digit. Therefore, you should specify a compatible word size: a multiple of four in Hex mode and a multiple of three in Octal mode.*

Double Multiply

The [DBLX] function multiplies two single-word quantities in the X-and Y-registers to yield a double-word result in the X- and Y-registers. (The stack does not drop.) The result is right-justified, with the *least significant* bits returned in the Y-register and the *most significant* bits returned in the X-register.

The stack contents during this operation are shown below. The stack is filled with the values *t*, *z*, *y*, and *x*, each register holding one word.



* Section 7 (Programming Basics) includes a program for using [DBLX] with Decimal mode. Refer to page 78.

Example: To illustrate double multiplication, the calculation of (7×6) in binary with word size 5 and 2's Complement is shown below.

$$\begin{array}{r} 7 \\ \times\, 6 \\ \hline 42_{10} \end{array} \qquad \begin{array}{r} 00111 \\ \times\ 00110 \\ \hline \underbrace{00001}_{X}\ \underbrace{01010_2}_{Y} \end{array}$$

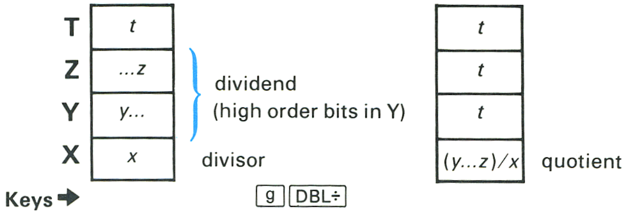
Five bits in Y.
Five bits in X.
Ten-bit representation of 42_{10} split between X- and Y-registers.

Keystrokes	**Display**	(STATUS : 2–05–1000)
BIN 111 ENTER	**00111 b**	Binary mode.
110 g DBLX	**00001 b**	Most significant bits are put in X.
$x \gtrless y$	**01010 b**	Least significant bits are put in Y. Result is therefore 00001 01010_2.

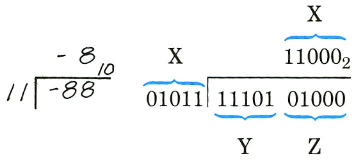
Double Divide

The DBL÷ function computes the quotient of a dividend of double word size in the Y- and Z-registers—the most significant bits of which are in the Y-register—divided by a single-word divisor in the X-register. The stack drops twice, placing the *single*-word result in the X-register.

Error 0 occurs if the answer cannot be represented in a single word size. Flag 4 (carry) is set if the remainder is not equal to zero. The stack contents during this operation are:



Example: The calculation of (-88 ÷ 11) in binary with word size 5 and 2's Complement mode is shown below.



Five-bit result in X.

Ten-bit representation of -88_{10} split between Y- and Z-registers.

Keystrokes	Display	(STATUS : 2–05–1000)
1000 ENTER	**01000 b**	Least significant bits of 10-bit dividend go into Z-register.
11101 ENTER	**11101 b**	Most significant bits of 10-bit dividend go into Y-register.
1011 g DBL÷	**11000 b**	Quotient.
g CF 3	**11000 b**	Restores suppression of leading zeros.

Double Remainder

The DBLR function operates like DBL÷ except that the remainder is returned instead of the quotient. If the *quotient* exceeds 64 bits, **Error 0** results.

The remainder is determined as for the RMD function (page 43), with the sign of the result matching the sign of the dividend.

Example: Applying Double Divide

Compute the quotient of $\dfrac{5714AF2_{16}}{7E14684_{16}}$ to 16 hexadecimal places.

Although the result is a fraction, this problem can be solved in Integer mode by first finding the integer quotient of

$$\frac{5714AF2\overbrace{000\ldots0}^{16\text{ zeros}}{}_{16}}{7E14684_{16}}$$

and then placing a decimal point to the left of the result (thereby dividing the result by 2^{64}). Use DBL÷ to accommodate a numerator this large.

Keystrokes	Display	(STATUS : 2–64–0000)
HEX		Hex mode.
f SET COMPL UNSGN		Unsigned mode allows a larger result, thereby preventing an out-of-range result.
0 ENTER	**0 h**	Double-sized dividend is $5714AF2 \times 2^{64}$.
5714AF2 ENTER	**5714AF2 h**	
7E14684 g DBL÷	**7E985d8C .h**	Carry bit set.
f WINDOW 1	**b0d06F6A h.**	Result is B0D06F6A $7E985D8C_{16}$, so answer to original problem is $0.B0D06F6A\ 7E985D8C_{16}$.

Section 5

Floating-Point Numbers

To cover other aspects of computing, the HP-16C was designed to perform floating-point decimal arithmetic also. In Floating-Point Decimal mode, numbers are left-justified and the word size is automatically set to 56 bits.

> **Note:** Numbers in Floating Point mode and Integer mode are represented in two different, incompatible formats.* Therefore, any values stored in the storage registers in one format are not meaningfully converted when the calculator switches to the other format. Their integrity is retained, however, when the calculator is restored to the original mode.

Converting to Floating-Point Decimal Mode

The [FLOAT] (*floating point*) function establishes Floating-Point Decimal mode and converts the contents of the X- and Y-registers (as explained below) to a floating-point decimal number.

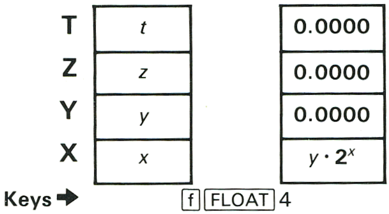
Pressing [f] [FLOAT] {0 to 9, [·]} will set Floating-Point Decimal mode. The number you specify determines how many decimal places will be displayed; specifying [·] will set scientific notation. (If the calculator is already in Floating Point mode, no further conversion of the stack takes place.)

Conversion in the Stack

When you make the conversion to Floating Point mode, you might want to be able to retain a certain number for further use. The HP-16C provides a conversion routine that allows you to retain and reuse a value from Integer mode when the calculator switches to Floating Point mode. Upon executing [FLOAT] in Integer mode, the numbers in the Y- and X-registers are converted to the floating-point decimal equivalent of $(y)(2^x)$, which is then placed into the X-register. The Y-, Z-, and T-registers are cleared.†

* Floating Point mode in BCD (Binary Coded Decimal) form and Integer mode in binary form.

† Appendix D provides a program for a conversion between the proposed IEEE floating-point binary format and HP-16C floating-point decimal format.



If $y \cdot 2^x$ is greater than $9.999999999 \times 10^{99}$, flag 5 (out-of-range) is set and the overflow display (all 9's) results. If there is no overflow, flag 5 is cleared.

Example: Convert $25E47_{16}$ to a decimal, floating-point format.

Keystrokes	Display	(STATUS : 2–64–0000)
HEX		Hex mode.
25E47 ENTER	**25E47 h**	Mantissa.
0	**0 h**	Exponent of 2.
f FLOAT 2	**155,207.00**	Sets Floating-Point Decimal mode and a display of two decimal places. The number is equivalent to $(25E47_{16}) \cdot 2^0$.

Other Effects of Converting to Floating Point Mode

Switching from Integer mode (Hexadecimal, Decimal, Octal, or Binary mode) into Floating-Point Decimal mode also sets the following conditions:

- The word size is set to 56.
- The stack (except the X-register) and the LAST X register are cleared. Stack lift is enabled.
- The storage registers are *not* cleared. However, an attempt to recall register contents (including the Index register) that were *not* stored in Floating Point mode usually will result in an **Error 6**.*

* Refer to page 68 for details.

- The complement mode functions remain active, but will *not* affect the arithmetic functions or number representations. The complement mode *will* affect the conversion of the X-register when the calculator returns to Integer mode.

Digit Entry and Other Display Formats

Changing Signs. Pressing [CHS] (*change sign*) will reverse the sign of the displayed number. To key in a negative number, press [CHS] after its digits have been keyed in. This does not terminate digit entry in Floating Point mode.

Scientific Notation. Pressing [f] [FLOAT] [·] sets Floating Point mode *and* sets scientific notation display format. Numbers will be displayed with six decimal places.

Exponents. Numbers with exponents are entered using [EEX] (*enter exponent*). First key in the mantissa, then press [f] [EEX] and key in the exponent. (For a negative mantissa, press [CHS] before pressing [EEX].) For a negative exponent, press [CHS] after keying in the exponent.

Digits from the mantissa that spill into the exponent field will disappear from the display when [EEX] is pressed but will be retained internally.*

Mantissa Display. Regardless of the display format in Floating-Point Decimal mode, the calculator internally represents each number as a 10-digit mantissa with a two-digit exponent of 10. If you want to view the full 10-digit mantissa of a number in the X-register, press [f] CLEAR [PREFIX]. The full mantissa will be displayed as long as the [PREFIX] key is held down.

Keystrokes	**Display**	([STATUS]: 2-56-0000)
[f] [FLOAT] 3		
45 [g] [$\sqrt{x}$]	6.708	
[f] CLEAR [PREFIX]		
(hold)	6708203932	
(release)	6.708	

*To prevent a misleading display, [EEX] will not operate with a number having more than seven digits to the left of the decimal point, nor with a mantissa smaller than 0.000001. To key in such a number, use a form having a greater exponent value (whether positive or negative).

Overflow and Underflow. When the result of a floating-point calculation in the X-register is a number with a magnitude greater than $9.999999999 \times 10^{99}$, $\pm 9.999999999 \times 10^{99}$ is placed in the affected register (the last three digits of the mantissa are not displayed). Flag 5 (out-of-range) is set and the **G** annunciator displayed.

If the result of a floating-point calculation in the X-register is a number with a magnitude less than $1.000000000 \times 10^{-99}$, that number will be replaced by zero. Overflow and underflow do *not* halt program execution.

Returning to Integer Mode

Integer mode is restored (and Floating Point mode exited) when you set one of the number base modes.

Conversion in the Stack

When exiting Floating Point mode, the X- and Y-registers undergo the reverse of the conversion when Floating Point mode is set. Considering the number in the X-register to be in the form $(y)(2^x)$, an integer y is placed in the Y-register and a power x of 2 is placed in the X-register. The value for y is defined such that $2^{31} \leqslant |y| < 2^{32}$ unless $y = 0$. That is, y, the mantissa, is rounded to a 32-bit integer. The value for x, the exponent, is determined by y such that $(y)(2^x)$ equals the value in the X-register before mode conversion.

The new x- and y-values will be expressed in the number base mode specified. (In Unsigned mode, the absolute values of x and y are placed in X and Y.)

The conversion of an integer pair y, x from Integer mode into Floating-Point Decimal mode and back to Integer mode generally will not restore the *original* pair y, x, but an *equivalent* one.

Example: Convert 1.284×10^{-17} to a decimal integer multiplied by a power of 2.

Keystrokes	**Display**	(STATUS: 2-56-0000)
1.284	1.284	
f EEX	1.284 00	
17 CHS	1.284 -17	

Keystrokes	Display	
DEC	-88 d	Converts to Integer mode. X-register contains exponent of 2.
$x \gtrless y$	73787526 .d	Mantissa.
f WINDOW 1	39 d.	Answer is $(3973787526 \times 2^{-88})_{10}$.

Example: The following key sequence shows that the conversion of 1 in Integer mode ($1 \cdot 2^0 = 1$) to Floating Point mode and back to Integer mode ($80000000_{16} \times 2^{-31}$) yields equivalent but different representations of 1.

Keystrokes	Display	(STATUS: 2–56–0000)
HEX		
1 ENTER	1 h	$y = 1$. $\lvert y \rvert$ is *not* an integer such that $2^{31} \leqslant \lvert y \rvert < 2^{32}$.
0	0 h	$x = 0$.
f FLOAT 4	1.0000	$(y) \cdot (2^x) = (1) \cdot (2^0) = 1$.
HEX	FFFFFFE1 .h	Converts back to Integer mode.
f SHOW DEC	-31 d	$x = -31_{10}$.
$x \gtrless y$	80000000 h	$y = 80000000_{16} = 2^{31}$. y is now an integer such that $2^{31} \leqslant \lvert y \rvert < 2^{32}$ and $y \cdot 2^{-31} = 1$.

Other Effects of Converting to Integer Mode

When you switch out of Floating-Point Decimal mode into Integer mode, the following conditions will exist:

- Word size *remains* 56 bits.
- The stack and storage registers are *not* cleared, although any storage register values which were not entered in Integer mode will be altered from their original meanings.*
- The complement mode does not change.

* Such values *are retrievable in their original form* when the calculator converts back to Floating Point mode, assuming you have not changed the register contents.

Floating-Point Arithmetic

Functions

All of the arithmetic functions which operate in Integer mode ($\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, $\boxed{\div}$, $\boxed{\sqrt{x}}$) also operate in Floating Point Decimal mode. The $\boxed{\div}$ and $\boxed{\sqrt{x}}$ functions are not limited to integer answers in Floating Point mode.

The $\boxed{1/x}$ (*reciprocal*) function, which is active only in Floating-Point Decimal mode, calculates the reciprocal of the number in the X-register.

The Out-of-Range Flag

The arithmetic functions $\boxed{+}$, $\boxed{-}$, $\boxed{\times}$, and $\boxed{\div}$ are the only functions in Floating-Point Decimal mode that affect flag 5 (out-of-range). Flag 5 is set or cleared whenever these functions are executed. Executing $\boxed{\text{FLOAT}}$ {0 to 9, $\boxed{\cdot}$} likewise affects flag 5.

The carry flag (flag 4) is not affected in Floating-Point Decimal mode.

Functions Not Active in Floating Point Mode

Generally speaking, the display and number control functions and bit manipulation operations are *not* active in Floating-Point Decimal mode. Appendix B contains a complete list of all operations that are not active in Floating Point mode.

Digit Separators

The HP-16C is set to use a period as a radix mark (separating integer and fractional parts of a number) and a comma between groups of three integer digits. You can reverse this setting to conform to the numerical convention in many countries. This procedure ($\boxed{\text{ON}}$ / $\boxed{\cdot}$) will reverse the current setting:

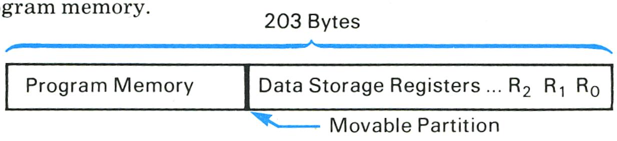
1. Turn off the calculator.
2. Press and hold $\boxed{\text{ON}}$.
3. Press and hold $\boxed{\cdot}$.
4. Release $\boxed{\text{ON}}$, then release $\boxed{\cdot}$.

Section 6

Memory and Storage

Memory Allocation

Data storage, in the form of *registers*, and program memory, in the form of *program lines*, share a common memory space in the HP-16C. This memory space consists of 203 bytes (eight bits to one byte), all of which are initially* allocated to data storage. However, as you enter program instructions (as explained in section 7, Programming Basics) memory space is *automatically allocated* to program memory.



Converting Storage Registers to Program Memory

The automatic partitioning of memory space from data storage into programming takes place in segments of seven bytes, each segment worth seven program lines. Starting with the first line you store in a program and continuing with each succeeding seventh line, seven bytes (lines) of memory are converted from data storage into program memory.

Automatic Memory Allocation (Bytes)

Program Lines* Recorded	Program Memory Allocated	Storage Memory Allocated
0 bytes	0 bytes	203 bytes
1 to 7	7	196
8 to 14	14	189
15 to 21	21	182
⋮	⋮	⋮
190 to 196	196	7
197 to 203	203	0

* One line equals one byte.

* "Initially" means the condition of the calculator when it leaves the factory or when Continuous Memory is reset.

Therefore, the number of bytes dedicated to storage registers and the number of bytes dedicated to programming are always multiples of seven.

> **Note:** The calculator converts data storage registers to program memory in reverse numerical order, from the highest numbered to the lowest numbered. Furthermore, *any data contained in a storage register will be lost* when that register is converted to lines of program memory.

Converting Program Memory to Storage Registers

Once you have stored program instructions they are well protected. The allocation of program memory space *back* to data storage registers is accomplished only by your intentionally deleting program instructions—either singly or all at once (by CLEAR [PRGM] or Continuous Memory reset). What is deleted from program memory is reallocated to data storage memory in increments of seven bytes.

Note that this means *you cannot unintentionally lose program instructions*. If you try to address a storage register whose space is occupied by program lines, an **Error 3** (*nonexistent storage register*) will result.

Storage Register Size

Each register represents a word, and so its size depends on the current word size.* The size of a storage register is *always the smallest multiple of four bits* (half-bytes) equal to or greater than the current word size. For example, a current word size of either 13, 14, 15, or 16 will produce a storage register length of 16 bits (two bytes).

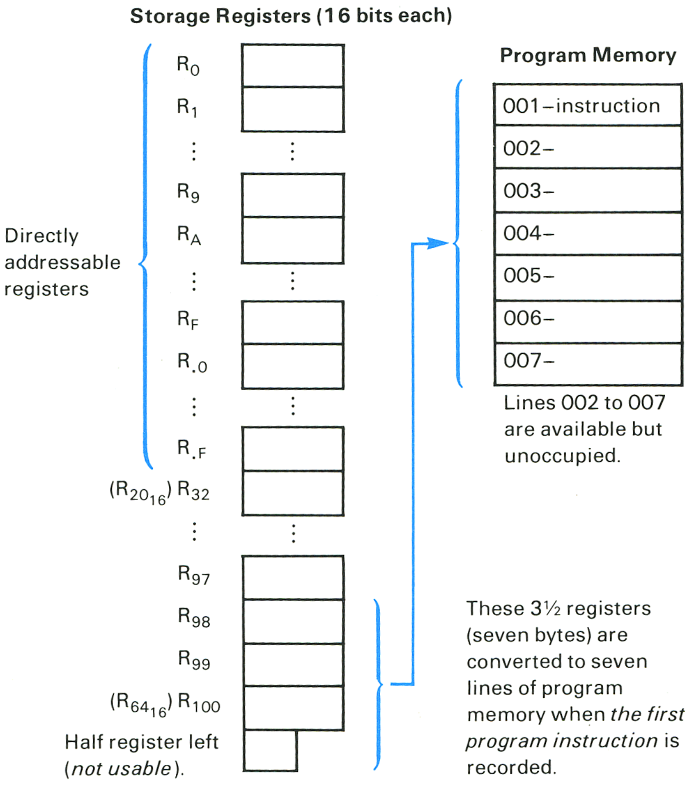
In Floating-Point Decimal mode, the word size and therefore the storage register size are automatically set to 56 bits (7 bytes).

The total possible number of data storage registers equals the number of available bytes (203 minus the bytes of program memory) divided by the number of 8-bit bytes per register. For example, if the current word size is 16 bits, then each register comprises 2 bytes (16 bits). If program memory is cleared (203 bytes

* Except the Index register (discussed later in this section), which is of constant size (68 bits) and *not* convertible to program memory.

available for storage), then 101 storage registers(R_0 to $R_{100_{10}}$) are available, since 203/2 = 101.5 and *half registers are not available for data storage.*

Given a word size of 16, the configuration of memory allocation would look like this with the first recorded program line:



Storage Registers (16 bits each)
Directly addressable registers
R_0
R_1
R_9
R_A
R_F
$R_{.0}$
$R_{.F}$
$(R_{20_{16}})$ R_{32}
R_{97}
R_{98}
R_{99}
$(R_{64_{16}})$ R_{100}
Half register left (*not usable*).
Program Memory
001–instruction
002–
003–
004–
005–
006–
007–
Lines 002 to 007 are available but unoccupied.
These 3½ registers (seven bytes) are converted to seven lines of program memory when *the first program instruction* is recorded.

When the eighth program instruction is recorded, another 3½ registers (seven bytes) will be converted to program memory. This will leave 94½ registers, only the whole 94 (R_0 to $R_{93_{10}}$ or $R_{5D_{16}}$) of which remain available for actual use as data storage registers. (The remaining half register will be used in the next conversion to program memory.)

Viewing the Status of Memory Allocation (MEM)

A temporary display of the current allocation of memory results when you press f MEM. (Hold MEM to prolong the display.) The information in the display will be

P – *B* r – *RRR*

B	is the number of bytes (program lines) which may be added to program memory before another seven bytes are allocated (diminishing available storage register space by seven bytes). $0 \leqslant B \leqslant 6$.
RRR	is the total number of storage registers currently available for data storage. $0 \leqslant RRR \leqslant 406$, with *RRR* always given in base 10. (Note the smallest possible register size is four bits, yielding a maximum number of 406 registers.)

In the situation diagrammed on the preceding page—initial memory configuration, word size 16—the MEM function would show the following information:

Keystrokes	**Display**	
ON / - BSP	0 h	Continuous Memory reset; word size defaults to 16 and program memory is cleared.
f MEM (hold) (release)	P-0 r-101 0 h	Zero lines to go before seven bytes will be converted to program memory. 101 storage registers available.
g P/R	000-	Sets calculator to Program mode; line 000.
1	001- 1	Records one program line.

Keystrokes	Display	
[f] [MEM]	P-6 r-098 001- 1	Six lines (bytes) to go before another seven bytes are converted to program memory. 98 storage registers available.
[f] CLEAR [PRGM]	000-	Clears program memory.
[g] [P/R]	0 h	Exits Program mode.

Storage Register Operations

When numbers are stored or recalled, they are *copied* between the display (X-register) and a data storage register. The total number of available storage registers will depend on the current word size and the amount of available memory, as explained above.

Of those registers available, up to 32 can be addressed *directly* by name (0 to 9 and A to F, .0 to .9 and .A to .F); the remainder (up to 69 in word size 16) can only be accessed *indirectly*, using the Index register (described later in this section). Since the highest-numbered registers are the first to be converted to program memory, it is wisest to store data in the lowest-numbered registers available. (See also the diagram on the inside back cover.)

Storing and Recalling Numbers Directly

To *store* or *recall* numbers between one of the directly accessible data storage registers and the X-register, press [STO] {0 to F, .0 to .F, [I]} or [RCL] {0 to F, .0 to .F, [I]}. The [STO] operation replaces the contents of the addressed register with a copy of the X-register contents; the [RCL] operation lifts the stack (if it is enabled) and then places a copy of the addressed register contents into the X-register.

If you attempt to address a nonexistent register (including one that cannot exist because the memory space is occupied by program lines), the display will show **Error 3**. Remember also that you can lose stored data values when data storage registers are *automatically* converted to program memory, and that the highest-numbered registers are converted first.

Example: Store 10^8 in R_0, recall it, and multiply by 2.

Keystrokes	Display	
f FLOAT 0		Display depends on last contents of X and Y.
f EEX 8	1 08	
STO 0	100,000,000.	
BSP	0.	
RCL 0	100,000,000.	
2 ×	200,000,000.	

Alteration of Register Contents

Whenever you change the current word size, the contents of the stack are affected (as explained under Word Size in section 3), *while those of the storage registers are not.* However, a change in word size *does* change the size and boundaries (and total number) of the storage registers. This can make a specific set of stored bits inaccessible in its original form—*until you restore the original word size.* Therefore, you can temporarily change the word size (for instance, for a calculation), and still recover your stored data when you return to the original word size.

Example: The following sequence illustrates what happens to data stored in R_0 and R_1 when the word size is doubled (from 16 to 32) and then restored to its original size.

Keystrokes	Display	
HEX 10 f WSIZE		Hex (Integer) mode; word size 16.
f CLEAR REG		Clears storage registers.
1234 STO 0	1234 h	Stores in R_0.
5678 STO 1	5678 h	Stores in R_1.
20 f WSIZE	5678 h	Doubles word size.
RCL 0	56781234 h	Current R_0 is now a concatenation of the old R_0 and R_1.
RCL 1	0 h	Current R_1 is cleared.
10 f WSIZE	0 h	Restores original word size.
RCL 0	1234 h	Original contents of R_0 and R_1 unchanged.
RCL 1	5678 h	

When the calculator converts between Integer and Floating-Point Decimal modes, the register contents are *not* changed. *However,* due to differences in the internal representation of the two modes, contents stored in Integer mode will not have the same value in Floating Point mode, and vice-versa. The integrity of the contents is preserved, however, when the calculator converts back to the original mode and word size.

Clearing Data Storage Registers

Pressing [f] CLEAR [REG] (*clear registers*) clears the contents of all data storage registers to zero. (It does not affect the stack or the LAST X register.) To clear a single data storage register, store zero in that register. Resetting Continuous Memory clears all storage registers and the stack.

The Index Register

The Index register (R_I) is a *permanent* storage register that can be used to *indirectly* address other storage registers, *indirectly* branch to program labels, and hold loop counters for program loop control. (The latter two uses are discussed in section 9.) Unlike other storage registers, *the Index register is always 68 bits,* regardless of current word size, and *it is never converted to lines of program memory.*

Abbreviated Key Sequences

Whenever the [I] or [(i)] key is used following another function key (such as [STO], [RCL], [GTO], or [GSB]), the [f] prefix key before the [I] or [(i)] can be omitted since the sequence is unambiguous. This is called an *abbreviated key sequence.* For example, [STO] [I] is shorter than [STO] [f] [I] but has the same effect.

Storing and Recalling Numbers in the Index Register

The contents of the Index register itself are accessed using the [I] function: [STO] [I], [RCL] [I], or [x⋛I]. A number stored in R_I will be represented in a 68-bit format, numerically equivalent to the number in the X-register. A number copied from R_I into X will be truncated to fit the word size, preserving the least significant bits.*

* If the contents of R_I are recalled in a word size smaller than that used to store the contents, the recalled value may not be numerically equivalent to the value in R_I.

Store and Recall. Numbers are copied into and out of the Index register using $\boxed{\text{STO}}\,\boxed{\text{I}}$ and $\boxed{\text{RCL}}\,\boxed{\text{I}}$, operating the same as the other registers.

X Exchange I. Analogous to the $\boxed{x \gtrless y}$ function, $\boxed{x \gtrless \text{I}}$ will exchange the contents of the X-register and the Index register.

Storing and Recalling Numbers Indirectly

Storage registers can be accessed *indirectly* using the $\boxed{\text{(i)}}$ function: $\boxed{\text{STO}}\,\boxed{\text{(i)}}$, $\boxed{\text{RCL}}\,\boxed{\text{(i)}}$, or $\boxed{x \gtrless \text{(i)}}$. The absolute value of the number in R_I determines the storage register address.* (In Floating-Point Decimal mode, only the integer part is used.) The table below shows the correspondence between R_I and storage register addresses.

Indirect Addressing

If R_I Contains:		$\boxed{\text{(i)}}$ Will Address:
0	(0_{16})	R_0
⋮	⋮	⋮
9	(9_{16})	R_9
10	(A_{16})	R_A
⋮	⋮	⋮
15	(F_{16})	R_F
16	(10_{16})	$R_{.0}$
⋮	⋮	⋮
31	$(1F_{16})$	$R_{.F}$
32	(20_{16})	R_{32} $(= R_{20_{16}})$
33	(21_{16})	R_{33} $(= R_{21_{16}})$
⋮	⋮	⋮

The $\boxed{\text{STO}}\,\boxed{\text{(i)}}$, $\boxed{\text{RCL}}\,\boxed{\text{(i)}}$, and $\boxed{x \gtrless \text{(i)}}$ functions are the *only* means to access a storage register beyond the first 32 (beyond 0 through .F), but they *can* be used for *any* register, as the table illustrates.

*The absolute value is computed using a 68-bit word size and the current complement mode.

Example: Store 3 in R_{326}. In order to have a storage register as high as R_{326} available for data storage, the word size must be at most four bits long (203 bytes/0.5 byte = 406 registers.) To be able to store a number as large as 326 in the Index register, however, the word size must be larger. (The Index register must be used to address any register higher than R_{31}, which is $R_{.F}$.) To store 3 in R_{326} will therefore require two manipulations of word size. Assume that the calculator is set to word size 4.

Keystrokes	**Display**	(STATUS : 2–04–0000)
DEC 0 f WSIZE		Sets a large word size (64).
326	**326 d**	
STO I	**326 d**	Stores 326_{10} in R_I.
4 f WSIZE	**6 d**	406 registers of 4 bits each are now available.
3 STO (i)	**3 d**	Stores 3 in R_{326}.
BSP	**0 d**	Clears display.
RCL (i)	**3 d**	Recalls contents of R_{326}.

Part II
HP-16C Programming

Section 7

Programming Basics

The next three sections are dedicated to explaining aspects of programming the HP-16C. These programming sections will first discuss basic techniques (The Mechanics), then give example(s) for the implementation of these techniques, and lastly discuss any details of operation (Further Information). This allows you to read only as far as you need to support your use of the HP-16C.

The Mechanics

Creating a Program

Programming the HP-16C is an easy matter, based simply on recording the keystroke sequence used when calculating manually. (This is called "keystroke programming.") To create a program out of a series of calculation steps requires two extra manipulations: deciding where and how to enter your data; and loading and storing the program. In addition, programs can be designed to make decisions and perform iterations through conditional and unconditional branching.

Stepping through the fundamentals of programming, we'll work through a program designed to concatenate two 16-bit words in the X- and Y-registers into one 32-bit word in the X-register.

Loading a Program

Program Mode. Press [g] [P/R] (*program/run*) to set the calculator to *Program mode* (**PRGM** annunciator on). Most functions are stored and not executed when keys are pressed in Program mode.

Keystrokes	Display	
[g] [P/R]	000–	Switches to Program mode; **PRGM** annunciator and line number displayed.

Keystrokes in Program mode become program instructions occupying program lines. These numbered lines indicate the calculator's position in program memory. Line 000 marks the beginning of program memory and cannot be used to store an instruction; the first line that contains an instruction is line 001. No program lines except 000 exist until instructions are stored in them.

Programs are usually started at line 001, though you *can* start a program at any existent line. As you enter instructions, any existing programs will be preserved and "bumped" down in program memory, thereby incrementing their line numbers.

Beginning a Program. Clearing program memory will erase all programs in memory and position the calculator to line 000. To do so, press [f] CLEAR [PRGM] *in Program mode.*

If the calculator is not at line 000 and you do *not* want to clear program memory, you can position the calculator to line 000 by pressing [f] CLEAR [PRGM] or [GTO] [·] 000 in *Run mode,* or by pressing [GTO] [·] 000 in *Program mode.* (The [GTO] [·] instruction cannot be recorded.)

A [LBL] (*label*) instruction—[g] [LBL] followed by a digit or letter *label* {0 to 9, A to F}—is used to define the beginning of a program or routine. The use of labels allows you to quickly select and run a particular program or routine.

Keystrokes	**Display**	
[f] CLEAR [PRGM]	000–	Clears program memory and sets calculator to line 000 (start of program memory).
[g] [LBL] A	001–43,22, A	Keycode for label "A".

Recording a Program. A program consists of the same keystrokes you would use to solve a problem manually. Keys pressed in Program mode are recorded in memory as programmed instructions.* The display contains a line number and *keycode(s).* Keycodes are one- or two-digit numbers indicating the position of keys on the keyboard (described in more detail later).

* Except for the *nonprogrammable functions*, which are listed on page 81.

Example: The body of the concatenation program is listed below. Assuming that two separate numbers are given in the X- and Y-registers, program lines 002 to 008 below will concatenate those two 16-bit words into one 32-bit word. The word initially in the X-register will become the most significant bits of the result.

Keystrokes	Display	
HEX	002– 23	
2	003– 2	Doubles the word size from 16 to 32, providing 16 extra bits to the left of the numbers in X and Y.
0	004– 0	
f WSIZE	005– 42 44	
g LSTx	006– 43 36	Brings back word size (32).
f SR	007– 42 b	Computes one-half of word size (16).
f RLn	008– 42 E	Shifts number left 16 bits.
f OR	009– 42 40	OR operation here concatenates the contents of X and Y.

Ending a Program.

- The instruction g RTN (*return*) will end a program, return to line 000, and halt.* This instruction can be omitted if the program is the last one in memory, since the end of the program memory contains an automatic RTN.
- The instruction R/S (*run/stop*) will stop a program *without* moving the line position to line 000.

Keystrokes	Display	
g RTN	010– 43 21	Optional if this is the last program in memory.

* Except when a subroutine return is pending, as discussed in section 9, page 94.

Running a Program

Run Mode. Switch back to Run mode (no **PRGM** annunciator displayed) when you are done programming by pressing [g] [P/R]. Program execution can take place only in Run mode.

Keystrokes	Display	
[g] [P/R]		Run mode; no **PRGM** annunciator displayed. Display will show previous result.

The position in program memory does not change when the calculator transfers between Run mode and Program mode. Whenever the calculator has been off, it "wakes up" in Run mode.

Executing a Program. In Run mode, press [GSB] *label*. This addresses a particular program and starts its execution. The display will flash **running**. (The [GSB] key is the same one used—under different circumstances—to "go to subroutine".)

Keystrokes	Display	([STATUS]: 2–16–0000)
[HEX] FFFE [ENTER]	**FFFE h**	Enter the first number into the X- and Y-registers.
DDDC	**dddC h**	Key the second number into the X-register. These digits will become the most significant ones.
[GSB] A	**dddCFFFE h**	The concatenated hex number.
[f] [STATUS]	**2–32–0000**	The word size is now 32.

Alternatively, you can position the calculator to a particular line using [GTO] [·] *nnn* (three-digit line number) or [GTO] *label* and then start execution by pressing [R/S].

Intermediate Program Stops

Use [f] [PSE] (*pause*) as a program instruction to *momentarily* stop a program and display an intermediate result. Use more than one [PSE] for a longer pause.

A [R/S] instruction will stop the program indefinitely at the line after the [R/S]. You can resume program execution (from that line) by pressing [R/S] in Run mode—that is, from the keyboard.

Data Input

Every program must take into account how and when data will be supplied. Data input can occur before executing the program or during planned interruptions in the program.

Prior Entry.

1. You can store the values (with [STO]) into a storage register, from which they will be recalled (with a programmed [RCL]) within the program.

   For example, the concatenation program could *recall* a value for the word size instead of writing it into the program:

Keystrokes	
[g] [LBL] A	
[HEX]	
[RCL] 1	Recalls the word size from R_1.
[f] [WSIZE]	
⋮	

2. If data will be used in the first line(s) of a program, you can enter it into the stack before starting the program. Don't start the program with [ENTER]—the [LBL], [GSB], and [GTO] functions terminate digit entry and enable stack lift.* This method was used in the preceeding example.

   The presence of the stack makes it possible to load more than one variable prior to running a program. Keeping in mind how the stack moves with subsequent operations and how the stack can be manipulated (as with [$x \gtrless y$] and [R↓]), it is possible for you to write a program to use variables which have been keyed into all four registers.

   This is the method used on page 75, where the two numbers were placed into the X- and Y-registers before running the program. In fact, you could store the lower and higher order

* Note, however, that [R/S] is *neutral* with respect to stack lift. There is a complete list of neutral and lift-disabling functions in appendix B.

words (numbers) in the Z- and Y-registers, and the word size in the X-register. Given these values in the stack, the first part of the program could then be:

Keystrokes	
[g] [LBL] A	Word size in X-register.
[HEX]	
[f] [WSIZE]	Stack drops. The higher order word (from Y) is now in the X-register; the lower order word (from Z) is in the Y-register.
⋮	

Direct Entry. Enter the data as needed as the program runs. Write a [R/S] (*run/stop*) instruction into the program where data values are needed so the program will stop execution there. Enter your data, then press [R/S] to restart the program.

Program Memory

The HP-16C has 203 bytes available for data storage and program lines. Program memory is automatically allocated from data storage, seven bytes (for seven lines) at a time. Space for data storage registers is decremented accordingly. Refer to the explanation of memory allocation in section 6.

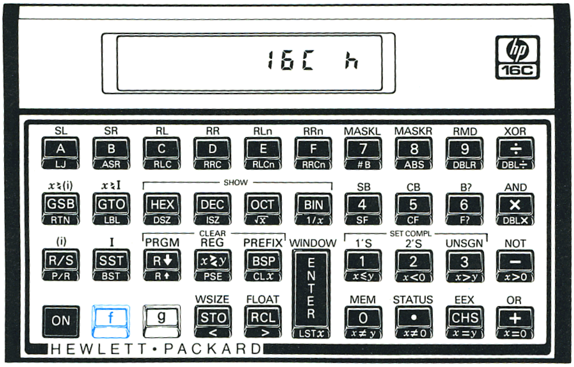
Program Instructions and Keycodes

Each digit, decimal point, and function key is considered an instruction and is stored in one line (occupying one byte) of program memory. Instructions that include prefixes (such as [f], [STO], [GTO], and [LBL]) still occupy only one line.

Each key on the HP-16C keyboard is identified in Program mode by a one- or two-digit "keycode".

The first digit of a two-digit keycode refers to the row (1 to 4 from top to bottom), and the second digit refers to the column (1, 2, ..., 9, 0 from left to right). The keycode for a digit key (including A to F) is simply that digit. For example:

Instruction	Code	
[g] [LBL] 1	001–43,22, 1	
[f] SET COMPL [UNSGN]	002– 42 3	SET COMPL [UNSGN] is "3".



42: Fourth Row, Second Column

Example

The following program utilizes the "double" functions (explained on pages 52–55) to multiply large numbers *of any base* and obtain an exact *decimal* answer at least 38 digits long (that is, up to and not including $10^{19} \times 2^{64}$). The double-sized result is placed into registers X (the most significant digits) and Y.

Since the "double" functions operate internally in *binary*, it is necessary to perform the extra manipulations below (dividing by 10^{19}, the largest exponent of 10 that can be held in one register) to obtain a meaningful *decimal* answer.

Keystrokes	**Display**	
[f] CLEAR [PRGM]		Sets program memory to line 000 but does *not* clear it. (This function only clears in Program mode.)
[g] [P/R]	**000–**	Program mode (**PRGM** annunciator displayed).
[g] [LBL] 1	**001–43,22, 1**	

Keystrokes	Display		
f SET COMPL UNSGN	002–	42 3	Allows a larger possible answer since there is no sign bit.
g DBL×	003–	43 20	Double-multiplies the contents of the X- and Y-registers.
STO 1	004–	44 1	Stores the most significant digits of the result into R_1.
x⪌y	005–	34	
STO 2	006–	44 2	Stores the least significant digits of the result into R_2.
x⪌y	007–	34	
RCL 0	008–	45 0	Recalls (for the divisor) the largest possible power of 10.
g DBLR	009–	43 9	
RCL 2	010–	45 2	Least significant digits of product.
RCL 1	011–	45 1	Most significant digits.
RCL 0	012–	45 0	Divisor.
g DBL÷	013–	43 10	
DEC	014–	24	Ensures that the result is expressed in base 10.
g RTN	015–	43 21	

To run the program, set the word size to 64 and store 10^{19} (the largest possible power of 10 in Unsigned mode) into R_0. Then enter the numbers 12345678987654 and 987654321234567 into the X- and Y-registers.

Keystrokes	Display	
g P/R		Returns to Run mode (no **PRGM** annunciator). Display shows last result.
0 f WSIZE DEC		Sets word size 64, the largest possible word size.
f SET COMPL UNSGN		
10000000 00000000	00000000 .d	
0000 STO 0	00000000 .d	Stores 10^{19} in R_0.
12345678987654		
ENTER	78987654 .d	Enters the two numbers to be multiplied.
987654321234567	21234567 .d	
GSB 1	19326320 .d	Executes program labeled "1"; resulting product is in X- and Y-registers. Most significant word is 1,219,326,320.
f WINDOW 1	12 d.	
x≷y	31035818 .d	Least significant word is 731,267,636,031,035,818. Exact answer is 1,219, 326,320,731,267,636,031, 035,818$_{10}$.
f WINDOW 1	12676360 .d.	
f WINDOW 2	73 d.	

To repeat the program with different values for the multiplicands, just place those numbers in the X- and Y-registers and press GSB 1. (Flag 4 is set during execution of this program because the DBL÷ operation leaves a remainder not equal to zero. However, this is of no significance because the program calculates the remainder in line 009.)

Further Information

Program Labels

Labels in a program (or subroutine) are markers telling the calculator where to begin execution. There are 16 possible labels (the digits 0 through 9 and A through F) for program and subroutine use.

Following a search instruction like GSB *label*, the calculator will search downward in program memory for the corresponding label. If need be, the search will wrap around at the end of program memory and continue at line 001. When it encounters an appropriate label, the search stops and execution begins.

Since the calculator searches in only one direction from its present position, it is possible (though not advisable) to use duplicate program labels. Execution will begin at the first appropriately labeled line encountered.

Unprogrammed Program Stops

Pressing Any Key. Pressing any key will halt program execution. It will *not* halt in the middle of an operation, however.

Error Stops. Program execution is immediately halted when the calculator attempts an improper operation that results in an **Error** display.

To see the line number and keycode of the error-causing instruction (the line at which the program stopped), press any one key to remove the **Error** message, then switch to Program mode.

If an out-of-range condition (including overflow) occurs during a program, flag 5 and the **G** annunciator are set. Program execution will *not* stop.

Nonprogrammable Functions

When the calculator is in Program mode, almost every function on the keyboard can be recorded as an instruction in program memory. The following functions *cannot* be stored as instructions in program memory.

f CLEAR PREFIX	g P/R	SST
f CLEAR PRGM	f MEM	g BST
ON / −	f STATUS	BSP
ON / ·		GTO · *nnn*

Section 8

Program Editing

The HP-16C is equipped with several editing features to help you alter an instruction or program already stored in the calculator.

The Mechanics

Making a program modification of any kind involves two steps: moving to the proper line (the location of the needed change) and making the deletion(s) and/or insertion(s).

Moving to a Line in Program Memory

The Go To (GTO) Instruction. The sequence GTO · *nnn* will move program memory to line number *nnn,* whether pressed in Run mode or Program mode (**PRGM** displayed.) This key sequence is *not* programmable; it is for *manually* finding a specific position in program memory. The number *nnn* must be a three-digit number corresponding to an existing program line in the range $000 \leqslant nnn \leqslant 203$.

The Single Step (SST) Instruction. To step through program memory one line at a time, press SST (*single step*). This (nonprogrammable) function can be used to trace a program *with or without* executing it.

In Program mode, SST will move the memory position forward one line and display that instruction. The instruction is *not* executed. If you hold the key down, the calculator will continuously scroll through the lines in program memory.

In Run mode, SST will display the current program line while the key is held down. When the key is released, the current instruction is executed, the result displayed, and the calculator steps to the next program line to be executed. This function is very useful for tracing the execution of your program, one line at a time.

The Back Step (BST) Instruction. To move one line *backwards* in program memory, press g BST (*back step*) in Program or Run mode. This function is not programmable. BST will scroll (with the BST key held down) in Program mode. Program instructions are *not* executed.

Deleting Program Lines

Deletions of program instructions are made with [BSP] (*back space*) *in Program mode.* Move to the line you want to delete, then press [BSP]. All subsequent program lines will be renumbered to stay in sequence.

Pressing [BSP] in Run mode does not affect program memory, but is used for display clearing. (Refer to page 17.)

Inserting Program Lines

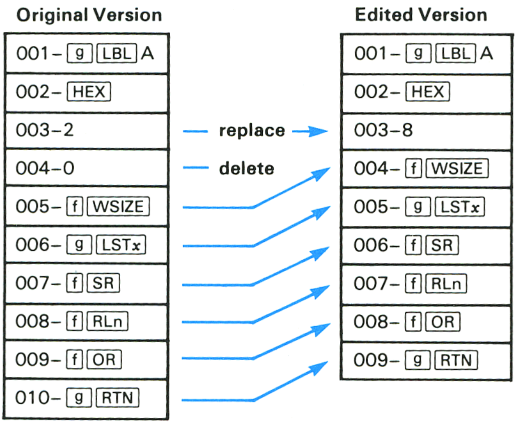
Additions to a program are made by moving to the line *preceding* the point of insertion. Any instruction you key in will be added *following* the line currently in the display. To alter or replace an instruction, first delete it, then add the new version.

If all memory space is occupied, the calculator will not accept any program instruction insertions, and **Error 4** will be displayed.

Example

The description below uses the concatenation program (from page 74, section 7) to illustrate the editing features of the HP-16C. First we'll change the given word size from 32 (20_{16}) to 8 (8_{16}), then single-step through the revised program to monitor its execution.

The editing process is diagrammed below. The given line numbers assume that this program occupies lines 001 to 010 in memory.



Making a deletion or additon of a program line alters the line numbers of those program lines following. By editing the program from the end backwards, you can preserve the original line numbers of parts of the program not yet edited. This allows you to access remaining lines by their original numbers. The editing steps below assume the concatenation program occupies line 001 to 010.

Keystrokes	Display	
g P/R		Program mode. (Line position depends on where calculator position was last.)
GTO · 004 (or use SST)	**004– 0**	Moves position to line 004 (instruction is digit 0).
BSP BSP	**002– 23**	Deletes lines 004 and 003.
8	**003– 8**	Adds a new line 003: 8. Effect of last three steps is to change 20 to 8.

To trace the operation of the revised program, return to Run mode, set word size 4, and place 7_{16} in the Y-register and 6_{16} in the X-register. Then execute the program one step at a time with SST. The concatenated result with twice the word size should be 67_{16}.

Keystrokes	Display	(STATUS: 2–04–0000)
g P/R		Run mode.
HEX		
7 ENTER	**7 h**	Four-bit, least significant word.
6	**6 h**	Four-bit, most significant word.
GTO A	**6 h**	Positions program memory to label "A".
SST (hold)	**001–43,22, A**	Program line 001: label "A".
(release)	**6 h**	X-register contents.

Keystrokes	Display	
SST	002– 23 6 h	Line 002: Hexadecimal mode set.
SST	003– 8 8 h	Line 003: 8.
SST	004– 42 44 6 h	Line 004: f WSIZE. Word size is now 8.
SST	005– 43 36 8 h	Line 005: g LSTx. Brings back word size.
SST	006– 42 b 4 h	Line 006: f SR. Calculates one-half of word size.
SST	007– 42 E 60 h	Line 007: f RLn. Number shifted left four bits.
SST	008– 42 40 67 h	Line 008: f OR. Eight-bit concatenated word.
SST	009– 43 21 67 h	Final instruction (return).

Further Information

Line Position

The calculator's position in program memory does not change when it is shut off or when Program/Run modes are changed. Therefore, if the calculator shuts itself off, you need only turn it on and switch to Program mode (the calculator always "wakes up" in Run mode) to be back where you were.

The calculator cannot move to a program line that does not contain an instruction (is not yet "created"). When you use SST, the calculator will "wrap around" to line 000 after encountering the end of current program memory.

Initializing Calculator Status

The contents of storage registers and the status of calculator settings will affect a program if the program uses those registers or depends on a certain status setting. If the current status is incorrect for the program being run, you will get incorrect results. Therefore, it is wise to initialize the calculator—such as clear registers and set relevant modes—either just prior to running a program or within the program itself. Note that any status condition set by a program will also affect any subsequent programs.

Functions used to initialize conditions in the calculator are: the "CLEAR" functions, the "SET COMPL" functions, the number base modes, Floating-Point Decimal mode, [WSIZE], [SF] and [CF].

Section 9

Program Branching and Controls

The branching capabilities of the HP-16C include simple and conditional branching—in which program execution depends on a certain condition—and subroutines. The use of the Index register, which can hold a counter value, greatly enhances the calculator's branching and looping control.

The Mechanics

Branching

The Go To ($\boxed{\text{GTO}}$) Instruction. Simple branching—that is, unconditional branching—is carried out with the instruction $\boxed{\text{GTO}}$ *label*. In a running program, $\boxed{\text{GTO}}$ {0 to 9, A to F, $\boxed{\text{I}}$} will transfer execution to the designated program or routine.* (It is not possible to branch to a line *number*.) The calculator searches forward in memory for the indicated label, wrapping around to line 001 and beyond if necessary.

$\boxed{\text{GTO}}$ *label* can also be used in Run mode (that is, from the keyboard) to move to a labeled position in program memory. No execution occurs.

Go To Subroutine ($\boxed{\text{GSB}}$). Transfer to a labeled subroutine is executed by the sequence $\boxed{\text{GSB}}$ *label*. Program execution automatically transfers back to the main program when a $\boxed{\text{RTN}}$ instruction is encountered.† Subroutine execution is described later in this section (page 94).

* $\boxed{\text{GTO}}$ $\boxed{\text{I}}$ and $\boxed{\text{GSB}}$ $\boxed{\text{I}}$ are *abbreviated key sequences* (no $\boxed{\text{f}}$ keystroke necessary), as explained on page 68.

† If no subroutine return is pending, a $\boxed{\text{RTN}}$ instruction halts execution and returns the calculator to the top of program memory.

Indirect Branching Using the Index Register

By placing an *index value* in R_I (the Index register), you can *indirectly* branch to a location (GTO I) and *indirectly* call a subroutine (GSB I).

These functions will transfer execution to the label that corresponds to the *absolute value of the number in the Index register* according to the table below.* (In Floating Point mode, only the integer portion of the number in R_I is used.) There are 16 possible labels: 0 to 9 and A to F.

For instance, if the Index register contains -14_{10} (STATUS : **2–08–0000**), then a GTO I instruction would transfer program execution to LBL E ($|-14_{10}| = E_{16}$).

Indirect Branching

If R_I Contains:		GTO I or GSB I will transfer execution to:
0	(0_{16})	LBL 0
⋮	⋮	⋮
9	(9_{16})	LBL 9
10	(A_{16})	LBL A
⋮	⋮	⋮
15	(F_{16})	LBL F

Conditional Tests

Another way to alter the sequence of program execution is by a *conditional test,* a true/false test which compares the number in the X-register either to zero or to the number in the Y-register. (In 1's Complement mode, these tests consider -0 equal to 0.)

The HP-16C provides eight different tests (all g -shifted functions):

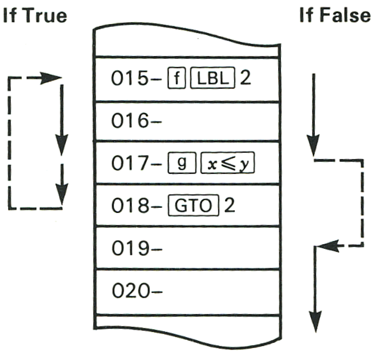
$x \leq y$ $x < 0$ $x > y$ $x > 0$ $x \neq y$ $x \neq 0$ $x = y$ $x = 0$

Following a conditional test, program execution follows the "Do if True" rule: it proceeds sequentially if the condition is true, and it *skips* one *instruction if the condition is false.* A GTO instruction is

*The absolute value is computed using a word size of 68 and the current complement mode.

often placed right after a conditional test, making it a *conditional branch*; that is, the GTO branch is executed only if the test condition is met.

Program Execution After Test



"Do if True"

Testing for Set Flags and Set Bits

Additional tests for conditional branching are provided by the F? (*flag set?*) and B? (*bit set?*) functions. Following these instructions—as with the other conditional tests—program execution follows the "Do if True" rule (illustrated above): it proceeds sequentially if the flag (or bit) is set, and skips one line if the flag (or bit) is clear.

As discussed in section 3, the flag numbers and their meanings are:

0, 1, 2	User flags (used to control programming).
3	Controls display of leading zeros.
4	Carry or borrow condition.
5	Out-of-range condition.

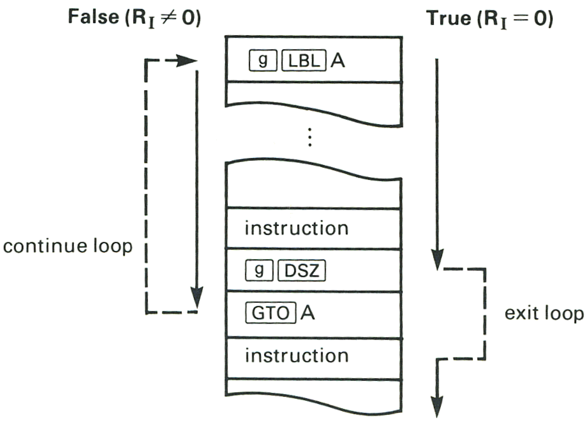
Although flags 4 and 5 are set automatically by the calculator, the user can also set them. If you set flag 4, the carry bit is set to 1. Refer to section 3 (page 36) for a full discussion of setting flags, and section 4 (page 50) for a full discussion of setting bits.

Loop Control with Counters: DSZ and ISZ

The DSZ (*decrement and skip next line if counter equals zero*) and ISZ (*increment and skip if zero*) functions can control loop execution by referencing and adjusting (incrementing/decrementing) a counter value in the Index register. Then, when that counter value reaches zero, program execution skips one line.

Each time one of these functions is encountered in a running program, the given counter value in the Index register is either *decremented* (DSZ) or *incremented* (ISZ) by *one*. If the resulting value equals zero, the next instruction is skipped. This allows exit from a loop if the skipped line was a branch into a loop.

Conditional Branch with DSZ or ISZ



"Skip if True"

The value in R_I is interpreted according to the current complement mode. It can be positive or negative, in integer or floating-point format. The instructions DSZ and ISZ do not affect the status of the carry and out-of-range flags.

Example

A "checksum" routine can be used to test the integrity of stored data values. Using #B you can determine the sum of a bit pattern and then compare that sum to the sum of the same bit pattern at a later time.

The following program sums all the bits in the bit pattern in a given storage register, yielding a checksum. The contents of storage registers R_A through R_1 are sequentially checksummed. As the bits are summed, they are added to the updated, double-sized checksum being held in registers Y and Z. This is what the stack contains just before line 012:

T	
Z	Current checksum: most significant word.
Y	Current checksum: least significant word.
X	Number whose bits will be summed and added to the current double-word contents in Y and Z.

The resulting checksum will be placed in registers X and Y.

This program uses DSZ to decrement a register pointer in the Index register and to control conditional loop branching.

Keystrokes	Display	
g P/R	000-	
f CLEAR PRGM	000-	
g LBL D	001-43,22, d	
f SET COMPL UNSGN	002- 42 3	Unsigned mode for adding bits.
4	003- 4	
f WSIZE	004- 42 44	Word size four bits.
HEX	005- 23	
A	006- A	
STO I	007- 44 32	Stores number of top register (R_A) in R_I.
0	008- 0	
ENTER	009- 36	Initializes checksum to 0.
g LBL 0	010-43,22, 0	Start of summing loop. (Enables stack lift.)
RCL (i)	011- 45 31	Recalls contents of current register whose number is stored in R_I.

Keystrokes	Display	
g #B	012– 43 7	Sums the bits in the X-register.
+	013– 40	Adds this sum to least significant part of current checksum. Might set carry flag.
$x \gtrless y$	014– 34	Brings most significant part of current checksum into X.
0	015– 0	Places 0 in X.
g RLC	016– 43 C	Places a 1 into X if a carry was generated in the preceding addition.
+	017– 40	Adds carry bit to most significant part of checksum.
$x \gtrless y$	018– 34	Returns least significant part of checksum to X.
g DSZ	019– 43 23	Decrements the current register number stored in R_I.
GTO 0	020– 22 0	If register number in R_I is not yet zero, then continues with loop.
g RTN	021– 43 21	

Now, calculate an updated checksum (bit summation) given the following 4-bit hexadecimal values in R_1 through R_A:

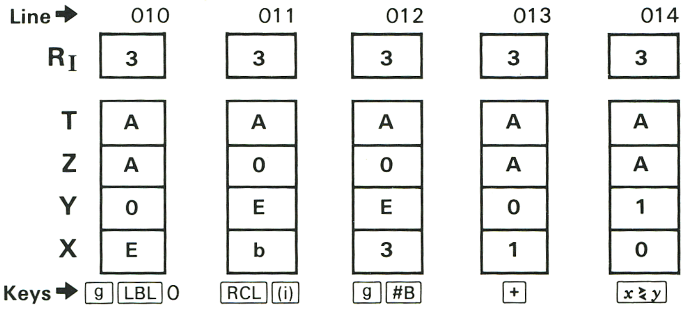
R_1: A	R_3: B	R_5: 3	R_7: A	R_9: D
R_2: 7	R_4: 1	R_6: D	R_8: 2	R_A: 6

Keystrokes	Display	(STATUS : 0–04–0000)
g P/R		Returns to Run mode.
HEX		
A STO 1	A h	Store the above values in R_1 through R_A.
⋮	⋮	
6 STO A	6 h	

Keystrokes	Display	
GSB D	6 h	Least significant bits of double-word checksum.
$x \gtrless y$	1 h	Most significant bits: sum of bits in above pattern is 16_{16} or 22_{10}.

When writing or analyzing a program, it is often helpful to use a diagram showing the contents of the stack before and after each instruction. The stack diagrams below show the movement of the stack contents in the loop portion (LBL 0: lines 010 through 019) of the above program.

On the eighth iteration of this loop, the carry is set in step 013 when the checksum for the contents of R_3 is added to the prior checksum (equalling E_{16}), thereby exceeding a single word size. This iteration is shown here. (The A in the T- and Z-registers is a remnant from lines 006 and 007.)


Line ➡ 010 011 012 013 014
R_I 3 3 3 3 3
T A A A A A
Z A 0 0 A A
Y 0 E E 0 1
X E b 3 1 0
Keys ➡ g LBL 0 | RCL (i) | g #B | + | x≷y

Line 012 does a checksum of the contents of the register currently addressed by R_I, and line 013 adds this checksum to the least significant part of the checksum. Lines 014 to 017 add in the carry bit from the previous add to the most significant word of the checksum.

Line ➡	015	016	017	018	019
R_I	3	3	3	3	2
T	A	A	A	A	A
Z	1	1	A	A	A
Y	0	0	1	1	1
X	0	1	1	1	1
Keys ➡	0	g RLC	+	x≷y	g DSZ

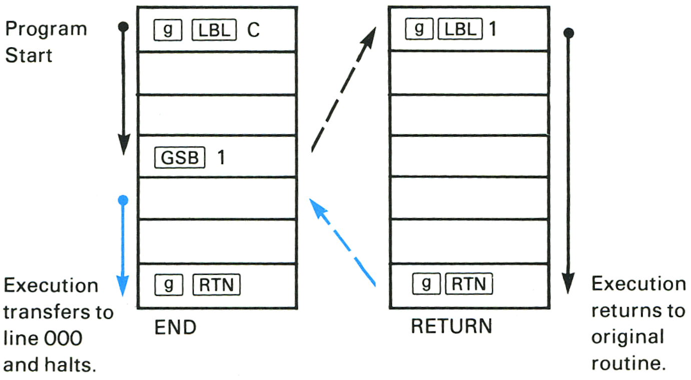
Line 019 decrements the storage register address in R_I so the next loop will sum bits from a new register.

Further Information

Subroutines

Execution. The *go to subroutine* instruction, GSB *label*, is a simple branch with a special trait: it establishes a *pending return* condition. Program execution, once transferred to the line containing the designated label, continues *until the first subsequent* RTN *instruction is encountered*—at which point execution transfers back to the instruction immediately following the corresponding GSB instruction.

Subroutine Execution


Program Start
g LBL C
GSB 1
g RTN
END
Execution transfers to line 000 and halts.
g LBL 1
g RTN
RETURN
Execution returns to original routine.

Nesting. Subroutine "nesting"—the execution of a subroutine within a subroutine—is limited to a nest of subroutine returns four levels deep (not counting the main program level).

If you attempt to call a subroutine nested more than four levels deep, the calculator will halt and display **Error 5** when it encounters the GSB at the fifth level. Note that there is no limitation (other than memory size) to the number of nonnested subroutines or *sets* of nested subroutines that you may use.

Program Versus Keyboard Use of GSB.

The GSB key is used for two different functions: 1) to execute programs from the keyboard, and 2) to call subroutines within a running program. Note that when GSB is used from the keyboard to execute a program, it does not establish pending subroutine returns. When a RTN is subsequently encountered, program execution halts and returns to line 000—unless there were intervening programmed GSB instructions (subroutine calls).

Appendix A

Errors and Flags

Error Conditions

If you attempt an operation containing an improper parameter—such as referencing a nonexistent flag number—the display shows **Error** and a number. To clear an error display, press any one key. This also restores the display that existed prior to the error message. Below is a list of error conditions by number.

Error 0: Improper Mathematical Operation

$\boxed{\div}$, where $x = 0$.
$\boxed{\text{RMD}}$, where $x = 0$.
$\boxed{\text{DBL}\div}$, where:

- $x = 0$.
- the quotient exceeds a single word size.

$\boxed{\text{DBLR}}$, where:

- $x = 0$.
- the *quotient* exceeds 64 bits.

$\boxed{\sqrt{x}}$, where $x < 0$.
$\boxed{1/x}$, where $x = 0$ (Floating-Point Decimal mode only).

Error 1: Improper Flag, Window, $\boxed{\text{FLOAT}}$, or $\boxed{\text{GTO}}\boxed{\cdot}$ Number

Attempted flag number greater than 5.
Attempted window number greater than 7.
Attempted $\boxed{\text{FLOAT}}$ number greater than 9.
Attempted $\boxed{\text{GTO}}\boxed{\cdot}$ digit greater than 9.

Error 2: Improper Bit Number

$\boxed{\text{MASKL}}$ or $\boxed{\text{MASKR}}$, where $|x| >$ current word size.
$\boxed{\text{RLn}}$ or $\boxed{\text{RRn}}$, where $|x| >$ current word size.
$\boxed{\text{RLCn}}$ or $\boxed{\text{RRCn}}$, where $|x| >$ (current word size + 1).
$\boxed{\text{SB}}$, $\boxed{\text{CB}}$, or $\boxed{\text{B?}}$, where $|x| \geqslant$ current word size.
$\boxed{\text{WSIZE}}$, where $|x| > 64$.

Error 3: Improper Register Number

Storage register named is nonexistent.

Error 4: Improper Label or Line Number Call

Label or line number called is nonexistent. Attempted to load more than 203 program lines.

Error 5: Subroutine Level Too Deep

Subroutine nested more than four deep.

Error 6: Invalid Register Contents

In Floating-Point Decimal mode, an attempt was made to recall the contents of a storage register (including R_I) whose contents are not in floating-point decimal format.

In Floating-Point Decimal mode, $\boxed{\text{DSZ}}$ or $\boxed{\text{ISZ}}$ was used when the contents of R_I were not in floating-point format.

Error 9: Service

Self-test discovered circuitry problem, or wrong key pressed during key test. Refer to appendix C.

Pr Error (*Power Error*)

Continuous Memory interrupted and reset because of power failure.

Functions That Affect Flags

Two important flags, the carry flag (4) and the out-of-range flag (5), are affected (set or cleared) by certain arithmetic and shifting functions *in Integer mode*. These functions are listed below.

× = sets or clears 0 = always clears — = no effect

Function	Effect On		Registers Used:	
	Carry (4)	Out-Of-Range (5)	Operand(s)	Result
+	×	×*	X, Y	X
−	×	×*	X, Y	X
×	—	×*	X, Y	X
÷	×	×*	X, Y	X
$\sqrt{x}$	×	—	X	X
DBL×	—	0	X, Y	X, Y
DBL÷	×	0	X, Y, Z	X
ABS	—	×	X	X
CHS	—	×	X	X
SL	×	—	X	X
SR	×	—	X	X
ASR	×	—	X	X
RL	×	—	X	X
RR	×	—	X	X
RLC	×	—	X	X
RRC	×	—	X	X
RLn	×	—	X, Y	X
RRn	×	—	X, Y	X
RLCn	×	—	X, Y	X
RRCn	×	—	X, Y	X

* Also in Floating-Point Decimal mode.

Appendix B

Classes of Operations

Operations Terminating Digit Entry

Most operations on the calculator, whether executed as instructions in a program or pressed from the keyboard, terminate digit entry. This means that the calculator knows that any digits you key in after any of these operations are part of a new number.

The *digit entry* operations do *not* terminate digit entry. They are:

[0] through [9]	[·]	[CHS] in Floating Point mode
[A] through [F]	[EEX]	[BSP]

Operations Affecting Stack Lift

There are three types of operations on the calculator based on how they affect stack lift. These are stack-disabling operations, stack-enabling operations, and neutral operations.

Disabling Operations

There are two stack lift-disabling functions on the HP-16C. These operations disable stack lift so that a number keyed in *after* one of these operations writes over the current number in the X-register and the stack does not lift. These operations are:

[ENTER] [CLx]*

* And [BSP] if digit entry has been terminated.

Neutral Operations

Many operations are neutral—they do not alter the previous status of the stack lift, whether enabled or disabled.

The following operations are neutral with regard to stack lift.*

HEX	WINDOW	MEM	PSE
DEC	<, >	STATUS	R/S
OCT	GTO · *nnn*	CLEAR PREFIX	SST
BIN	P/R	CLEAR REG	BST

SHOW { HEX , DEC , OCT , BIN }

SET COMPL { 1′s , 2′s , UNSGN }

FLOAT (when in Floating Point mode)

Enabling Operations

Most calculator operations are stack lift-enabling. A number keyed in *after* one of these operations will lift the stack (because the stack has been "enabled" to lift).

All operations not listed above as disabling or neutral are enabling.

Operations Affecting the LAST X Register

The following operations save x in the LAST X register:

−	RMD	XOR	ABS	LJ	RLCn	RLC	SB	RRn
+	DBL×	NOT	$\sqrt{x}$	ASR	RRCn	RRC	CB	#B
×	DBL÷	OR	$1/x$	RL	SL	MASKL	B?	CHS †
÷	DBLR	AND	WSIZE	RR	SR	MASKR	RLn	

Executing the FLOAT function from Integer mode clears the LAST X register.

* Also: the digit entry operations are neutral before termination of digit entry.

† In Integer mode.

Operations Affecting Scrolling

The following operations *do not reset scrolling,* that is, they do not restore window 0 to the display. All other operations do reset scrolling.

$\boxed{<}$, $\boxed{>}$	$\boxed{\text{P/R}}$	$\boxed{\text{PSE}}$	$\boxed{\text{SST}}$
$\boxed{\text{ON}}$	$\boxed{\text{LBL}}$	$\boxed{\text{R/S}}$	$\boxed{\text{BST}}$
$\boxed{\text{STO}}$	$\boxed{\text{RTN}}$	$\boxed{\text{GSB}}$	$\boxed{\text{DSZ}}$
$\boxed{\text{MEM}}$	$\boxed{\text{SF}}$, $\boxed{\text{CF}}$, $\boxed{\text{F?}}$	$\boxed{\text{GTO}}$	$\boxed{\text{ISZ}}$
$\boxed{\text{STATUS}}$			

SHOW {(current base)} CLEAR {$\boxed{\text{PRGM}}$, $\boxed{\text{REG}}$, $\boxed{\text{PREFIX}}$}

$\boxed{x \leqslant y}$, $\boxed{x < 0}$, $\boxed{x > y}$, $\boxed{x > 0}$, $\boxed{x \neq y}$, $\boxed{x \neq 0}$ $\boxed{x = y}$, $\boxed{x = 0}$

Prefix Keys

The prefix keys on the HP-16C are:

$\boxed{\text{f}}$	$\boxed{\text{STO}}$	$\boxed{\text{SF}}$	$\boxed{\text{LBL}}$	$\boxed{\text{GTO}}$ $\boxed{\cdot}$
$\boxed{\text{g}}$	$\boxed{\text{RCL}}$	$\boxed{\text{CF}}$	$\boxed{\text{GSB}}$	
$\boxed{\text{FLOAT}}$	$\boxed{\text{WINDOW}}$	$\boxed{\text{F?}}$	$\boxed{\text{GTO}}$	

Operations Not Active in Floating-Point Decimal Mode

$\boxed{\text{SL}}$	$\boxed{\text{RR}}$	$\boxed{\text{MASKL}}$	$\boxed{\text{SB}}$	$\boxed{\text{NOT}}$
$\boxed{\text{LJ}}$	$\boxed{\text{RRC}}$	$\boxed{\text{MASKR}}$	$\boxed{\text{CB}}$	$\boxed{\text{OR}}$
$\boxed{\text{SR}}$	$\boxed{\text{RLn}}$	$\boxed{\text{RMD}}$	$\boxed{\text{B?}}$	$\boxed{<}$
$\boxed{\text{ASR}}$	$\boxed{\text{RLCn}}$	$\boxed{\text{DBLR}}$	$\boxed{\text{WSIZE}}$	$\boxed{>}$
$\boxed{\text{RL}}$	$\boxed{\text{RRn}}$	$\boxed{\text{DBL}\div}$	$\boxed{\text{XOR}}$	$\boxed{\text{A}}$ to $\boxed{\text{F}}$
$\boxed{\text{RLC}}$	$\boxed{\text{RRCn}}$	$\boxed{\text{DBL}\times}$	$\boxed{\text{AND}}$	

SHOW {$\boxed{\text{HEX}}$, $\boxed{\text{DEC}}$, $\boxed{\text{OCT}}$, $\boxed{\text{BIN}}$}

The only operations not active in Integer mode are $\boxed{\cdot}$, $\boxed{\text{EEX}}$, and $\boxed{1/x}$.

Appendix C

Battery, Warranty, and Service Information

Batteries

The HP-16C is powered by three batteries. In "typical" use, the HP-16C has been designed to operate six months or more on a set of alkaline batteries. The batteries supplied with the calculator are alkaline, but silver-oxide batteries (which should last twice as long) can also be used.

A set of three fresh alkaline batteries will provide at least 80 hours of *continuous* program running (the most power-consuming kind of calculator use*). A set of three fresh silver-oxide batteries will provide at least 180 hours of *continuous* program running. If the calculator is being used to perform operations other than running programs, it uses much less power. When only the display is on—that is, if you are not pressing keys or running programs—very little power is consumed.

If the calculator remains turned off, a set of fresh batteries will preserve the contents of Continuous Memory for as long as the batteries would last outside of the calculator—at least 1½ years for alkaline batteries or at least 2 years for silver-oxide batteries.

The actual lifetime of the batteries depends on how often you use the calculator, whether you use it more for running programs or more for manual calculations, and which functions you use.*

The batteries supplied with the calculator, as well as the batteries listed below for replacement, are *not* rechargeable.

*Power consumption in the HP-16C depends on the mode of calculator use: off (with Continuous Memory preserved); idle (with only the display on); or "operating" (running a program, performing a calculation, or having a key pressed). While the calculator is turned on, typical calculator use is a mixture of idle time and "operating" time. Therefore, the actual lifetime of the batteries depends on how much time the calculator spends in each of the three modes.

WARNING

Do not attempt to recharge the batteries; do not store batteries near a source of high heat; do not dispose of batteries in fire. Doing so may cause the batteries to leak or explode.

The following batteries are recommended for replacement in your HP-16C. Not all batteries are available in all countries.

Alkaline	**Silver Oxide**
Eveready A76 *	Eveready 357 *
UCAR A76	UCAR 357
RAY-O-VAC RW82	RAY-O-VAC RS76 or RW42
National or Panasonic LR44	Duracell MS76 or 10L14
Varta 4276	Varta 541

Low-Power Indication

An asterisk (∗) flashing in the lower left corner of the display when the calculator is on signifies that the available battery power is running low.

With alkaline batteries installed:

- The calculator can be used for at least 2 hours of continuous program running after the asterisk first appears. †
- If the calculator remains turned off, the contents of its Continuous Memory will be preserved for at least 1 month after the asterisk first appears.

With silver-oxide batteries installed:

- The calculator can be used for at least 15 minutes of continuous program running after the asterisk first appears. †
- If the calculator remains turned off, the contents of its Continuous Memory will be preserved for at least 1 week after the asterisk first appears.

* Not available in the United Kingdom or Republic of Ireland.

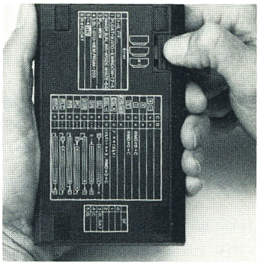
† Note that this time is the minimum available for *continuous program running*—that is, while continuously "operating" (as described in the footnote on the preceding page). If you are using the calculator for manual calculations—a mixture of the idle and "operating" modes—the calculator can be used for a much longer time after the asterisk first appears.

Installing New Batteries

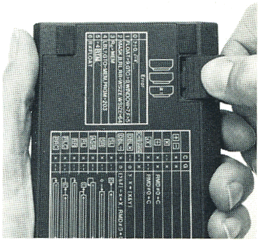
The contents of the calculator's Continuous Memory are preserved for a short time while the batteries are out of the calculator (provided that you turn off the calculator before removing the batteries). This allows you ample time to replace the batteries without losing data or programs. If the batteries are left out of the calculator for an extended period, the contents of Continuous Memory may be lost.

To install new batteries, use the following procedure:

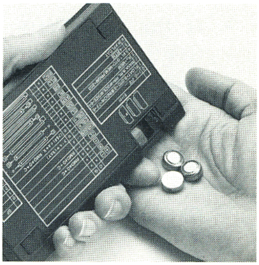
1. Be sure the calculator is off.
2. Holding the calculator as shown, press outward on the battery compartment door until it opens slightly.



3. Grasp the outer edge of the battery compartment door, then tilt it up and out of the calculator.



Note: In the next two steps, be careful not to press any keys while batteries are not in the calculator. If you do so, the contents of Continuous Memory may be lost and keyboard control may be lost (that is, it will not respond to keystrokes).

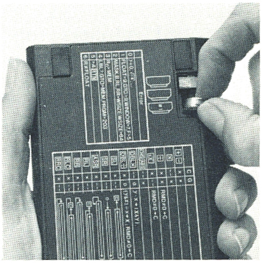


4. Turn the calculator over and gently shake, allowing the batteries to fall into the palm of your hand.

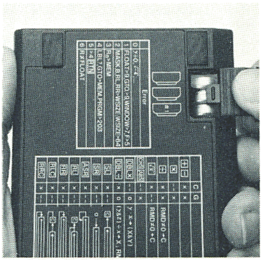
CAUTION

In the next step, replace *all three* batteries with fresh ones. If you leave an old battery inside, it may leak. Furthermore, be careful not to insert the batteries backwards. If you do so, the contents of Continuous Memory may be lost and the batteries may be damaged.

5. Holding open the two plastic flaps shielding the battery compartment, insert three new batteries. The batteries should be positioned with their flat sides (the sides marked +) facing *toward* the nearby rubber foot, as shown in the illustration on the calculator case.

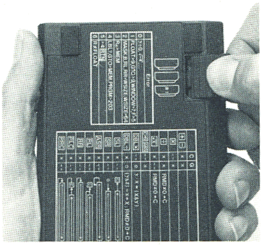


6. Insert the tab of the battery compartment door into the slot in the calculator case.



7. Lower the battery compartment door until it is flush with the case, then push the door inward until it is tightly shut.

8. Turn the calculator on. If for any reason Continuous Memory has been reset (that is, its contents have been lost), the display will show **Pr Error**. Pressing any key will clear this message from the display.



Verifying Proper Operation (Self-Tests)

If it appears that the calculator will not turn on or otherwise is not operating properly, review the following steps.

For a calculator that does *not* respond to keystrokes:

1. Press the [D] and [ON] keys simultaneously, then release them. This will alter the contents of the X-register, so clear the X-register afterward.
2. If the calculator still does not respond to keystrokes, remove and reinsert the batteries. Make sure the batteries are properly positioned in the compartment.
3. If the calculator still does not respond to keystrokes, leave the batteries in the compartment and short both battery terminals together. (Fold back the plastic flaps to expose the terminals, which are the metal strips on either side of the battery compartment.) *Only momentary contact is required.* After you do this, the contents of Continuous Memory will be lost, and you may need to press the [ON] key more than once to turn the calculator back on.
4. If the calculator still does not turn on, install fresh batteries. If there is still no response, the calculator requires service.

For a calculator that *does* respond to keystrokes:

1. With the calculator off, hold down the [ON] key and press [×].
2. Release the [ON] key, then release the [×] key. This initiates a complete test of the calculator's electronic circuitry. If everything is working correctly, within about 15 seconds (during which the word **running** flashes) the display should show **-8,8,8,8,8,8,8,8,8,8,** and all of the status indicators (except the ∗ low-power indicator) should turn on.* If the display shows **Error 9**, goes blank, or otherwise does not show the proper result, the calculator requires service.†

* The status indicators turned on at the end of this test include some that normally are not displayed on the HP-16C.

† If the calculator displays **Error 9** as a result of the [ON]/[×] test or the [ON]/[+] test, but you wish to continue using your calculator, you should reset Continuous Memory as described on page 20.

Note: Tests of the calculator's electronics are also performed if the [+] key or the [÷] key is held down when [ON] is released.*† These tests are included in the calculator to be used in verifying that it is operating properly during manufacture and service.

If you had suspected that the calculator was not working properly but the proper display was obtained in step 2, it is likely that you made an error in operating the calculator. We suggest you reread the section in this handbook applicable to your calculation. If you still experience difficulty, write or telephone Hewlett-Packard at an address or phone number listed under Service (page 110).

* If the calculator displays **Error 9** as a result of the [ON]/[×] test or the [ON]/[+] test, but you wish to continue using your calculator, you should reset Continuous Memory as described on page 20.

† The [ON]/[+] combination initiates a test that is similar to that described above, but continues indefinitely. The test can be terminated by pressing any key, which will halt the test within 15 seconds. The [ON]/[÷] combination initiates a test of the keyboard and the display. When the [ON] key is released, certain segments in the display will be lit. To run the test, the keys are pressed in order from left to right along each row, from the top row to the bottom row. As each key is pressed, different segments in the display are lit. If the calculator is operating properly *and all the keys are pressed in the proper order,* the calculator will display **16** after the last key is pressed. (The [ENTER] key should be pressed both with the third-row keys and with the fourth-row keys.) If the calculator is not working properly, *or if a key is pressed out of order,* the calculator will display **Error 9**. *Note that if this error display results from an incorrect key being pressed, this does* not *indicate that your calculator requires service.* This test can be terminated by pressing any key out of order (which will, of course, result in the **Error 9** display). Both the **Error 9** display and the **16** display can be cleared by pressing any key.

Limited One-Year Warranty

What We Will Do

The HP-16C is warranted by Hewlett-Packard against defects in material and workmanship for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is automatically transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. **ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY.** Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. **IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES.** Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty, please contact:

- In the United States:

Hewlett-Packard
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330
Telephone: (503) 758-1010
Toll-Free Number: (800) 547-3400 (except in Oregon, Hawaii, and Alaska)

- In Europe:

Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2
Geneva
Switzerland
Telephone: (022) 83 81 11

Note: Do *not* send calculators to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do *not* send calculators to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard calculator products normally are repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could vary depending upon the time of year and work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for handheld and portable calculator products is located in Corvallis, Oregon:

Hewlett-Packard Company
Corvallis Division Service Department
P.O. Box 999/1000 N.E. Circle Blvd.
Corvallis, Oregon 97330, U.S.A.
Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your calculator.

AUSTRIA
HEWLETT-PACKARD GmbH
Wagramerstr.-Lieblgasse
A 1220 VIENNA
Telephone: (222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Boulevard de la Woluwe 100
Woluwelaan
B 1200 BRUSSELS
Telephone: (2) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK 3460 BIRKEROD (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF 02100 ESPOO 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
S.A.V. Calculateurs de Poche
Division Informatique Personnelle
F 91947 LES ULIX CEDEX
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D 6000 FRANKFURT 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I 20063 CERNUSCO SUL NAVIGLIO (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
NL 1181 KK AMSTELVEEN (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N 1345 OESTERAAS (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E MADRID 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Enighetsvagen 3
Box 205 02
S 161 BROMMA 20 (Stockholm)
Telephone: (8) 730 05 50

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Allmend 2
CH 8967 WIDEN
Telephone: (057) 50111

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
Winnersh, Wokingham
GB BERKSHIRE RG11 5AR
Telephone: (734) 784774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP calculator products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under Obtaining Repair Service in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Service Repair Charge

There is a standard repair charge for out-of-warranty repairs. The repair charges include all labor and materials. In the United States, the full charge is subject to the customer's local sales tax. In

European countries, the full charge is subject to Value Added Tax (VAT) and similar taxes wherever applicable. All such taxes will appear as separate items on invoiced amounts.

Calculator products damaged by accident or misuse are not covered by the fixed repair charges. In these situations, repair charges will be individually determined based on time and material.

Service Warranty

Any out-of-warranty repairs are warranted against defects in materials and workmanship for a period of 90 days from date of service.

Shipping Instructions

Should your unit require service, return it with the following items:

- A completed Service Card, including a description of the problem.
- A sales receipt or other proof of purchase date if the one-year warranty has not expired.

The product, the Service Card, a brief description of the problem, and (if required) the proof of purchase should be packaged in the original shipping case or other adequate protective packaging to prevent in-transit damage. Such damage is not covered by the one-year limited warranty; Hewlett-Packard suggests that you insure the shipment to the service center. The packaged unit should be shipped to the nearest Hewlett-Packard designated collection point or service center. Contact your dealer for assistance. (If you are not in the country where you originally purchased the unit, refer to International Service Information, above.)

Whether the unit is under warranty or not, it is your responsibility to pay shipping charges for delivery to the Hewlett-Packard service center.

After warranty repairs are completed, the service center returns the unit with postage prepaid. On out-of-warranty repairs in the United States and some other countries, the unit is returned C.O.D. (covering shipping costs and the service charge).

Further Information

Service contracts are not available. Calculator product circuitry and design are proprietary to Hewlett-Packard, and service manuals are not available to customers.

Should other problems or questions arise regarding repairs, please call your nearest Hewlett-Packard service center.

Programming and Applications Assistance

Should you need technical assistance concerning programming, applications, etc., call Hewlett-Packard Customer Support at (503) 757-2000. This is not a toll-free number, and we regret that we cannot accept collect calls. As an alternative, you may write to:

Hewlett-Packard
Corvallis Division Customer Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Dealer and Product Information

For dealer locations, product information, and prices, please call (800) 547-3400. In Oregon, Alaska, or Hawaii, call (503) 758-1010.

Temperature Specifications

- Operating: 0° to 55° C (32° to 131° F)
- Storage: -40° to 65° C (-40° to 149° F)

Potential for Radio and Television Interference (for U.S.A. Only)

The HP-16C generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If your HP-16C does cause interference to radio or television reception, you are encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the calculator with respect to the receiver.
- Move the calculator away from the receiver.

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful: *How to Identify and Resolve Radio-TV Interference Problems*. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4.

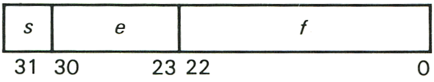
Appendix D

Programs for Format Conversion

Different computing machines use various formats for representing numbers. Consequently, it is often necessary to convert numbers from one format to another. This appendix provides two programs to convert numbers between the proposed IEEE standard floating-point binary format and the floating-point decimal format used in the HP-16C. *

Formats

The proposed IEEE single-precision, floating-point binary format is:



in a 32-bit format with 1-bit sign s,
8-bit biased exponent e, and
23-bit fraction f.

The value v of a number x (the contents of the X-register) is interpreted as follows:

(a) If $e = 255$ and $f \neq 0$, then $v = \text{NaN}$ (*not a number*).

(b) If $e = 255$ and $f = 0$, then $v = (-1)^s \infty$.

(c) If $0 < e < 255$, then $v = (-1)^s\, 2^{(e-127)}\, (1.f)$.

(d) If $e = 0$ and $f \neq 0$, then $v = (-1)^s\, 2^{(-126)}\, (0.f)$.

(e) If $e = 0$ and $f = 0$, then $v = (-1)^s\, 0$.

In Floating-Point Decimal mode on the HP-16C, the following conventions are used:

* The standard for the floating-point binary format is a proposal of the IEEE Computer Society's Floating-Point Committee, Task 754. It has been set forth in *Computer*, March 1981, pages 51–62.

IEEE Number	X-Register	Carry (Flag 4)	Out-of-Range (Flag 5)
0	0	0	0
−0	0	1	0
±∞	$\pm 9.999999999 \times 10^{99}$	1	1
Other Numbers	As defined above under (c) and (d)	0	0
Not a Number	$(-1)^s (0.f)\ 2^{23}$	1	0

Program: Conversion from IEEE Format to HP-16C Format

The following program converts a number from IEEE single-precision, floating-point binary format to floating-point decimal format.

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
g LBL B	001–43,22, b	x⪌y	018– 34
HEX	002– 23	8	019– 8
f SET COMPL 2's	003– 42 2	f MASKL	020– 42 7
2	004– 2	g x=y	021– 43 49
0	005– 0	GTO 4	022– 22 4
f WSIZE	006– 42 44	R↓	023– 33
f SL	007– 42 A	g x=0	024– 43 40
ENTER	008– 36	GTO 3	025– 22 3
ENTER	009– 36	x⪌y	026– 34
g x=0	010– 43 40	1	027– 1
GTO 2	011– 22 2	8	028– 8
1	012– 1	f SB	029– 42 4
8	013– 8	g LBL 1	030–43,22, 1
f MASKR	014– 42 8	g F? 4	031–43, 6, 4
f AND	015– 42 20	CHS	032– 49
f XOR	016– 42 10	x⪌y	033– 34
g LSTx	017– 43 36	8	034– 8

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
f RLn	035– 42 E	g CLx	051– 43 35
9	036– 9	g $x \neq y$	052– 43 0
7	037– 7	GTO 5	053– 22 5
−	038– 30	1	054– 1
g CF 4	039–43, 5, 4	4	055– 4
g LBL 2	040–43,22, 2	5	056– 5
f FLOAT ·	041–42,45,48	ENTER	057– 36
g RTN	042– 43 21	g LBL 5	058–43,22, 5
g LBL 3	043–43,22, 3	$x \gtrless y$	059– 34
1	044– 1	g F? 4	060–43, 6, 4
8	045– 8	CHS	061– 49
f SB	046– 42 4	g ASR	062– 43 b
$x \gtrless y$	047– 34	$x \gtrless y$	063– 34
GTO 1	048– 22 1	g SF 4	064–43, 4, 4
g LBL 4	049–43,22, 4	GTO 2	065– 22 2
R↓	050– 33		

Examples:

Keystrokes	Display	(STATUS : 2–32–0000)
HEX 80000000	80000000 h	−0.
GSB B	0.000000 00	**C** set.
HEX 7F800000		$+\infty$.
GSB B	9.999999 99	**C** and **G** set.
HEX 00800000		$2^{-126} \times (1.00 \ldots 00)$.
GSB B	1.175494–38	
HEX 3F800001		$2^0 \times (1.00 \ldots 01) = 1 + 2^{-23}$.
GSB B	1.000000 00	
f CLEAR PREFIX	100000119	

Program: Conversion from HP-16C Format to IEEE Format

The following program converts a number from Decimal Floating-Point mode to IEEE single-precision floating-point binary format. Flag 5 (out-of-range) is set if ±∞ is the result. (The labels used in this program are different from those in program 1 so that both programs may be in memory at the same time.)

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
g LBL A	001–43,22, A	0	025– 0
f SET COMPL 2's	002– 42 2	f WSIZE	026– 42 44
HEX	003– 23	8	027– 8
g CF 4	004–43, 5, 4	0	028– 0
g CF 5	005–43, 5, 5	+	029– 40
g x=y	006– 43 49	1	030– 1
g RTN	007– 43 21	8	031– 8
9	008– 9	f MASKL	032– 42 7
D	009– d	f AND	033– 42 20
+	010– 40	g F? 4	034–43, 6, 4
x≷y	011– 34	g ISZ	035– 43 24
g CF 0	012–43, 5, 0	f SL	036– 42 A
g x<0	013– 43 2	RCL I	037– 45 32
g SF 0	014–43, 4, 0	F	038– F
g ABS	015– 43 8	F	039– F
x≷y	016– 34	g x>y	040– 43 3
g x<0	017– 43 2	GTO 7	041– 22 7
GTO 9	018– 22 9	x≷y	042– 34
1	019– 1	R↓	043– 33
+	020– 40	R↓	044– 33
g LBL 6	021–43,22, 6	g CLx	045– 43 35
STO I	022– 44 32	g R↑	046– 43 33
R↓	023– 33	g R↑	047– 43 33
2	024– 2	g SF 5	048–43, 4, 5

KEYSTROKES	DISPLAY	KEYSTROKES	DISPLAY
g LBL 7	049–43,22, 7	g LBL 9	062–43,22, 9
R↓	050– 33	g ABS	063– 43 8
f OR	051– 42 40	3	064– 3
g F? 0	052–43, 6, 0	0	065– 0
GSB 8	053– 21 8	g $x \leq y$	066– 43 1
9	054– 9	$x \gtrless y$	067– 34
f RRn	055– 42 F	R↓	068– 33
g CF 4	056–43, 5, 4	0	069– 0
g RTN	057– 43 21	$x \gtrless y$	070– 34
g LBL 8	058–43,22, 8	f SB	071– 42 4
8	059– 8	÷	072– 10
f SB	060– 42 4	0	073– 0
g RTN	061– 43 21	GTO 6	074– 22 6

Examples:

Keystrokes	Display	(STATUS : 2–32–0000)
f FLOAT ·		
8 f EEX 72	8 72	
GSB A	7F800000 h	G set. Overflows to $+\infty$.
f FLOAT ·		
1.404 f EEX	1.404 00	
45 CHS	1.404 −45	
GSB A	1 h	
f FLOAT ·		
3.141592654	3.141592654	π.
GSB A	40490Fdb h	

Function Summary and Index

This section presents a summary of the various calculator functions with page references. The functions are presented in the following groups:

ON Turns the calculator's display on and off **(page 16)**. Also used with - to reset Continuous Memory **(page 20)**, with · to change the digit separator **(page 61)**, and with other keys to test the calculator's operation **(page 106)**.

Clearing

BSP Backspace. In Run mode: clears last digit; clears whole display if digit entry was terminated **(page 17)**. In Program mode: deletes the current instruction (**page 83**).

CLx Clear X. Clears contents of X-register to zero **(page 17)**.

CLEAR PRGM Clear program memory. In Run mode: repositions program memory to line 000 *without* deleting lines. In Program mode: deletes all program memory **(page 73)**.

CLEAR REG Clear registers. Clears all data storage registers **(page 68)**.

CLEAR PREFIX Cancel prefix entry. Cancels any prefix keystroke from a partially entered key sequence **(page 17)**. Also temporarily displays the full 10-digit mantissa of the number in the X-register (Floating-Point Decimal mode only) **(page 58)**.

Digit Entry

0 through 9, A through F digit keys. Can be used only in the proper number base mode **(page 28)**.

· Decimal point. Used in Floating-Point Decimal mode only **(page 58)**.

ENTER Copies the number in the X-register into the Y-register, terminates digit entry and disables the stack. Used to separate multiple number entries **(page 22)**.

CHS Change sign. Returns the appropriate complement or negative of the number in the X-register **(pages 30 and 58)**.

EEX Enter exponent. Used only in Floating-Point Decimal mode; digits keyed in following EEX are considered exponents of 10 **(page 58)**.

Stack Rearrangement

$x \gtrless y$ X exchange Y. Exchanges contents of X- and Y-stack registers **(page 23)**.

R↓, R↑ Roll down, roll up. Rolls contents of stack up or down one register **(page 23)**.

Number and Display Control

HEX, DEC, OCT, BIN Number base modes. Convert display to the specified base in Integer mode **(page 28)**.

SHOW {HEX, DEC, OCT, BIN} Temporarily display the contents of the X-register in the specified base **(page 29)**.

SET COMPL {1's, 2's, UNSGN} Set Complement mode. Establish 1's Compl., 2's Compl., or Unsigned mode for calculator operation **(page 30)**.

WSIZE Word size. Uses the absolute value of the number in the X-register (0 to 64) to specify word size (1 to 64); the stack then drops **(page 32)**.

WINDOW {0 to 7} Displays the specified eight-digit segment of the number in the X-register **(page 33)**.

<, > Scroll left, scroll right. Scrolls the number in the display one digit to the left or right to view obscured digits **(page 33)**.

SF, CF Set flag, clear flag. Sets or clears the flag specified (0 to 5) **(page 36)**.

STATUS Temporarily displays the current complement mode, word size, and flag status **(page 37)**.

FLOAT {0 to 9, ·} Establishes Floating-Point Decimal mode, displaying the given number of decimal places or (with ·) scientific notation. When going into Floating Point mode from Integer mode, the contents of the X- and Y-registers are converted to the floating-point decimal equivalent of $(y)(2^x)$ in the X-register (the rest of the stack is cleared **(page 56)**.

Mathematics

+, −, ×, ÷ Arithmetic operators; cause the stack to drop. In Integer mode, ÷ does not display the fractional part **(page 41)**.

RMD Remainder. Calculates $|y|$ MOD $|x|$ (sign matches y) and drops the stack **(page 43).**

$\sqrt{x}$, $1/x$ Square root and reciprocal. Use the value in the X-register; no stack movement. $1/x$ works only in Floating Point mode. In Integer mode, $\sqrt{x}$ does not display the fractional part **(pages 44 and 61).**

DBL×, DBL÷, DBLR Double multiplication, division, and remainder. DBL× returns a double-word sized product in X and Y; DBL÷ and DBLR take a double-word sized dividend in Y and Z (divisor in X) and return the result in X **(page 52).**

ABS Absolute value. Acts on number in X-register **(page 44).**

Bit Manipulation

SL, SR Shift left, shift right. Shifts the bits in the X-register one place to the left or right. A bit shifted out goes into the carry; a new bit is always zero **(page 46).**

ASR Arithmetic shift right. Shifts all bits in X one place to the right and replicates the sign bit on the left (in 1's and 2's Complement modes only) **(page 47).**

RL, RR Rotate left, rotate right. Rotates the bits in the X-register one place to the left or right; a bit shifted out at one end of the word re-enters at the other end and also goes into the carry **(page 48).**

RLC, RRC Rotate left through carry, rotate right through carry. Rotates bits as above except that the bits shifted out "pass through" the carry bit before rotating back into the word **(page 48).**

RLn, RRn, RLCn, RRCn Multiple rotation. Rotates (left/right, through the carry or not) the bit pattern in the Y-register and the number of places specified in the X-register, then drops the stack, placing the new pattern in X **(page 49).**

LJ Left-justify. Left-justifies the word in the X-register and places it in the Y-register. The number of bit-shifts necessary for the justification is placed in the X-register **(page 47).**

MASKL, MASKR Mask left or right. Creates a left- or right-justified mask of set bits. The number of bits is specified by the absolute value of the number in the X-register **(page 50).**

SB, CB Set bit, clear bit. Sets (to 1) or clears (to 0) a bit—specified by the magnitude of the value in X—in a bit pattern in Y. (The stack drops.) Bits are numbered from zero to (word size – 1), where zero is the least significant bit **(page 50).**

#B Number of Bits. Sums the bits in the X-register and returns that sum to X. The stack does not move **(page 52).**

NOT, OR, AND, XOR Logical (Boolean) operators. OR, AND, and XOR operate on the binary values in the X- and Y-registers and return the Boolean result in the X-register. (The stack drops.) NOT uses only the X-register **(page 44).**

Memory and Storage

STO Store. Places a copy of the number in the X-register into the storage register specified (0 to .F, I, (i)) **(pages 66, 69).**

RCL Recall. Places a copy of the number in the specified storage register (0 to .F, I, (i)) into the X-register **(pages 66, 69).**

x≷I, x≷(i) Direct and indirect Index register exchanges. Refer to Index Register Control.

LSTx LAST X register. Recalls into the X-register the number that was in X before the last operation **(page 23).**

MEM Memory status. Temporarily displays 1) the number of instructions which may be added to program memory before another seven lines are allocated; and 2) the number of storage registers currently available for data storage **(page 65).**

CLEAR REG, CLEAR PRGM Clear storage registers, clear program memory. Refer to Clearing, above.

Index Register Control

I Index register (R_I). Storage register that also can be used for indirect program execution (with GTO and GSB) and program loop control (with DSZ and ISZ) **(page 68).**

(i) Indirect operations. Used to indirectly address any storage register, and is the only means to address those registers above $R_{.F}$. The Index register contains the number of the storage register **(page 69).**

x≷I X exchange R_I. Exchanges the value in the X-register with the value in the Index register **(page 69).**

x≷(i) Exchanges the value in the X-register with that in the storage register indirectly addressed by R_I **(page 69).**

DSZ, ISZ Loop counting and control using R_I. Refer to Conditionals **(page 124).**

Programming

P/R Program/Run mode. Sets the calculator to Program mode, for entering program lines, or Run mode, for running programs and performing other operations **(page 72).**

LBL {0 to F} Label. Used to access programs **(page 73)**.

RTN Return. Halts execution of a running program and returns position in program memory to line 000. If a subroutine is running, RTN merely returns execution to the line after the corresponding GSB instruction **(page 74)**.

R/S Run/Stop. Begins or stops execution at the current line in program memory **(page 76)**.

PSE Pause. Halts program execution briefly to display the X-register contents **(page 75)**.

GTO *label*. Transfers program execution to the given label. Programmable **(page 87)**.

GTO · *nnn*. Positions the calculator to the existing line number specified by *nnn*. Not programmable **(page 82)**.

GSB *label*. Go to subroutine. Within a program, this transfers execution to the given subroutine, which returns execution to the body of the program when a RTN instruction is encountered. From the keyboard, GSB is used to start execution of a labeled program **(page 87)**.

SST Single step. Program mode: moves calculator forward one or more lines in program memory; scrolls if key is held. Run mode: displays and executes the current program line, then steps to the next line to be executed **(page 82)**.

BST Back step. Moves calculator back one line in program memory (will also scroll in Program mode). Displays line number and contents of previous program line **(page 82)**.

Conditionals

F?, B? Flag set? Bit set? Tests for the flag or bit specified. If set, program execution continues; if cleared, program execution skips one line before continuing **(page 89)**.

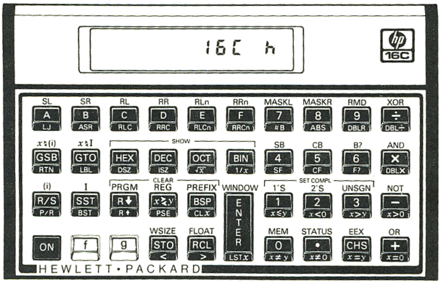
$x \leqslant y$, $x < 0$, $x > y$, $x > 0$, $x \neq y$, $x \neq 0$, $x = y$, $x = 0$ Conditional tests. Each test compares the value in the X-register against zero or the value in the Y-register. If the comparison is true, program execution continues; if false, program execution skips one line before continuing **(page 88)**.

DSZ, ISZ Decrement and skip if zero, increment and skip if zero. Decrements or increments value in Index register and skips execution of the next program line *if* the new index value equals zero **(page 90)**.

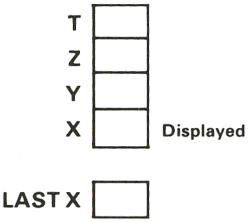
Subject Index

Page numbers in **bold** type indicate primary references; page numbers in regular type indicate secondary references.

The HP-16C Keyboard and Continuous Memory

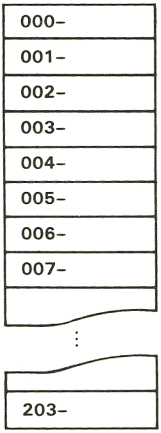

16C h
hp 16C
SL SR RL RR RLn RRn MASKL MASKR RMD XOR
A B C D E F 7 8 9 ÷
LJ ASR RLC RRC RLCn RRCn #B ABS DBLR DBL÷
x⇄(i) x⇄I SHOW SB CB B? AND
GSB GTO HEX DEC OCT BIN 4 5 6 ×
RTN LBL DSZ ISZ √x 1/x SF CF F? DBL×
(i) I PRGM CLEAR REG PREFIX WINDOW 1'S SET COMPL 2'S UNSGN NOT
R/S SST R↓ x≷y BSP ENTER 1 2 3 −
P/R BST R↑ PSE CLx LSTx x≤y x<0 x>y x>0
WSIZE FLOAT MEM STATUS EEX OR
ON f g STO RCL 0 • CHS +
< > x≠y x≠0 x=y x=0
HEWLETT·PACKARD

AUTOMATIC MEMORY STACK



PROGRAM MEMORY

Made available in blocks of seven lines.

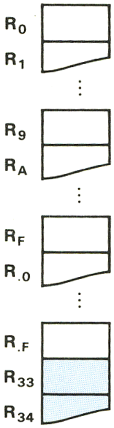


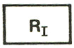
There are 203 bytes (lines) of total memory for programming and storage. All memory is initially in storage registers and is automatically allocated to program memory as necessary.

When each seventh line (001, 008, 015, ..., 197) is keyed into a program, seven more bytes of memory are converted from data storage into program memory. The number of registers required to supply seven bytes depends on the current word size.

STORAGE REGISTERS

Maximum number depends on word size. *Only R_0 to $R_{.F}$ are directly accessible.*





Index Register (not convertible)

HEWLETT
PACKARD

Corvallis Division
1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

00016-90001 Printed in U.S.A. 4/82