A Pocket Calculator for Computer Science Professionals

This compact, yet powerful pocket calculator is designed for technical professionals working in computer science and digital electronics. Boolean operations and bit manipulation are some of its capabilities.

by Eric A. Evett

LOGIC DESIGN and computer programming require mathematical operations not ordinarily provided by small calculators. A large amount of tedious paperwork is often required to convert among number bases, perform logic operations, shift and rotate bits in a word, or check processor instruction flow. To simplify such work, Hewlett-Packard recently introduced a programmable pocket calculator especially designed for people who deal with bits. The HP-16C (Fig. 1), like other HP calculators, uses a reverse-Polish-notation (RPN) system and provides standard floating-point decimal arithmetic (including square root). Its novel capabilities become apparent, however, when the HP-16C is switched into the integer mode. Only integers are allowed in this mode, and they can be keyed in and displayed in either hexadecimal, octal, binary, or decimal format. In this mode, number base conversion, integer arithmetic, logical operations, and bit manipulations can be done.

Integer Mode

In the integer mode, all numbers are represented internally in binary form. The word size is selected by the user and can range from 1 to 64 bits. The user also can select whether the numbers are to be interpreted as one's complement, two's complement, or unsigned integers. In the unsigned integer mode with a 64-bit word size, numbers up to 2^{64-1} (18,446,744,073,709,551,615) can be represented. Although the HP-16C normally displays the eight least-significant digits of a number, a scrolling capability is provided to display higher-order digits.

Programming

In addition to the four-register RPN stack, 203 bytes of user memory are available for storing program steps and use as storage registers. When the program memory is cleared, all 203 bytes are allocated to storage registers. The number of storage registers available depends on the selected word

Fig. 1. *The HP-16C Programmable Calculator is designed for computer science and digital electronics applications. Besides the normal four-function calculator features, it has a number of capabilities for setting number bases and word sizes, performing Boolean operations, and manipulating bits.*

Real (Floating-Point) Format

Real numbers are represented in the HP 3000 memory by 32 bits (two consecutive 16-bit words) separated into three fields. These fields are the sign, the exponent, and the mantissa. The format is known as excess 256. Thus, a real number consists of (see Fig. 1):

- Sign (S), bit 0 of the first 16-bit word. Positive=0, negative=1. A value X and its negative −X differ only in the value of the sign bit.
- Exponent (E), bits 1 through 9 of the first 16-bit word. The exponent ranges from 0 to 777 octal (511 decimal). This number represents a binary exponent biased by 400 octal (256 decimal). The true exponent, therefore is E−256; it ranges from −256 to +255.
- Fraction (F), a binary number of the form 1.xxx, where xxx represents 22 bits stored in bits 10 through 15 of the first 16-bit word and all bits of the second 16-bit word. Note that the 1 is not actually stored, there is an assumed 1 to the left of the binary point. Floating-point zero is the only exception. It is represented by all 32 bits being zero.

The range of nonzero real values for this format is from 0.863617×10^{-77} to 0.1157920×10^{78}. The formula for computing the decimal value of a floating-point representation is: Decimal value = $(-1)^S \times 2^{E-256} \times F$.

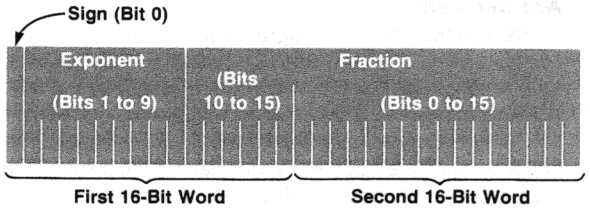


Fig. 1. *Diagram of real (floating-point) format used in the HP 3000.*

```
LBL A
2'S      Set two's complement mode.
OCT      Convert to octal integer mode,
         and return integers y and x
         such that 2^x*y=original input.
SF 3     Leading zeros will be displayed.
X=Y      Was input 0?
GTO 1    If yes, then branch to Label 1.
4
3
7
+        Bias exponent. 287=256+31
STO I    Store biased exponent in index register.
X≷Y      Swap exponent and mantissa.
SF 0     Set flag 0.
X<0      Mantissa negative?
CF 0     If yes, clear flag 0.
ABS      Absolute value of mantissa.
4
0
WSIZE    Set word size to 32.
4
0
0
+        Round mantissa to 23 bits.
2
7
MASKL    Create mask of 23 bits, left-justified.
AND      Extract upper 23 bits.
F? 4     Did round cause a carry-out
         of most significant bit?
ISZ      If yes, increment exponent.
SL       Shift off implied 1 bit.
RCL I    Recall biased exponent.
OR       Concatenate exponent to fraction part.
F? 0     Is mantissa sign to be positive?
GTO 1    If yes, branch to Label 1.
1
1
SB       Set the sign bit.
LBL 1
1
2
RRn      Rotate sign, exponent, and fraction
         to proper position.
STO 0    Store the 32-bit result in register 0.
2
0
WSIZE    Change word size to 16 bits.
RCL 1    Recall 16-bit word 1.
RCL 0    Recall 16-bit word 2.
RTN
```

Fig. 2. *Outline of HP-16C subroutine to convert numbers given in the HP 3000 Computer's real format to decimal floating-point format.*

Using the HP-16C

Listing the features of a programmable calculator rarely provides a complete picture of its capabilities. Examples of the application of the calculator's features are often required to demonstrate to the user what can be done and why a particular feature is useful. Hence, several examples of the use of the HP-16C are given below.

Add with Carry

The HP-16C can be programmed to simulate instructions commonly found in commercial processors. The following subroutine performs an add with carry (Y+X+C→X). It adds the numbers in registers X and Y along with the carry bit indicated by the state of flag 4 and returns the result in the X register. The carry flag is set (indicated by the C annunciator in the display) if there is a carry-out of the most significant bit of the result.

001	**LBL A**	Labels subroutine
002	**0**	
003	**RLC**	Generates 0 or 1 depending on carry flag
004	**+**	Adds carry to second operand
005	**CF 0**	
006	**F? 4**	Copies carry flag 4 to flag 0
007	**SF 0**	
008	**+**	Adds first operand to the total
009	**F? 0**	
010	**SF 4**	Sets carry flag if first add carried
011	**RTN**	

To use this routine, enter the two operands in registers Y and X, and press **GSB A**.

Example:

2'S	Set two's complement mode
HEX	Set number base to hexadecimal
8	
WSIZE	Set word size to 8
CF 4	Clear carry flag
FE	First operand
ENTER	Enter first operand into Y register
72	Second operand
GSB A	Displays 70 (FE+72+0) with carry set (C annunciator on)

Bit Extraction

The following subroutine extracts a field from a bit pattern. The field is specified by the bit numbers of the pattern corresponding to the lowest-order and highest-order bits of the field. The least-significant bit of the bit pattern is bit number 0. Hence, the result in the X register is the bits of the pattern in the Z register from the bit number in the Y register to the bit number in the X register, inclusive.

001	**LBL B**	Labels subroutine
002	**R↓**	Bring down value in Y register
003	**RRn**	Right-justifies field
004	**R↑**	Raise stack
005	**LST x**	Recall Y value
006	**–**	Subtract Y from X
007	**1**	
008	**+**	Computes number of bits in field
009	**MASKR**	Creates mask same width as field
010	**AND**	Extracts field
011	**HEX**	Exits in the hexadecimal mode
012	**RTN**	

size. When the word size is eight bits, 203 registers are available; a 16-bit word size results in 101 available registers, and so on. Each programmable instruction takes one byte of memory. As program steps are inserted, the number of available storage registers decreases. A program can have up to 203 steps if no storage registers are required.

Editing capabilities to make program development easier include insert, delete, back-step, single-step, and go-to-line-number operations. The user may single-step through program execution to help debug programs. Other programming features include label addressing (sixteen labels), subroutines (up to four levels deep), conditional tests, branching, and six user flags.

These flags can be set, cleared, and tested under program control. Three of the flags are special. Leading zero digits in a word are suppressed in the display unless flag 3 is set. Flag 4 is the carry flag, and flag 5 is the overflow flag. Two annunciators in the display (C for carry and G for > largest representable number) give a visual indication of the state of flags 4 and 5, respectively. The overflow flag is set if the true result of an operation cannot be represented in the selected word size and complement mode. The carry flag is set under various conditions, depending on the operation. For example, addition sets the carry flag if there is a carry-out of the most significant bit; otherwise the carry is cleared (see box above for examples). The shift-left instruction sets the carry if a 1 bit is shifted off the left end of the word; otherwise the carry is cleared.

Logic Operations

The rich selection of bit manipulation and logical operations, along with user-selectable complement mode and word size, make the HP-16C a flexible logic and program design tool. Programs can be written to simulate individual instructions commonly found on commercial processors, to extract a field from a bit pattern, or to convert from one numeric format to another.

A common problem is the conversion between the internal binary floating-point format of a particular machine and decimal floating-point format. The HP-16C provides a feature that can be used to great advantage in programs designed to perform such conversions. This feature provides a mode for performing standard decimal floating-point calculations. Upon switching from the integer mode to decimal floating-point mode (by using the **FLOAT** function), the integers y and x in the Y and X stack registers are converted to the floating-point equivalent of $2^x y$, which is then placed in the X register and displayed. Converting back to integer mode (by pressing the **HEX**, **DEC**, **OCT**, or **BIN** keys), causes the contents of the X register to be converted to a pair of integers y and x such that y is a 32-bit integer ($2^{31} \leq |y| < 2^{32}$ unless $y=0$) and $2^x y$ is equal to the value in the X register before mode conversion. The integers y and x are then placed in the Y and X registers.

To use this routine, the user places the bit pattern in register Z, the number of the lowest-order bit in the field in register Y, and the number of the highest-order bit in the field in register X. The user then presses **GSB B**.

Example: Extract bits 2 through 5 from 39_{16} (00111001).

8	
WSIZE	Set wordsize to 8
HEX	Set hexadecimal mode
39	Bit pattern
ENTER	
2	Lowest-order bit
ENTER	
5	Highest-order bit
GSB B	Displays E (1110) as result.

Conversion Between Binary and Gray Code

Gray code has the property that only one bit changes between the representations of any two adjacent numbers. If the word size is n bits, then binary-to-Gray-code conversion is given by

$$G_0 = B_0 \text{ XOR } B_1$$

$$G_1 = B_1 \text{ XOR } B_2$$

$$\vdots$$

$$G_{n-2} = B_{n-2} \text{XOR } B_{n-1}$$

$$G_{n-1} = B_{n-1}$$

where G is the Gray code number, B is the binary number, and subscript 0 indicates the least-significant bit of G and B, subscript 1 indicates the next least-significant bit, and so forth.

The Gray-code-to-binary conversion is given by

$$B_0 = G_0 \text{ XOR } G_1 \text{ XOR } \cdots G_{n-1}$$

$$B_1 = G_1 \text{ XOR } G_2 \text{ XOR } \cdots G_{n-1}$$

$$\vdots$$

$$B_{n-2} = G_{n-2} \text{ XOR } G_{n-1}$$

$$B_{n-1} = G_{n-1}$$

Binary-to-Gray-code subroutine:

001	**LBL C**	
002	**ENTER**	Copies binary number to Y register
003	**SR**	Shifts binary number in X register to the right
004	**XOR**	Computes Gray-code equivalent
005	**RTN**	

Gray-to-binary-code subroutine:

001	**LBL D**	
002	**ENTER**	Copies Gray code number to Y register
003	**LBL 2**	
004	**SR**	Shift Gray code number in X register to the right
005	**XOR**	Exclusive OR operation
006	**LST x**	Recall previous number
007	**X≠0**	Loop until Gray code number is 0
008	**GTO 2**	
009	**R↓**	
010	**RTN**	

To use these routines, the user sets the HP-16C to the binary mode by pressing **BIN**, places the number in the X register and presses **GSB C** for binary-to-Gray or **GSB D** for Gray-to-binary-code conversions.

LBL B	
HEX	**Set hexadecimal base mode.**
2'S	**Set two's complement mode.**
2	
0	
WSIZE	**Set word size to 32 bits.**
X≷Y	**Swap word 1 and word 2.**
1	
0	
RLn	**Shift word 1 16 bits to left.**
OR	**Concatenate word 2 to word 1.**
SL	**Shift sign bit left into carry flag.**
ENTER	
X=0	**Is input 0?**
GTO 0	**If yes, branch to Label 0.**
1	
7	
MASKR	**Create mask of 23 bits, right-justified.**
AND	**Extract fraction part.**
XOR	**Extract biased exponent part.**
LST x	**Recover fraction part.**
1	
7	
SB	**Set bit 23 (the implied 1-bit). Bit 0 is the least significant.**
F? 4	**Test carry flag. Was sign bit negative?**
CHS	**If yes, complement mantissa.**
ASR	**Arithmetic shift right mantissa 1 bit.**
X≷Y	**Swap mantissa and exponent part.**
9	
RLn	**Rotate exponent part 9 bits left, placing it at right end.**
1	
1	
6	
−	**Unbias exponent. E−2⁷8=E−256−32**
LBL 0	
FLOAT	**Compute 2^{xy}**
RTN	

Fig. 3. *Outline of HP-16C subroutine to convert decimal floating-point numbers into the format used by HP 3000 Computers.*

The subroutines listed in Fig. 2 and Fig. 3 convert between the HP 3000 Computer's FORTRAN real (floating-point) format[2] and decimal floating-point format (see box on page 37). Because the HP-16C views bit 0 as the least significant bit of a word and the HP 3000 views it as the most significant bit, some of the steps listed in Fig. 2 and Fig. 3 are used to convert between these two opposing views.

To use these programs after they are entered in the HP-16C, a user performs the following steps.

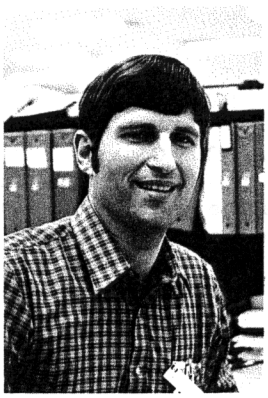
- HP 3000 to decimal:
  1. Select octal base (**OCT**).
  2. Enter word 1 in the Y register and word 2 in the X register.
  3. Execute **GSB B**. Answer is displayed.
  4. Repeat steps 1, 2, and 3 for each new conversion.
- Decimal to HP 3000:
  1. Select decimal floating-point mode (**FLOAT 4**).
  2. Enter number in the X register.
  3. Execute **GSB A**. Word 1 is placed in the Y register, word 2 in the X register.
  4. Repeat steps 1, 2, and 3 for each new conversion.

Acknowledgments

Several people made significant contributions to the HP-16C software effort. John Van Boxtel developed many of the fundamental design concepts. Rich Carone also contributed to the software design and coded the complex routines that format and build the display. Stan Blascow did a large portion of the software testing and Diana Roy wrote the owner's handbook.

References

1. M.E. Sloan, *Computer Hardware and Organization*, Science Research Associates, Inc., 1976, pp 95-97.
2. *FORTRAN Reference Manual*, HP 3000 Computer Systems, Hewlett-Packard, Santa Clara, 1978, Section II.



Eric A. Evett

Eric Evett received the BS and MS degrees in mathematics from the University of Arizona in 1970 and 1972. He taught mathematics there until 1975 and then spent three years developing software for calculators before joining HP in late 1978. Eric wrote portions of the microcode for the HP-11C, HP-15C, and HP-16C Calculators and now is an R&D software project manager at HP's Corvallis, Oregon facility. He was born in Colfax, Washington, and now lives in Corvallis. He is married and has two sons. His outside interests include jogging, hiking, basketball, reading, movies, and tennis (he played on his college's tennis team which was then ranked 7th in the U.S.A.).

CORRECTION

In the March issue, the Pascal statements at the top of the back page were printed in the wrong order. Here is the correct version.

```
buffer[w]:=getch;
c:=c+a[buffer[w]];
w:=w+1;
```


Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, California 94304

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

HEWLETT-PACKARD JOURNAL

MAY 1983 Volume 34 • Number 5

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3000 Hanover Street
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
1181 KK Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

CHANGE OF ADDRESS: To change your address or delete your name from our mailing list please send us your old address label. Send changes to Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, California 94304 U.S.A. Allow 60 days.

5953-8511