MAY 1974

HEWLETT-PACKARD JOURNAL

The "Personal Computer": A Fully Programmable Pocket Calculator

This 11-ounce battery-powered marvel has the computing power of an advanced scientific pocket calculator and is programmable as well, so it can adapt to any number of specialized uses.

by Chung C. Tung

AN ENGINEER OR SCIENTIST in need of an on-the-spot answer in the laboratory, a pilot making an in-flight course correction, a surveyor running a traverse in the field, a businessman estimating returns-on-investment during a conference, a physician evaluating patient data—it isn't difficult to think of everyday examples of people whose professions require certain types of calculations over and over again. Were one available, a computer or programmable calculator would obviously be of great assistance to such people. Unfortunately, you can't carry one of those around in your pocket, can you?

You can't, unless it's battery-powered, weighs only 312 grams, and measures only 8.1 × 14.7 × 3.4 cm, like the HP-65 Pocket Calculator. This new computing device is a combination of electronic calculator and general-purpose small computer. It offers the convenience and easy operation of a calculator, but its programmability makes it versatile enough to fit the needs of a wide variety of disciplines, including science, engineering, finance, statistics, mathematics, navigation, medicine, surveying, and many others.

Although the HP-65 is designed to operate in a logical, easy-to-learn way, it is capable of sophisticated computations. It has 51 built-in mathematical functions and data-manipulation operations, a four-register operational stack, nine addressable data registers, and five user-definable keys. It can run programs that have as many as 100 steps. There are two program flags and four comparison tests for program branching.

Perhaps most significant is that within its small package the HP-65 contains a tiny magnetic card reader and recorder. Users can store their programs on magnetic cards for later use, or they can take advantage of hundreds of preprogrammed cards containing programs commonly used in various disciplines.

Fig. 1. *The first fully programmable pocket calculator has 51 preprogrammed operations, a 100-step program memory, five user-definable keys, and a tiny magnetic card reader-recorder. After a program card is read it can be placed in the window above the keyboard to show the new functions of the user-definable keys.*

Keyboard Design

The HP-65 inherits its size and shape from its ancestor, the HP-35. However, its keyboard layout is quite different. Thirty-five keys control more than eighty operations (see Fig. 1).

In the interests of logical operation and simplicity, many different techniques were used in designing the keyboard layout. Keys of the same nature are grouped into clusters. Some nomenclature has been placed on the lower side of the keys to reduce busyness. Nomenclature for multiple-keystroke operations is color-coded to make the keystroke sequences associative. All functions are classified as immediate (+,−,×,÷), direct, inverse, or miscellaneous, and are grouped and color-coded accordingly. Key sizes, colors, value contrast, and nomenclature have all been chosen to guide the user.

Arithmetic operations (+,−,×,÷) and data entry keys are in the same locations as on the

PRINTED IN U. S. A.

HP-35. Data entry keys include the digits 0 through 9, the decimal point, the ENTER↑ key used for entering values into the four-register operational stack, and the EEX (enter exponent) and CLx (clear display) keys.

Keys in the top row, labeled A, B, C, D, and E, stand for user-definable functions or subroutines. In the second row, the GTO (go to), LBL (label), RTN (return), and SST (single step) keys are used for programming. The DSP (display) key is used for formatting the display. Numbers can be displayed either in fixed-point notation or in scientific notation with a selected number of digits after the decimal point.

Keys in the third row, labeled f, f^{-1}, STO, RCL, and g, are all prefix keys. They have to be followed by one or two more keystrokes to complete a command.

At the lower right corner of the HP-65 keyboard is the R/S (run/stop) key. Pressing it begins program execution. Execution of a running program halts if an R/S step is encountered in the program.

User-Definable Keys

When power is first applied to the HP-65 the top-row keys (A,B,C,D,E) call for the functions 1/x, $\sqrt{x}$, y^x, x⇆y, and R↓ (roll down the operational stack). However, the functions of these keys can be changed, either by keyboard programming or by inserting a previously recorded magnetic card into the HP-65's reader slot. The new functions of the keys can be written on the card and the card inserted into the window above the top row of keys to show the new functions.

With the five definable keys, the user can readily change the HP-65 into a specialized calculator tailored to his needs. For example, Standard Pac program 11A, a compound interest program, changes these keys to n (number of periods), i (interest rate), PV (present value), FV (future value), and COMPUTE, similar to keys on the HP-80 financial calculator (see Fig. 2). Standard Pac 06A, a π-network impedance-matching program, converts the HP-65 into an electrical engineer's calculator. Given input and output resistance, frequency, and Q, the program computes the values of the two shunt capacitors and the

Cover: *In the background is Model 9100A, HP's first calculator, a huge success when it was first introduced in 1968 at a price of $4900. The HP-65 in the foreground rivals the 9100A in computing power, but it fits in your pocket and costs only $795. Of course, the HP-65 doesn't work with a printer or other peripheral devices. Perhaps that's the next step.*

In this Issue:

COMPOUND INTEREST

COMPOUND INTEREST — STD 11A

n i PV FV COMPUTE

This program computes the answers to compound interest problems using the basic formula:

$$FV = PV\ (1 + i/100)^n$$

where:
- n = Number of time (compounding) periods.
- i = Interest rate per time period (in percent).
- PV = Present value (value at the beginning of the first time period).
- FV = Future value (value at the end of n time periods).

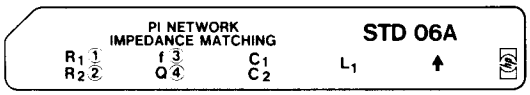
Any three of the variables (n, i, PV, FV) can be inputs. The program computes and stores the fourth variable. Input variables can be entered in any order and need not all be reentered to change one variable.

HP-65 User Instructions

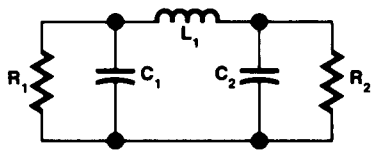
STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Enter program			
2	Initialize		RTN R/S	
3	Input three of the following			
	n	n	A	n
	and/or i	i(%)	B	i(%)
	and/or PV	PV	C	PV
	and/or FV	FV	D	FV
4	Compute the remaining variable			
	n		E A	n
	or i		E B	i(%)
	or PV		E C	PV
	or FV		E D	FV
5	To modify problem go to 3 and			
	change appropriate input(s).			

Fig. 2. *Users may write their own programs and store them on blank magnetic cards, or may take advantage of many preprogrammed cards. This compound interest program converts the calculator into a financial calculator.*

PI NETWORK IMPEDANCE MATCHING



A lossless network is often used to match between two resistive impedances, R_1 and R_2, as shown.



Given the values of R_1 and R_2, the frequency f, and the desired circuit Q (center frequency/desired half-power bandwidth), the values of C_1, C_2, and L_1 can be found.

Notes:

1. R_1 must always be greater than R_2 and

$$Q > \sqrt{R_1/R_2 - 1}\,.$$

2. Circled numbers on the magnetic card designate the register in which a variable is stored.

HP-65 User Instructions

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
1	Enter program				
2	Initialize		RTN	R/S	0.0000
3	Input R_1	$R_1(\Omega)$	E	A	$R_1(\Omega)$
	and R_2	$R_2(\Omega)$	A		$R_2(\Omega)$
	and f	f(Hz)	E	B	f(Hz)
	and Q	Q	B		Q
4	Compute C_1		E	C	C_1(F)
5	Compute C_2		C		C_2(F)
6	Compute L_1		D		L_1(H)
7	Recall inputs (optional)				
	R_1		RCL	1	$R_1(\Omega)$
	and/or R_2		RCL	2	$R_2(\Omega)$
	and/or f		RCL	3	f(Hz)
	and/or Q		RCL	4	Q
8	For new case change appropriate input in step 3.				

Fig. 3. *This program converts the HP-65 into an electrical engineer's calculator.*

series inductor (see Fig. 3).

For more information on HP-65 keys and programming, see the article beginning on page 8.

Magnetic-Card Reader/Writer

The HP-65's built-in magnetic card reader/recorder uses mylar-based ferrite-oxide-coated cards 0.95 cm wide and 7.1 cm long. Each card can store 100 program steps or 600 bits of information. A two-track self-clocking recording scheme is used to maximize the system's tolerance to head skew and motor speed variations (see box below).

When the right-hand switch just below the display is in the RUN position, insertion of a card into the reader slot in the right side of the calculator triggers the motor and read circuits. All 100 program steps on the card are read into the calculator's memory.

When the same switch is in the W/PRGM position, insertion of a card triggers the motor and writing circuits, and the contents of the calculator's memory are written on the card. A card that has the file-protect tab (upper left corner) clipped off will not trigger the writing circuits; the program on the card is thereby protected against accidental erasure.

More information on the design of the tiny card reader is contained in the article beginning on page 15.

System Organization

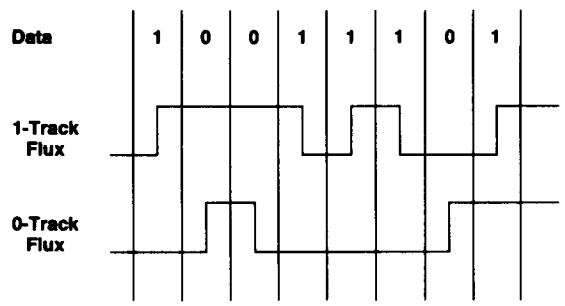
Like its ancestor, the HP-35, the HP-65 contains an arithmetic and register circuit, a control and timing circuit, and three read-only-memory (ROM) circuits. In the HP-65, each ROM is actually a quad ROM containing the equivalent of four single ROMs. Like its more recent predecessor, the HP-45, the HP-65 also

A Self-Clocking Two-Track Recording Technique

When the HP-65 records a program on one of its small magnetic cards, it places two side-by-side tracks of varying magnetic flux on the card. One track represents the logical 1's in the binary data stream coming from the program memory, and the other track represents the logical 0's. The 1 track contains a flux reversal for each 1 in the data sequence and no flux change for each 0. The 0 track contains a flux reversal for each 0 and no flux change for each 1. The diagram shows an example.

The technique is self-clocking because there is a flux transition for each bit. Thus no separate clock track is needed.

The scheme also maximizes the system's tolerance to head skew and motor speed variations. The data can be recovered correctly even if a transition in one track almost overlaps a transition in the other. By contrast, other two-track schemes usually have one clock track and one data track, and may misread data when there is only minor misalignment of clock and data transitions.



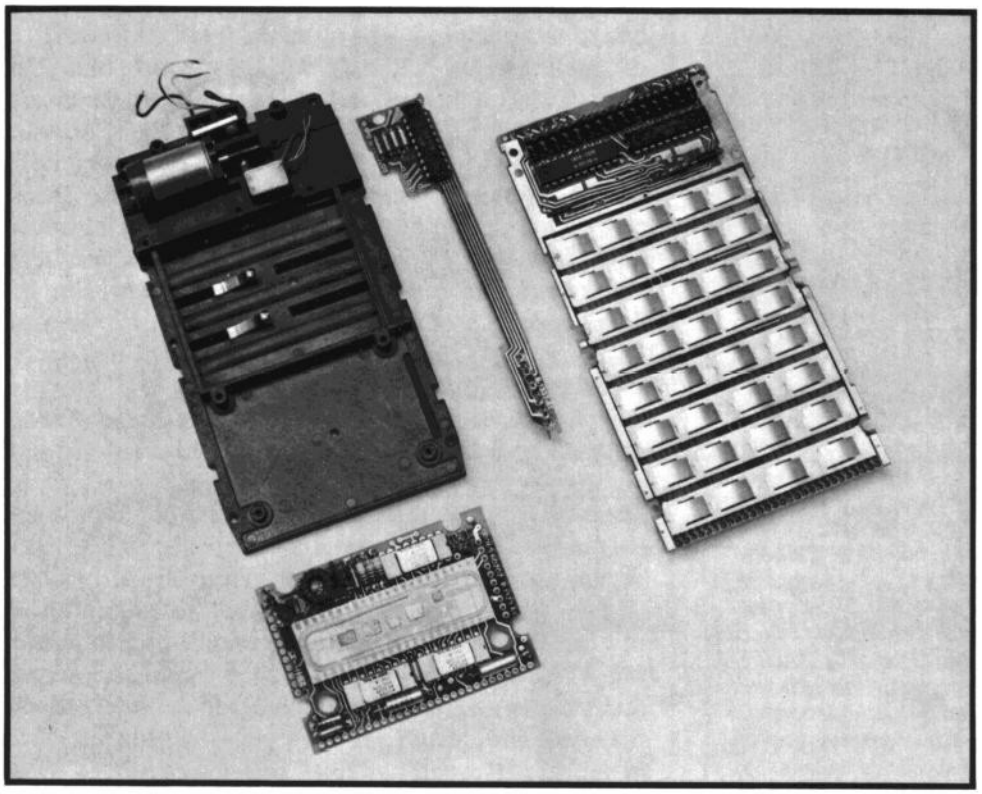


Fig. 4. *Inside the HP-65.*

has a data-register circuit. New in the HP-65 are a program storage circuit, a card-reader control circuit, and a card read/write circuit. Fig. 4 is a photo of the components and Fig. 5 is a block diagram.

The *data register circuit* is made up of ten addressable data registers of 56 bits each, enough to store 14 digits each. Data register 0 is used to implement a LSTx (last x) function and always contains the previously displayed number. Registers 1 through 9 are addressable from the keyboard. Addresses and data are transferred between the data register circuit and the arithmetic and register circuit by way of a bidirectional BCD bus.

The *program storage circuit* stores the user's program. Each program step is stored as a six-bit word. There are 100 words of storage.

The program memory is designed to act like a carousel. The memory is implemented in a dynamic shift register. Program words, a pointer, and a beginning-of-memory marker circulate continuously in this register, a complete circulation taking approximately 3½ milliseconds. The pointer always points to the next program word and can move freely within the memory. Thus program steps can be inserted into the program or deleted from it at any point, without re-keying the other steps. Addressing is symbolic rather than absolute, and label-searching hardware is built into the program storage circuit.

When the calculator is in use, every keystroke places a corresponding keycode on the I_A (instruction address) bus. This code is stored in the buffer of the program storage circuit. If the calculator is in W/PRGM mode, the code is then inserted into the program memory. If the calculator is in RUN mode, the keystroke is executed.

When a stored program is executed, the pointer points to the next executable memory word and the buffer contains a copy of that word. The buffer contents are decoded and placed on the I_S bus as a microinstruction that causes the calculator to enter a subroutine in the ROM to service the key that was pressed.

Card Reader Circuits

The *card-reader control circuit* and the program storage circuit together form a complete memory circulation path. When the HP-65 is used as an ordinary calculator the card-reader control circuit merely short-circuits the data-in and data-out lines of the program storage circuit.

When a card is being read, the card-reader control circuit places the outputs of the head sense amplifier onto the data-in line of the program storage circuit. When a card is being written the card-reader control circuit takes the memory words in sequence and transforms them into recording signals for the head write amplifiers.

When two unsynchronized sequential memories like the program memory and the magnetic card are working together, timing and control are critical.

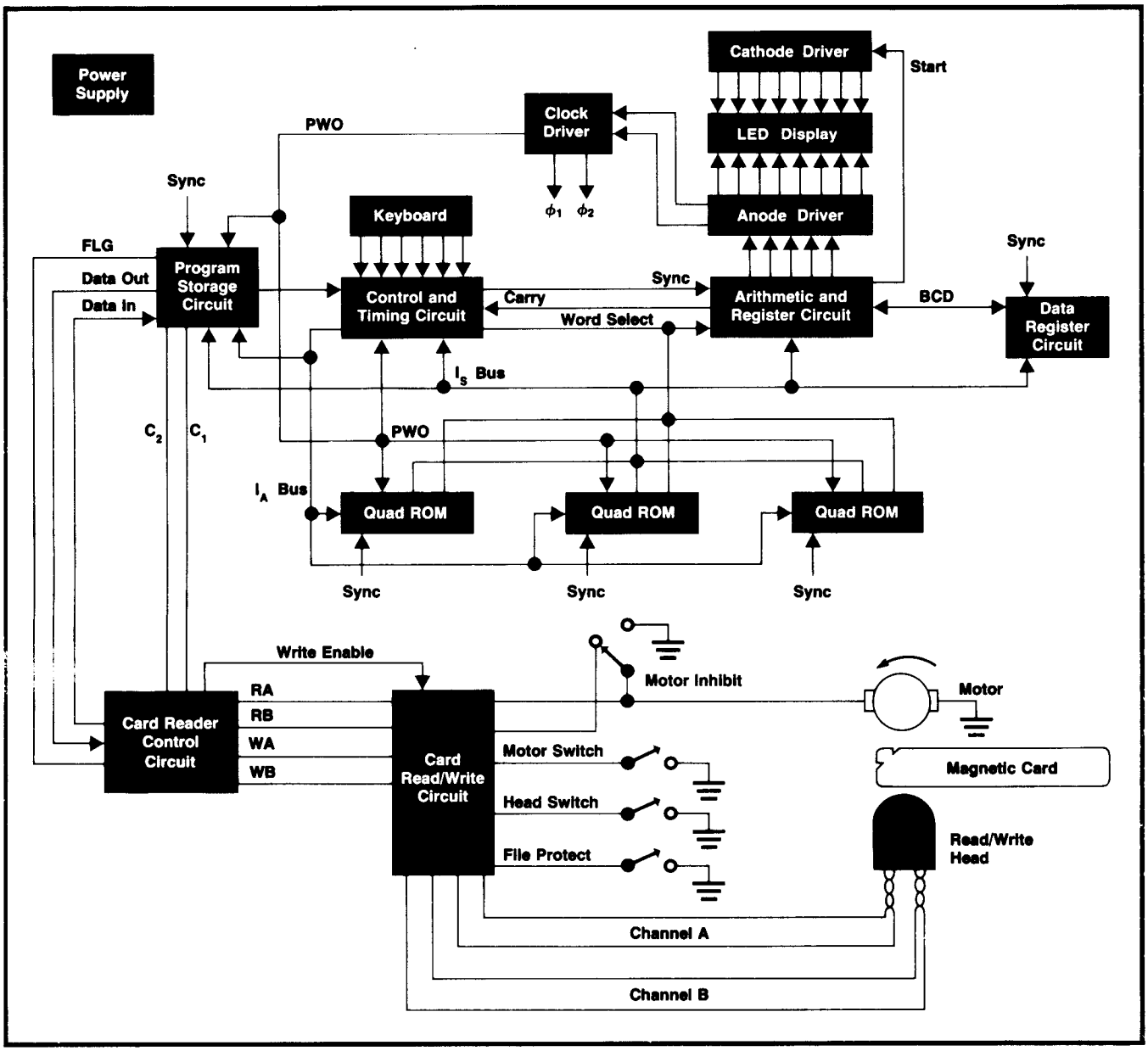


Fig. 5. *Architecture of the HP-65 is stack and bus oriented, like earlier HP pocket calculators. New circuits are the program storage circuit, the card-reader control circuit, and the card read/write circuit.*

Therefore, during reading or writing the card-reader control circuit assumes control of the program memory and controls the movement of the pointer. Thus only limited handshaking is needed between the card-reader control circuit and the program storage circuit.

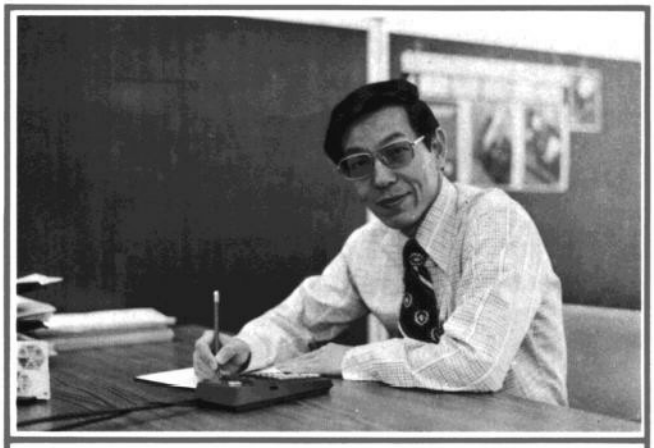
The *magnetic-card read/write circuit* is a bipolar LSI (large-scale-integrated) circuit. It serves as the interface between the magnetic read/write head and the card-reader control circuit. There are two identical channels on the chip, each consisting of a read amplifier, a threshold detector, a latch, and a write amplifier.

The read amplifier takes a one-millivolt signal from the head, amplifies it approximately 200 times and feeds it into the threshold detector. The detector removes noise and triggers the latch to provide a logic signal to the card reader control circuit. The write amplifier takes logic information from the card-reader control circuit and transforms it into a recording current that flows through the inductive head and generates the flux that magnetizes the card.

The read/write circuit also provides a dc motor control circuit that regulates the widely varying battery voltage so the card-reader motor runs at constant speed.

Acknowledgments

The HP-65 is the creation of a large number of people who contributed their efforts with great enthusiasm and engineering professionalism. The support of the IC departments of the Loveland and Santa Clara Divisions, and of the Loveland hybrid circuit department, was indispensable. Special acknowledgments are due Bill Hewlett for his encouragement; Tom Whitney and Paul Stoft for their direction and help in administration and coordination between a number of departments; Tom Osborne for valuable discussions and suggestions; Allen Inhelder, Darrel Lauer, and Ed Liljenwall for industrial design—their consistent human engineering consideration is particularly valuable to a simple, elegant keyboard design; Dave Cochran for card read/write circuit design; Rich Whicker for the design of the card-reader control circuit; Bob Schweizer for valuable suggestions on the design of the read/write principle and circuits and for help on system breadboarding and simulator design; Clarence Studley and Bernie Musch for packaging and mechanical part design; Bob Taggart for his creative design of the card reader mechanism, with the able assistance of Dick Barth; John Bailey for help on the motor and head switches; Bill Boller for his close liaison with Manufacturing Division; Homer Russell and his team for applications software support. Special thanks are also due the tool designers of Manufacturing Division, whose creative, outstanding tools made the state-of-the-art plastic parts for the HP-65 possible.



Chung C. Tung

Chung Tung received his BSEE degree from National Taiwan University in 1961 and his MSEE degree from the University of California at Berkeley in 1964. As a member of HP Laboratories from 1965 to 1972, he was involved in the design of the 9100 Calculator and the HP-35 and HP-45 Pocket Calculators. He subsequently joined the Advanced Products Division as manager of the HP-65 project and is now engineering section manager there. Chung has done work towards his PhD degree at Stanford University, although he isn't presently enrolled there. He is married, has a son and a daughter, and lives in Santa Clara, California. Music, reading, and table tennis are his principal modes of relaxation.

SPECIFICATIONS

HP-65 Programmable Pocket Calculator

PRE-PROGRAMMED FUNCTIONS:
- TRIGONOMETRIC: sin x • arc sin x • cos x • arc cos x • tan x • arc tan x
- LOGARITHMIC: log x • ln x • e^x • 10^x
- OTHER: y^x • $\sqrt{x}$ • 1/x • π • x^2 • n! • conversion between decimal angle, degrees/minutes/seconds, radians or grads • rectangular/polar coordinate conversion • decimal/octal conversion • degrees(hours)/minutes/seconds arithmetic • integer/fraction truncation

OTHER FUNCTIONS:
- REGISTER ARITHMETIC: Addition, subtraction, multiplication or division in serial, mixed serial, chain or mixed chain calculations

FEATURES:
- DISPLAY: Up to 10 significant digits plus 2-digit exponent and appropriate signs
- DYNAMIC RANGE: 10^{-99} to 10^{99}
- Primary functions activated by single keystroke; alternate functions use prefix keys
- Five user definable keys
- Four-register operational stack
- Program memory for storage of up to 100 steps
- Single step running and/or inspection of a program
- Insert/delete editing features
- Nine addressable memory registers
- "Last X" register for error correction and number reuse
- Two flags for skip or no-skip programming or branching to another part of program
- $x \neq y$, $x \leq y$, $x = y$, $x > y$ relational tests
- Magnetic card reader/writer
- Built-in counter
- Automatic decimal point positioning
- Selective round-off; range: 0-9 decimal places
- Two display modes: fixed point and scientific
- Indicators for improper operations and low battery condition
- Operates on rechargeable batteries or ac
- Light-emitting diode (LED) display

APPLICATION PROGRAM PACS:
- Standard Pac (17 miscellaneous programs, 2 diagnostics, 1 head cleaning card, 20 blank cards)
- Math Pac I (40 mathematical programs)
- Math Pac II (33 mathematical programs)
- Stat Pac I (37 statistical programs)
- Survey Pac I (34 surveying programs)
- Medical Pac I (33 medical programs)
- EE Pac I (35 electrical engineering programs)

POWER:
- AC: 115 or 230 V, ±10%, 50 to 60 Hz, 5 watts.
- BATTERY: 500 mW derived from nickel-cadmium rechargeable battery pack.

WEIGHT: 11 ounces (312 g) with battery pack. Recharge: 5 ounces (155 g). SHIPPING WEIGHT: approx. 3 lbs (1.4 kg).

DIMENSIONS:
- LENGTH: 5.8 in (14.7 cm).
- WIDTH: 3.2 in (8.1 cm).
- HEIGHT: 0.7 to 1.4 in (1.8 to 3.4 cm).

TEMPERATURE RANGES:
- OPERATING: 32°F to 104°F (0°C to 40°C).
- CARD READER: 50°F to 104°F (10°C to 40°C).

PRICE IN U.S.A.: $795. Includes rechargeable battery pack, 115/230V ac adapter/recharger, soft carrying case, safety travel case, owners handbook and quick reference guide, program forms, standard pac of prerecorded cards, HP-65 Newsletter, and 1-year subscription to Catalog of Contributed Programs.

MANUFACTURING DIVISION: ADVANCED PRODUCTS DIVISION
19310 Pruneridge Avenue
Cupertino, California 95014 U.S.A.

Programming the Personal Computer

Wherein are revealed the functions of the keys, how problems are solved, and a bit of what goes on inside.

by R. Kent Stockwell

THE HP-65 CALCULATOR uses the same reverse Polish keyboard language, the same four-register operational stack, and the same architecture as its predecessors, the HP-35,[1] the HP-45, and the HP-80.[2] It also has two important features that are new to hand-held calculators. One is its greatly expanded function set, and the other is programmability, complete with conditional and unconditional branching, user-definable functions, and magnetic-card program storage.

Function Set

Thirteen HP-65 keys are for data entry. These are the digits 0 to 9, the decimal point, CHS (change sign), and EEX (enter exponent). Numbers may be entered with or without a power-of-ten exponent.

Keyed-in digits set the value of the X register, which is also the display, in the four-register operational stack.* The CLx (clear x) key allows corrections. Any other key except SST and R/S terminates entry of a number.

The four arithmetic functions (+, −, ×, ÷) operate on x and y, the contents of the X and Y registers. Operands are loaded into the stack with the ENTER↑ key; they may then be operated upon by the function keys. Operations execute immediately and results appear in X.

Thirty-three other functions derive from using three prefix keys (f, f^{-1}, g) to condition eleven suffix keys (digits 0-9 and decimal point). The two gold-colored prefix keys, labeled f and f^{-1}, access the functions printed in gold above the suffix keys and the inverses or complements of these functions. The blue prefix key, g, accesses the functions printed in blue on the angled lower side of the suffix keys. (The no-prefix meanings of the suffix keys appear on their top faces.) All of these functions execute immediately, operating on x, or x and y, or the entire operational stack. Thus, for example, the key sequence f 4 obtains sin x in the display, f^{-1} 4 obtains $\sin^{-1} x$, and g 4 obtains 1/x.

Computations requiring more data storage than is provided by the operational stack may use any of nine data storage registers. For example, pressing STO 4 stores x into register four, leaving x unchanged. Pressing RCL 4 recalls r_4 to X, leaving r_4 unchanged. Arithmetic accumulation to any storage register is accomplished by inserting the desired operation key between STO and the digit key that addresses the register. Thus the key sequence STO <arithmetic operator><digit n> gives r_n<arithmetic operator>x in register R_n and leaves x in the display.

The user can change the display format as required by the particular problem. The key sequence DSP <digit n> rounds the display value to n digits after the decimal point in scientific notation,* while DSP . <digit n> results in an absolute display rounded to n digits following the decimal point. For example, 12.366 gives 1.24 01 in DSP 2 mode and 12.37 in DSP . 2 mode. Display rounding does not affect internal values.

All functions involving angles, that is, sin, cos, tan, R→P (rectangular to polar conversion), → D.MS (conversion to degrees, minutes, seconds), and the inverses of these functions accept arguments or produce results in degrees, radians, or grads, set by the key sequence g ENTER↑ or g CHS or g EEX, respectively. These settings remain in effect until changed.

On the theory that users should be able to correct key-sequence errors with minimal effort, any prefix key overrides any previous prefix key, and the sequence f ENTER↑ clears any prefix keys. Thus, for example, the key sequence STO + f g g 4 gives 1/x,

*Capital letters are names of registers and lower-case letters are register contents.

*One digit to the left of the decimal point with power-of-ten exponent, e.g., 2.54×10^{12}.

while g f ENTER↑ 4 gives the value 4 in the display.

By now it must be clear how key conditioning with color-coded keys and legends has been used to provide access to many functions with a limited number of keys on a small keyboard. Although another level of conditioning would further expand the function set (e.g., f g 4 or f^{-1} g 4 or g f 4 could possess functional meanings), this would greatly increase keyboard complexity, keyboard busyness, and internal control programming. For these reasons, most of the key conditioning remains at the one-prefix level.

HP-65 functions are listed on page 14. Fig. 1 shows an example of a problem solution.

Programming

All operations described so far apply when the switch in the upper right-hand corner of the HP-65 keyboard is in the RUN position. When this switch is in the W/PRGM position, the keystrokes are stored in the 100-step program memory instead of being executed. Twenty-five frequently used two-keystroke sequences merge into a single memory step; thus the program memory may actually contain more than 100 keystrokes.

Problem:

Evaluate $V_B - \frac{kT}{q}\ln\left(\frac{I_D}{I_S} + 1\right) - RI_D$

for V_B = 8 volts, kT/q = 0.026 volts, $I_D = 6 \times 10^{-3}$ amperes, $I_S = 10^{-10}$ amperes, R = 1200 ohms

Solution:

	Stack Registers			
Keystrokes	Display X	Y	Z	T
8	8.			
ENTER↑	8.00×10^{0}	8		
.026	.026	8		
ENTER↑	2.60×10^{-2}	.026	8	
.006	.006	.026	8	
ENTER↑	6.00×10^{-3}	.006	.026	8
EEX 10 CHS	10^{-10}	.006	.026	8
÷	6.00×10^{7}	.026	8	8
1	1	6×10^{7}	.026	8
+	6.00×10^{7}	.026	8	8
f ln	1.79×10^{1}	.026	8	8
×	4.66×10^{-1}	8	8	8
−	7.53×10^{-1}	8	8	8
1200	1200	7.53×10^{-1}	8	8
ENTER↑	1.20×10^{3}	1200	7.53×10^{-1}	8
.006	.006	1200	7.53×10^{-1}	8
×	7.20×10^{0}	7.53×10^{-1}	8	8
−	3.34×10^{-1}	8	8	8

Calculator in DSP 2 Mode

Fig. 1. *An example of HP-65 use as a scientific calculator.*

STO 1	RCL 1	g R↓
STO 2	RCL 2	g R↑
STO 3	RCL 3	g x⇄y
STO 4	RCL 4	g LSTx
STO 5	RCL 5	g NOP
STO 6	RCL 6	g x≠y
STO 7	RCL 7	g x≤y
STO 8	RCL 8	g x=y
		g x>y

Fig. 2. *User programs may have as many as 100 steps. These twenty-five keystroke sequences merge into a single step. Thus programs may contain more than 100 keystrokes.*

The memory itself contains no absolute addresses. Instead, it is a circulating shift register organized into six-bit words. One word is a marker that denotes the boundary between the beginning and the end of the memory. Another word is a pointer which denotes the last step executed in run mode, and the last step filled in program mode. As a program runs, this pointer is moved down through memory. Branching is accomplished by moving the pointer to the location of the destination label. User-defined function calls are implemented by leaving the main pointer at the call and activating a second pointer at the function location (see Fig. 3). When the return to the calling location occurs, the second pointer is deactivated and the first pointer reactivated. Neither the marker nor the pointers subtract from the 100 user steps.

Programs may contain three types of tests to allow conditional execution of all operations. These are x-y comparisons (x≠y, x≤y, x=y, x>y), four flag tests

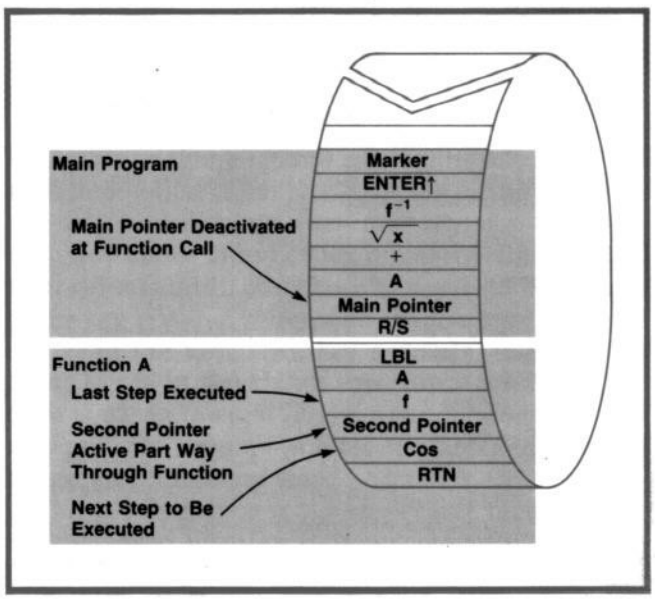
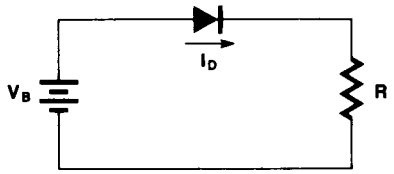


Fig. 3. *The program memory circulates continuously, its beginning and end denoted by a marker. The main pointer moves as programs are entered or executed. A second pointer is activated when a user-defined function is called.*

Problem:

Find the diode current I_D in the circuit shown. Also find its sensitivity with respect to V_B and R, i.e., $\partial I_D/\partial V_B$ and $\partial I_D/\partial R$.



Equations:

$$V_B = \frac{kT}{q}\ln\left(\frac{I_D}{I_S} + 1\right) + RI_D$$

$$\frac{\partial I_D}{\partial V_B} = \left[\frac{kT}{q}\left(\frac{1}{I_D + I_S}\right) + R\right]^{-1}$$

$$\frac{\partial I_D}{\partial R} = -I_D\left[\frac{kT}{q}\left(\frac{1}{I_D + I_S}\right) + R\right]^{-1}$$

I_S = diode saturation current in amperes
R = resistor value in ohms
V_B = battery voltage in volts
kT/q = thermal voltage in volts

Algorithm:

For Newton-Raphson iteration,

$$I_D(n+1) = I_D(n) - \frac{f[I_D(n)]}{f'[I_D(n)]}$$

where $I_D(n)$ = nth guess
$f[I_D(n)]$ = function evaluated for nth guess
$f'[I_D(n)]$ = first derivative of function, evaluated for nth guess
$I_D(n+1)$ = (n + 1)st guess

Let $f(I_D) = V_B - \frac{kT}{q}\ln\left(\frac{I_D}{I_S} + 1\right) - RI_D$

Then $f'(I_D) = -\left[\frac{kT}{q}\left(\frac{1}{I_D + I_S}\right) + R\right]$

Specify convergence criterion: if $|I_D(n+1) - I_D(n)| < C$ the algorithm halts.

Program halts after ten iterations. The user may then start ten more iterations.

Example:

$I_S = 10^{-10}$ A
R = 1.2 kΩ
V_B = 8 V
kT/q = 0.026 V
$C = 10^{-9}$ A

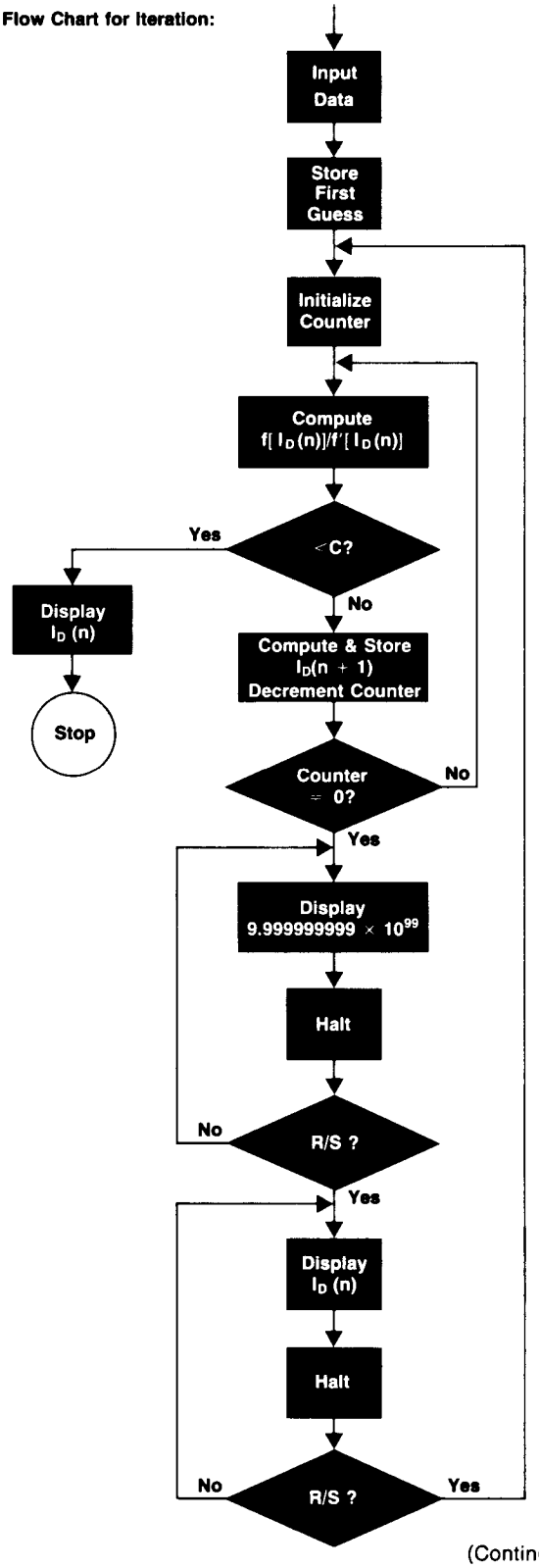
Load card and follow user instructions.

Results:

I_D = 6.278A
$\partial I_D/\partial V_B$ = 0.8305 mA/V
$\partial I_D/\partial R$ = −5.213 μA/Ω

Time required to compute I_D (step 3): 11 seconds.

Flow Chart for Iteration:



(Continued)

Fig. 4. *An example of HP-65 programming. A common problem in many disciplines is the solution of irreducible equations, such as x = 5 ln x. Finding the answer requires a clever first guess at the solution and, based on the results of the first guess, an even more clever second guess, and so on. The iterative procedure, tedious if done manually, can often be automated. In this example the Newton-Raphson method is used to solve an electrical engineering problem.*

Program:

HP-65 Program Form

Title: Diode Current Iteration — Page 1 of 2

KEY ENTRY	CODE SHOWN	COMMENTS
LBL	23	Compute $f(I_D)$
D	14	
RCL 6	34 06	
RCL 1	34 01	
RCL 2	34 02	
RCL 3	34 03	
+	81	
1	01	
+	61	
÷	31	
ln	07	
×	71	
−	51	
RCL 4	34 04	
RCL 2	34 02	
×	71	
−	51	
RTN	24	Leaves $f(I_D)$ in X
LBL	23	Compute $f'(I_D)$
E	15	
RCL 1	34 01	
RCL 2	34 02	
RCL 3	34 03	
+	61	
÷	81	
RCL 4	34 04	
+	61	
CHS	42	
RTN	24	Leaves $f(I_D)$ in X, old X in Y
LBL	23	Iterate for I_D
A	11	
EEX	43	First guess = 10^{-3} AMPS
CHS	42	
3	03	
STO 2	35 02	
LBL	23	Initialize counter
1	01	
1	01	
0	00	
STO 8	35 08	
LBL	23	Iterate
2	02	
D	14	
E	15	
÷	81	$f(I_D)/f'(I_D)$ in X
9	09	
ABS	04	
RCL 6	34 06	
$\geq$	36	$\lvert f(I_D)/f'(I_D)\rvert < C$?
GTO	24	
RCL 2	34 02	

KEY ENTRY	CODE SHOWN	COMMENTS
RTN	24	Done, I_D in X
RCL 2	34 02	Update I_D
LSTx	35 00	
−	51	
STO 2	35 02	
9	09	Check no. iterations
DSZ	83	
GTO	22	
2	02	
9	09	10 iterations done
0	00	
+	31	
TAN	06	
R/S	84	Display $9.999999999 \times 10^{99}$
RCL 2	34 02	
R/S	84	Display current I_D
GTO	22	
1	01	Iterate 10 more times
LBL	23	Compute $\partial I_D/\partial V_B$
B	12	
E	15	
CHS	42	
9	35	
1/x	04	
RTN	24	
LBL	23	Compute $\partial I_D/\partial R$
C	13	
RCL 2	34 02	
E	15	
÷	81	
RTN	24	
NOP	35 01	
NOP	35 01	

Note: D+E called as functions by the iteration program, C. Thus, to define a new problem simply redefine $f(I_D)$ and $f'(I_D)$ functions D+E and, if required, the first guess in function A.

REGISTERS: R1 kT/q; R2 $I_D(n)$; R3 I_S; R4 R; R5 V_B; R6 C; R7; R8 Counter; R9 Scratch, X ? Y

LABELS: A I_D; B $\partial I_D/\partial V_B$; C $\partial I_D/\partial R$; D $f(I_D)$; E $f'(I_D)$; 0; 1; 2; 3; 4; 5; 6; 7; 8; 9

FLAGS: 1; 2

HP-65 User Instructions

Title: Diode Current Iteration — Page 2 of 2

Programmer: R. K. Stockwell — Date: 3/6/74

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Enter Card 1			
2	Inputs (Any Order)			
	Thermal Voltage	kT/q, Volts	STO 1	
	Saturation Current	I_S, AMPS	STO 3	
	Resistance	R, OHMS	STO 4	
	Battery Voltage	V_B, Volts	STO 5	
	Convergence Criterion	C, AMPS	STO 6	
3	Compute Diode Current		A	I_D, AMPS
	If display $9.999999999 \times 10^{99}$,			
	do 4 and 5,			
	otherwise skip to 6			
4	Display Present Diode Current		R/S	I_D, AMPS
5	Continue (Go to 3, iterate ten more times)		R/S	
6	Either calculate voltage sensitivity		B	$\partial I_D/\partial V_B$, AMPS/VOLTS
	or			
	calculate resistance sensitivity		C	$\partial I_D/\partial R$, AMPS/OHMS
	or			
	calculate $f(I_D)$		D	$f(I_D)$, Volts
	or			
	calculate $f'(I_D)$		E	$f'(I_D)$, VOLTS/AMPS
	or			
	go to 2 and re-enter any or all inputs for a new problem.			

(there are two flags, each of which may be set or cleared and then tested for set or clear), and decrement and skip if zero (DSZ). Except for DSZ, each test, if false, causes program control to skip the next two memory steps; otherwise, execution continues normally. The DSZ operation decrements data-storage register R_8 by one, using integer arithmetic, and if the result is zero, program control skips the succeeding two steps.

Literal labels with the GO TO function implement branching. Thus LBL<n> is the destination for GTO <n>, where n is a digit or a key A-E in the top row.

The HP-65 user may store two types of programs in the program memory. First, he may precede a section of memory containing various functions with LBL <m>, where m is A, B, C, D, or E, and terminate the section with RTN (return). Thereafter, pressing key A, B, C, D, or E in the RUN mode causes that memory section to execute immediately. Any or all of keys A to E may be defined but the sum of memory steps for all functions cannot exceed 100. These user-defined functions behave exactly like the preprogrammed functions described earlier, yet the user may create the functions to fit his special needs.

The user's second option is to precede a block of code with a label definition and terminate it with the R/S (Run/Stop) key. In RUN mode this key stops an executing program; if no program is running, pressing the R/S key starts execution. Pressing GTO<label name>R/S then starts program execution, and the program halts at the R/S in memory. If the program starts at the beginning of memory no label is needed; in RUN mode control can be transferred to the beginning of memory by pressing RTN. Programs defined in this way may call any of the functions A through E; the desired key is simply entered into the program definition.

The SST (single step) and DEL (delete) functions implement debugging and editing. In W/PRGM mode, each depression of SST advances the memory pointer one step and displays each memory step as a two-digit key code. These codes represent digit keys by their values and all other keys by a row-column index of the key position referenced to the upper left-hand corner of the keyboard. For example, the decimal-point key is in the eighth row, third column, so its code is 83. In the RUN mode, each depression of SST advances the memory pointer one step and executes the adjacent memory step.

The key sequence g CLx in W/PRGM mode deletes the displayed memory step and moves up the next step to fill the gap. Any keys entered in W/PRGM mode are automatically inserted following the displayed memory step. Thus the replacement operation consists of a delete operation followed by the desired key.

The sequence f CLx clears the entire memory.

Programs can be stored on magnetic cards for later use. Cards can be recorded and rerecorded as many times as desired. To protect a recorded program on a card, further recording can be prevented by clipping the notched tab on the upper left corner of the card. Users may write on the card and place it in a slot above the keys A through E, thereby labeling any specially defined keys.

Fig. 4 shows an example of HP-65 programming.

Firmware

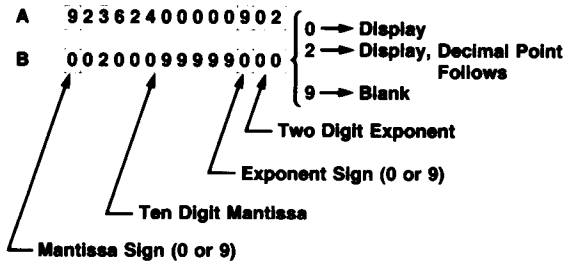
To direct the various computational and control functions of the HP-65, 3072 words of read-only memory (ROM) are used. Each ROM word contains ten bits and constitutes a calculator microinstruction. Microinstructions grouped together in blocks perform the various external functional tasks of the calculator. A task may require one block of words or several blocks woven together. For example, the CLx function requires only a few words, while the sin function uses the tan function, which uses the add function, and so on.

Although production of efficient microcode is an iterative process, the first step is the choice or design

HP-65 Normalization Routine

User input form is stored in two processor registers, A and B.

Example: -23.624×10^{-2}



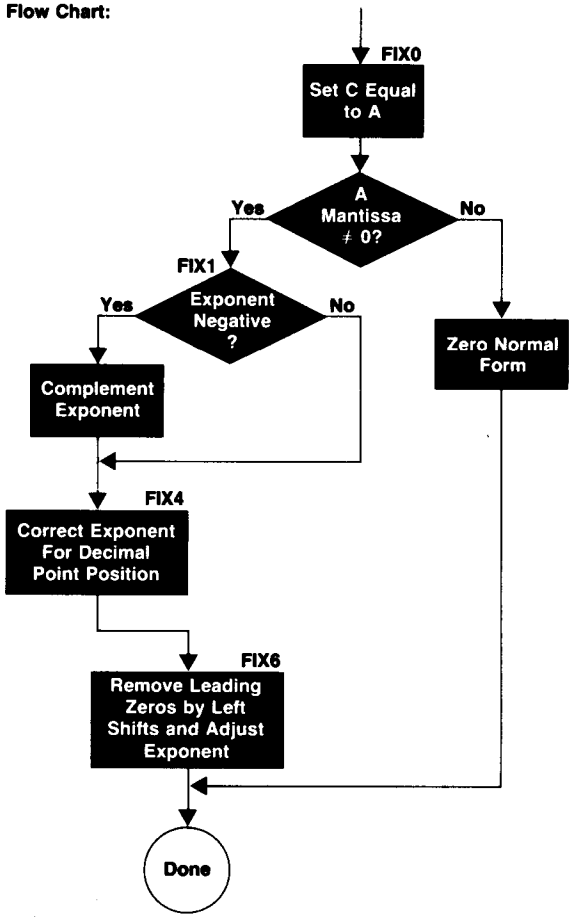
This is converted to internal normal form in a third register, C. For computational efficiency this is mantissa in sign-magnitude form with one digit to the left of an implied decimal point and exponent in ten's complement form (see ref. 4). In the mantissa sign position, 9 represents minus, 0 represents plus. Thus, -23.624×10^{-2} would store internally as

9 2 3 6 2 4 0 0 0 0 9 9 9

Program Listing:

Step Number	ROM Address	ROM Code	ROM Subroutine Addresses	Labels	Program Statements
51	L01063	....111.		FIX5	P - 1 -> P
52	L01064	.1111.1.1			C + 1 -> C[X]
53	L01065	.111.11.1	-> L1073		JSB FIX7
		:			:
56	L01070	..11.11.1		FIX3	0 -> C[XS]
57	L01071	11.1..11.		FIX4	13 -> P
58	L01072	.1.11.1.1			C - 1 -> C[X]
59	L01073	.......1.		FIX7	IF B[P] = 0
60	L01074	..11..1111	-> L1063		THEN GO TO FIX5
61	L01075	11....11.			12 -> P
62	L01076	1..11...1.		FIX6	IF A[P] >= 1
63	L01077	.11111..11	-> L1174		THEN GO TO FIX2
64	L01100	.1.....11.			SHIFT LEFT A[M]
65	L01101	.1.11.1.1.			C - 1 -> C[X]
66	L01102	..11111..1	-> L1076		JSB FIX6
		:			:
102	L01146	.111111.1.		FIX1	C + 1 -> C[XS]
103	L01147	...111...11	-> L1070		IF NO CARRY GO TO FIX3
104	L01150	...1.1.1.1.			0 - C -> C[X]
105	L01151	...111..1.1	-> L1071		JSB FIX4
		:			:
119	L01167	111.1.111.		FIX0	A EXCHANGE C[W]
120	L01170	.11...111.			C -> A[W]
121	L01171	1..11...11.			IF A[M] >= 1
122	L01172	.11..11.11	-> L1146		THEN GO TO FIX1
123	L01173	1.111...11.			0 -> A[M]
124	L01174	111.1...11.		FIX2	A EXCHANGE C[M]

Flow Chart:



Program Listing Notes:

Subscripts, e.g., C + 1 → C[X], refer to various register fields for arithmetic operation (see ref. 1).

S = Mantissa Sign
MS = Mantissa and Sign
M = Mantissa
XS = Exponent Sign
X = Exponent and Sign
P = Pointer
WP = Word up to and Including Pointer, Right to Left

Fig. 5. *An example of the HP-65's internal microprogramming. Even such a seemingly trivial operation as digit entry requires careful design so it seems trivial to the user. Values must be displayed as keyed in, yet be normalized to a standard internal form. This is the normalization routine and the flow chart and ROM listing for it.*

of an algorithm. This may involve such constraints as accuracy, execution speed, microinstructions required, or even available design time. Next, a functional flow chart is drawn to outline the sequence of various operations and any conditional operations. This flow chart is then expanded to sufficient detail that it can be translated to microinstructions and implemented on a calculator simulator. More often than not there are implementation errors to correct; sometimes the entire algorithm is faulty, requiring a new design. When the design is complete, integrated-circuit read-only memories are produced.

Where possible, the HP-65 uses the proven algorithm implementations from the HP-35 and HP-45 (trigonometric, logarithmic, and exponential routines). This saved development time and reduced implementation error probabilities.

Many HP-65 algorithms would provide interesting descriptions here, but one that demonstrates appreciable complexity is the digit-entry routine. Designing this seemingly trivial function so as to seem trivial to the user required considerable patience and careful thought. Usually, any entry will produce an undesirable result unless the designer specifically accounts for it. Values must be displayed as keyed in, yet they must be normalized to some internal form. The table below lists some of the design constraints on this algorithm.

USER ACTION	DESIRED RESULT
More than ten mantissa digits	Ignore all digits after tenth
First key of new entry	Overwrite existing x if key follows ENTER↑ or CLx; otherwise do automatic ENTER↑
Extra digits after EEX	Shift exponent left; new digit becomes least significant digit of exponent.
Multiple decimal point	Ignore all decimal points after first
Decimal point after EEX	Ignore
Leading zeros keyed in	Accept and display leading zeros, zero normal form.
EEX first key of new entry	Enter one in mantissa; following digits enter exponent.
Decimal point first key of new entry	Display only decimal point; zero normal form.
Digits after decimal point	Continue appending digits; no effect on internal exponent
Digits before decimal point	Continue appending digits, increment internal exponent.
following	Complement exponent sign
Multiple	Complement mantissa sign, or exponent sign if has been pressed.

Such an algorithm was explained in a previous issue.[3] Fig. 5 shows the flow chart and ROM listing for the normalizing routine.

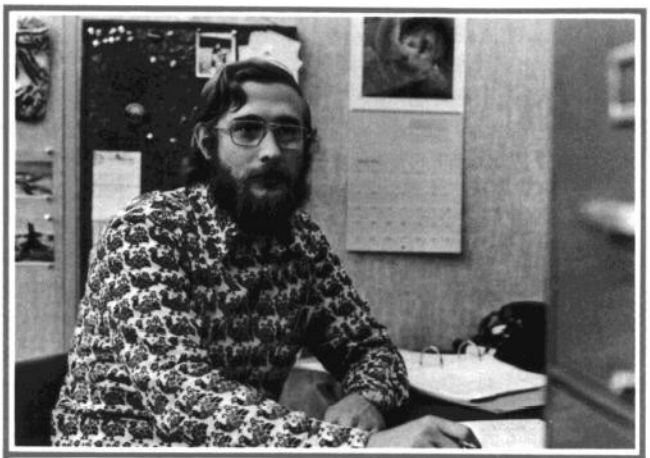
Acknowledgments

Many people of course, contributed ideas to this effort. Particular acknowledgment is due the following: Paul Stoft and Tom Whitney for bringing together the necessary technical resources and people; Dave Cochran, for the trigonometric and exponential routines used in the HP-35, and for help in understanding the HP-35 architecture; Francé Rodé for further explanations of the HP-35 architecture; Peter Dickinson for suggestions and criticisms concerning algorithm implementations, particularly the extension of the HP-35 algorithms; Tom Osborne for helpful advice and suggestions regarding the function set and the external behavior of the HP-65; Homer Russell and Wing Chan for helpful suggestions and criticisms for the function set, and for

patiently keeping up with numerous daily changes; Steve Walther for providing the microinstruction language compiler; Darrel Lauer and Al Inhelder for crystallizing the keyboard layout from a myriad of suggestions; Ed Heinsen and Lynn Tillman for extending the simulation software to accommodate the increased complexity of the HP-65.

References

1. T.M. Whitney, F. Rodé, and C.C. Tung, "The 'Powerful Pocketful': an Electronic Calculator Challenges the Slide Rule", Hewlett-Packard Journal, June 1972.
2. W.L. Crowley and F. Rodé, "A Pocket-Sized Answer Machine for Business and Finance", Hewlett-Packard Journal, May 1973.
3. D.S. Cochran, "Internal Programming of the 9100A Calculator", Hewlett-Packard Journal, September 1968.
4. M.M. Mano, "Computer Logic Design", Prentice-Hall, 1972, chapter 1.



R. Kent Stockwell

Kent Stockwell joined HP four years ago. As a member of HP Laboratories for most of that period, he's done program development, modeling, and numerical analysis for computer-aided circuit design and, more recently, the firmware development for the HP-65. Kent studied electrical engineering at Massachusetts Institute of Technology, graduating in 1970 with SB and SM degrees. A native of Kalamazoo, Michigan, he now lives in Palo Alto, California, where he's currently remodeling his house and putting his woodworking skills to good use. He also plays trombone and baritone horn, and enjoys backpacking in the mountains of California and Colorado.

APPENDIX

HP-65 Programmable Pocket Calculator Functions and Operations

Arithmetic
- add
- subtract
- multiply
- divide

Logarithmic
- natural logarithm (base *e*)
- natural antilogarithm (base *e*)
- common logarithm (base 10)
- common antilogarithm (base 10)

Trigonometric
- set operating mode (degrees, radians, or grads)
- sine
- arc sine
- cosine
- arc cosine
- tangent
- arc tangent
- add or subtract degrees/minutes/seconds
- convert angle from degrees, radians, or grads to degrees/minutes/seconds and vice versa
- convert polar coordinates to rectangular coordinates and vice versa

Exponential
- square
- square root
- raising a number to a power (y^x)
- reciprocal (can be used with y^x function to extract nth roots)

Other Preprogrammed Functions and Operations
- extract integer or decimal portion of a number
- factorial
- recall value of π to 10 significant digits
- convert decimal-base integers to or from octal-base integers
- "roll down" or "roll up" numbers in operational stack
- clear display
- clear operational stack
- clear all nine addressable memory registers
- recall last input argument from separate "last-x" storage register
- store or recall numbers from any of the nine addressable memory registers
- register arithmetic
- display formatting

Program Structure and Edit Functions
- clear program memory
- user-definable keys (A-E)
- label
- go-to
- return
- run/stop
- no-operation
- set flag 1
- test flag 1
- set flag 2
- test flag 2
- $x = y$
- $x \neq y$
- $x \leq y$
- $x > y$
- decrement and skip on zero
- delete program step
- single-step

Designing a Tiny Magnetic Card Reader

Here's how it was designed and how it works.

by Robert B. Taggart

ONE THING WE HAD WAS an abundance of ideas for tiny card readers. The HP-65 card reader project began in the electronic research laboratory of HP Laboratories before the introduction of the HP-35. The basic design goal for the card reader was to propel a magnetic card the size of a piece of chewing gum at a constant speed of 3¾ to 6½ cm/s. Of course, it also had to fit inside the HP-35 package.

Many schemes were tried, including music box mechanisms, hand feed, gravity feed, and dashpot systems. Motor-driven schemes didn't generate much enthusiasm at first. But one day while digging through a file on motors that was about to be discarded, I found a brochure describing a Swiss-made motor less than 1½ cm in diameter. Within a day we obtained two samples. They had ample torque and their unique construction provided very low brush noise. The motors had a tested lifetime of over one thousand hours.

These tiny motors turn at speeds in excess of 10,000 r/min. The problem of reducing this high speed down to 6 cm/s was solved by using a worm gear, which provides a large speed reduction in one stage. Other schemes were tried but the worm and wheel combination proved best.

Gripping of the card under all conditions was another problem that had to be solved. The card is very small and would be handled extensively. Even grease could not be allowed to stand between a card and good gripping. Plastics and rubbers of various kinds and textures were tried. We tried to put tire-like treads into the rubber and even put ridges in the card for better grip. But finally we found a polyurethane rubber with the right texture that grips even when the card is coated with oil.

The major problems we faced were caused by trying to squeeze so much information onto such a short card. If the card could have been longer the design would have been much easier. However, having a short card offers the user the convenience of labeling the top row of keys with the program card. This restricted the length of the card to less than the width of the calculator. All kinds of things become critical in trying to read and write on short magnetic cards in such a small machine at 300 or more bits per inch. Azimuth alignment of the magnetic head at 400 bits per inch must be accurate within ±¼ degree. The best way to align heads with this precision is under a microscope. But can you align a head to this kind of accuracy in a molded plastic part? Based on the HP Manufacturing Division's confidence in the plastic we decided to try. We succeeded, thereby achieving a significant cost saving over using a metal frame. Using such a short card required bunching the magnetic head very close to the drive roller and gear train. This unusual geometry combined with the tight tolerances of ±0.001 inch on some dimensions made the reader frame a complex challenge.

The short card and higher bit density created numerous problems related to keeping the vibration of the drive train to a minimum. Of all the problems we faced this proved to be the most difficult. Finding the right process for making the worm gear and refining that process to a high degree made it possible. A method was developed to couple the worm gear to the motor and all these ideas combined to provide the necessary precision and smoothness. The bit-to-bit speed variation was held to less than 10%.

Another challenging problem involved inventing the set of switches that turn on the motor when the card is inserted, then turn on the magnetic head when the card is over the head, and finally provide file protection when the card corner is cut off. Three switches are provided. Some of these serve double duty in that they help wrap the card over the magnetic head for better head contact. The geometry of the switches

is unusual and many new ideas were required to fit three switches, a gear train, a drive roller, a backup roller, and magnetic head pole tips in a volume of 1¼ × 1¼ × 2½ cm. This very tight spacing was made necessary by the short card length and high bit density.

The short card and the fact that the recording rate is controlled by the clock in the calculator requires that the speed of the card be kept nearly constant under conditions of varying temperature and humidity. For instance, with a slow clock and a fast card reader it is conceivable that the card might go through the machine before all the program could be recorded on it. Conversely, if the clock were fast and the card slow the bit density could exceed the maximum permitted by the head alignment.

To solve this problem the voltage across the motor is regulated and each machine's card speed is set within ±2% by an external trimming resistor on the bipolar circuit that sets the voltage across the motor. We had decided at the outset to use as large a motor as would possibly fit to maximize the amount of available torque. The greater the available stall torque the less sensitive the card reader will be to changes in load. This eliminates the need for feedback speed control.

Speed control turned out to be a very nasty problem particularly at low temperature. It was discovered that at freezing temperatures the polyurethane rubber becomes hard as a rock. This increases the current drain on the motor enormously and reduces the card speed to almost zero. Numerous other types of rubber were tried, many of which remained softer at low temperature, but none of which gripped as well as one polyurethane composition. Eventually by increasing the thickness of the polyurethane we reduced the magnitude of the problem. It was at this point that we appreciated the unique gripping properties of this type of polyurethane.

How the Card Reader Works

Fig. 1 is a diagram of the card reader. As the card is inserted into the right side of the machine, it is forced against one edge of the card slot by one of two tiny leaf springs. This helps align the card with the magnetic head. Pushing the card farther into the machine causes it to activate the motor start switch when the card approaches the rubber drive roller. This turns the motor on.

Each of the three switches in the card reader is activated in the same way. The card displaces a nylon ball resting on the bottom of the card slot. Movement of the ball forces a tiny finger of copper to move upward. The end point of this copper switch finger makes contact with a contact pad on the underside of the keyboard printed-circuit board. The contact point of the switch moves a distance several times the thickness of the card to provide a reliable contact. Each switch is adjusted to the proper contact position during assembly.

When the bipolar motor circuit is turned on a precise voltage is fed to the motor terminals to establish the motor/card speed to within a few percent. The motor turns at a speed near 10,000 revolutions per minute. The motor is directly coupled by a tiny polyurethane sleeve to a miniature worm gear. The end of the worm gear rests against a thrust ball bearing and drives a helical gear. The helical gear is pressed onto the hub of the polyurethane rubber drive roller, which grips the card.

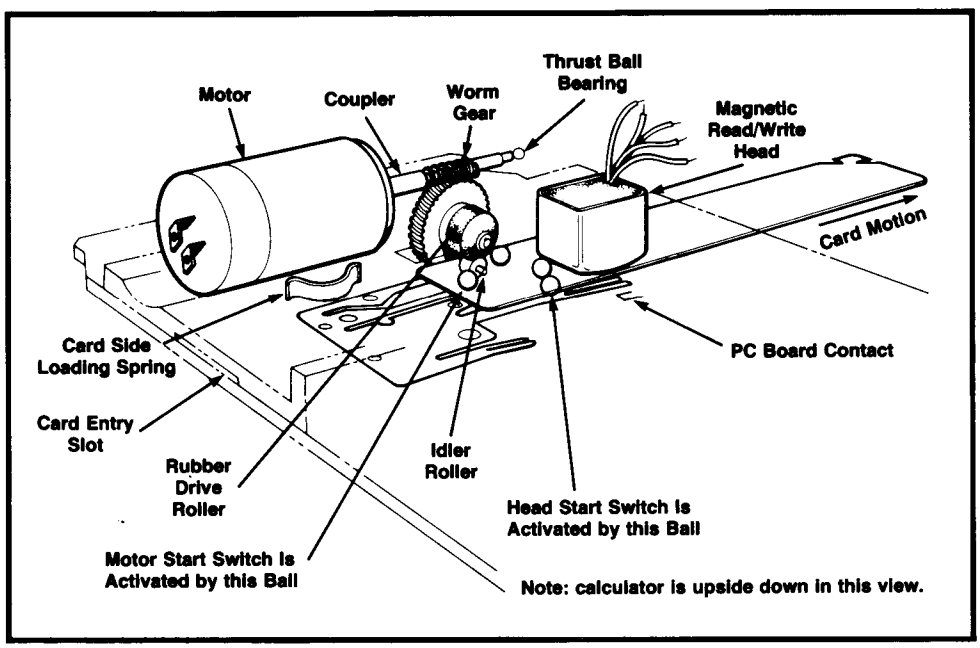


Fig. 1. *The HP-65 card reader.*

Once the motor is turned on by the motor start switch the user must push the card another small fraction of a centimeter so the rubber roller will grip the card. The card is then pinched and driven between the rubber roller and a fixed tiny idler roller made of nylon. The card proceeds through the machine toward the magnetic read/write head. Little bumps in the plastic support plate combine with the switch balls to wrap the card over the gap of the head.

As the card passes the magnetic gap of the head the leading edge of the card activates a second switch. This switch starts the write or read circuitry, depending on the position of the W/PRGM-RUN switch. Activation of this second switch lets the circuits know that the card is over the head and is assumed to be moving at the proper speed.

At nearly the same time that the second switch is activated, a third switch may or may not be activated depending on whether the corner of the card is cut off. This is the file protect scheme, which prevents the user from writing over a previous program. When the second switch is activated the third switch is interrogated. If the third switch is not activated (the corner being cut off) the machine will not write over that card when the calculator is in the W/PRGM mode. The data is written/read by a two-track recording scheme which is described elsewhere. As the card proceeds out the left side of the calculator it is held against the side of the card guide by a second side-loading leaf spring. When the trailing edge of the card passes the motor start switch the motor and read/write circuits are shut off. The card may then be removed from the machine.

Fig. 2 shows the card-reader parts disassembled.

Acknowledgments

The author would like to thank the following people for their contributions during the design stage: Dick Barth, Bob Schweizer, Bernie Musch, Clarence Studley, John Bailey, Craig Sanford, Bill Boller and the HP Manufacturing Division, Fred Rios and his model shop, the Advanced Products Division model shop, and the APD production tooling groups.

Robert B. Taggart

Bob Taggart is an engineering group leader at HP's Advanced Products Division. He holds a 1967 BS degree in mechanical engineering and a 1968 MS in biomedical engineering, both from Northwestern University, and the degree of Engineer from Stanford University (1970). In 1970 he joined HP Laboratories and for three years was involved in biomedical and pollution studies before moving to APD to develop the HP-65 card reader. Author of three papers on microwave antenna design and five patent applications (two granted), Bob is originally from Pompton Lakes, New Jersey. He skis, plays tennis, and enjoys white-water river running. He and his wife welcomed their first child, a boy, on March 31.

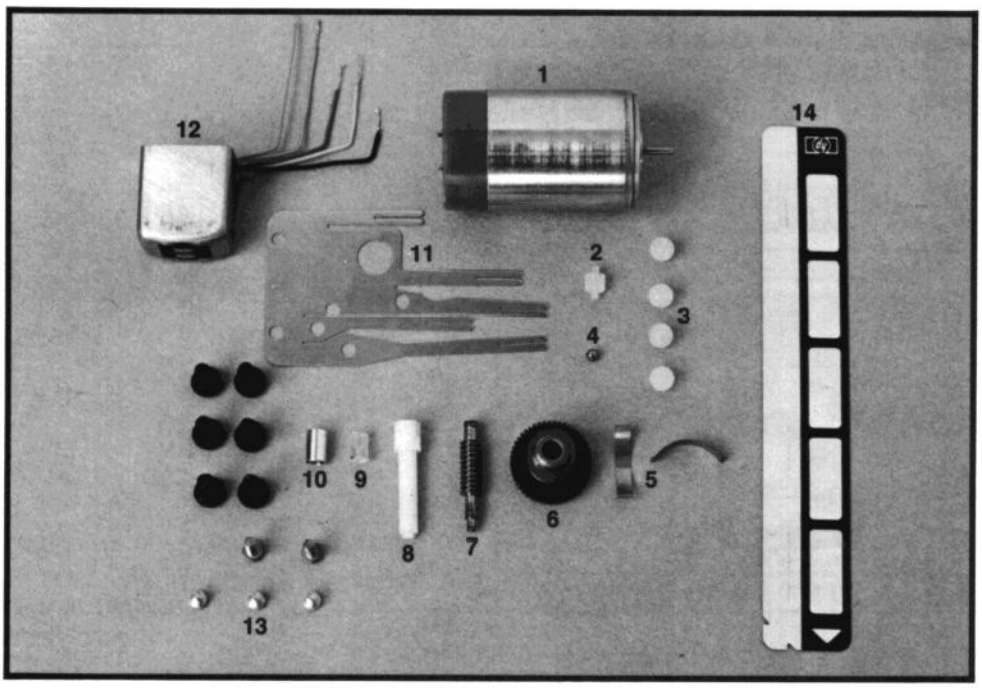


Fig. 2. *HP-65 card reader parts: 1. Motor 2. Idler roller 3. Switch activation balls 4. Thrust ball bearing 5. Card side loading springs 6. Rubber drive roller and helical gear 7. Worm gear 8. Drive pin 9. Coupler 10. Coupler sleeve 11. Switch contacts 12. Read/write head 13. Self-tapping switch adjustment screws 14. Magnetic card, 7.1 cm by 1 cm.*

Testing the HP-65 Logic Board

The board and its automatic test system are designed for rapid production testing and troubleshooting.

by Kenneth W. Peterson

DESIGNING AN INSTRUMENT for minimum cost means not only that parts and assembly costs are minimized. The time required for testing, troubleshooting, and repair is also critical and must be held to an absolute minimum. Yet too many products are designed without giving adequate consideration to the time that can be spent analyzing and locating any faults that may show up in production testing.

For a high-volume, complex product like the HP-65, computerized testing is the only feasible method. To aid the computer in diagnosing failures and isolating them to the responsible components, it's essential that access be provided to all of the product's pertinent nodes. The HP-65 logic board was designed with this in mind.

The layout of the HP-65 logic board is shown in Fig. 1. Input and output lines are brought out to two edges of the board. Additional test points are routed to a third edge, giving access to a total of 45 test points.

Test System

A block diagram of the computerized tester designed for the HP-65 logic board is shown in Fig. 2. The tester functionally compares the operation of a known-good unit with that of a test unit. All outputs from the test logic board are interfaced to voltage-comparison circuits to check for proper logic-level thresholds.

As the tester exercises the reference and test units with identical inputs (either on the key lines or on the card reader lines), the units' outputs are captured in two pattern storage registers on each clock time.

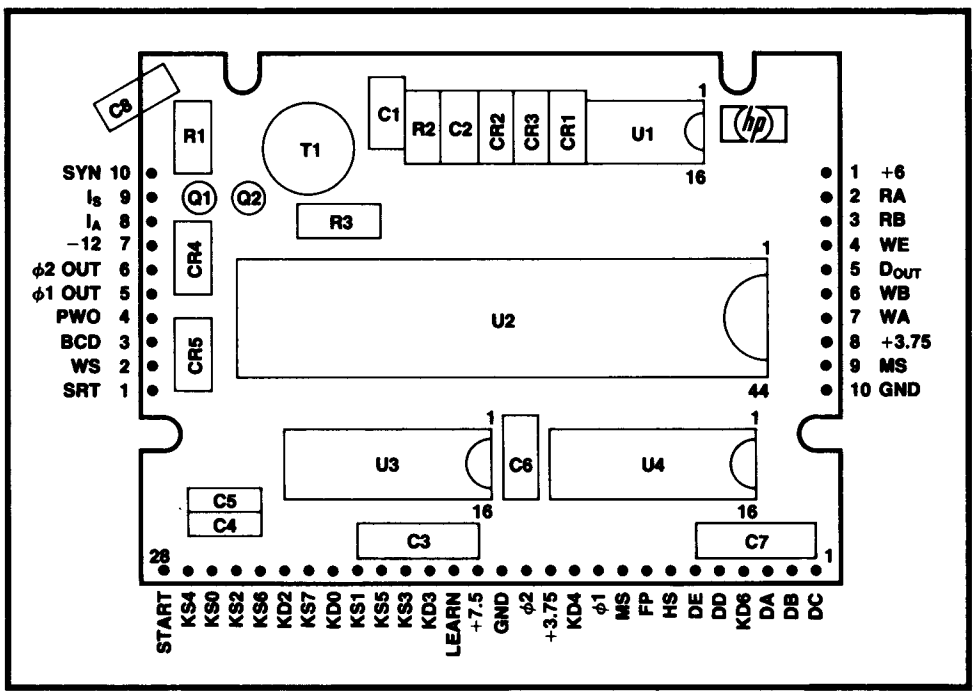


Fig. 1. *The HP-65 logic board is designed for automatic testing. Access to 45 test points is provided.*

The patterns are continually compared. If they ever fail to compare, their error condition is frozen in the pattern storage registers and a flag is sent to the computer. The computer can then request all pertinent information from the tester, which keeps a record of events leading up to a failure. For example, microinstructions are saved and decoded to show which chip was talking on the calculator buses when the failure occurred. Other information retained by the tester is the present ROM address, the previous ROM address, the identity of the active ROM (one of twelve), the bit count at the time of failure, and the number of word times since the start of the test sequence.

All this information and the contents of the two pattern storage registers are then sent to the computer. The computer sorts through the data with the aid of a diagnostic table and prints out the nature of the error and the component most likely to have caused it.

Tester Architecture

The HP-65 logic board tester has six main parts.

The *controller* is a 32-state ROM machine with 16 qualifiers and two-way branching on test conditions. There are eighteen instruction lines. Each state issues instructions that control the condition of other hardware in the tester. The controller also contains the master clock, a crystal-controlled oscillator set to the specified limit of the MOS circuits, 200 kHz.

The *interface circuits* consist of a voltage comparator and buffer amplifier between each calculator's MOS circuits and the input to the pattern storage register, which is a TTL circuit. The voltage comparator checks the upper and lower limits of the logic levels.

The *pattern storage and comparator circuit* consists of two 16-bit registers of D-type flip-flops with a 16-bit parallel comparison circuit between them to check for parity. If the patterns do not compare after each clock period (5μs) an error flip-flop is set. The clock is then inhibited to each of the 16-bit pattern storage registers and the error condition is saved.

Input/output information is received by the tester and transmitted to the computer over 16 parallel lines. Inputs are stored in a 16-bit buffer register which receives commands and data from the computer. Three of the input lines from the computer are assigned as output select control lines to specify which data to send back to the computer. The other 13 input lines are used to control the logic board: six lines specify the key code (function), four lines simulate switches to the card reader, one line varies the power-supply limits, and two lines are used for status information.

A 16-bit, eight-channel multiplexer controlled by the three output-select lines is used to return data to the computer. A full handshake between the tester and the computer is done; this assures proper data-transmission timing.

Inputs to the logic board are either through the key lines or the card reader control lines. The key lines are made through an 8 × 5 matrix. A six-bit code

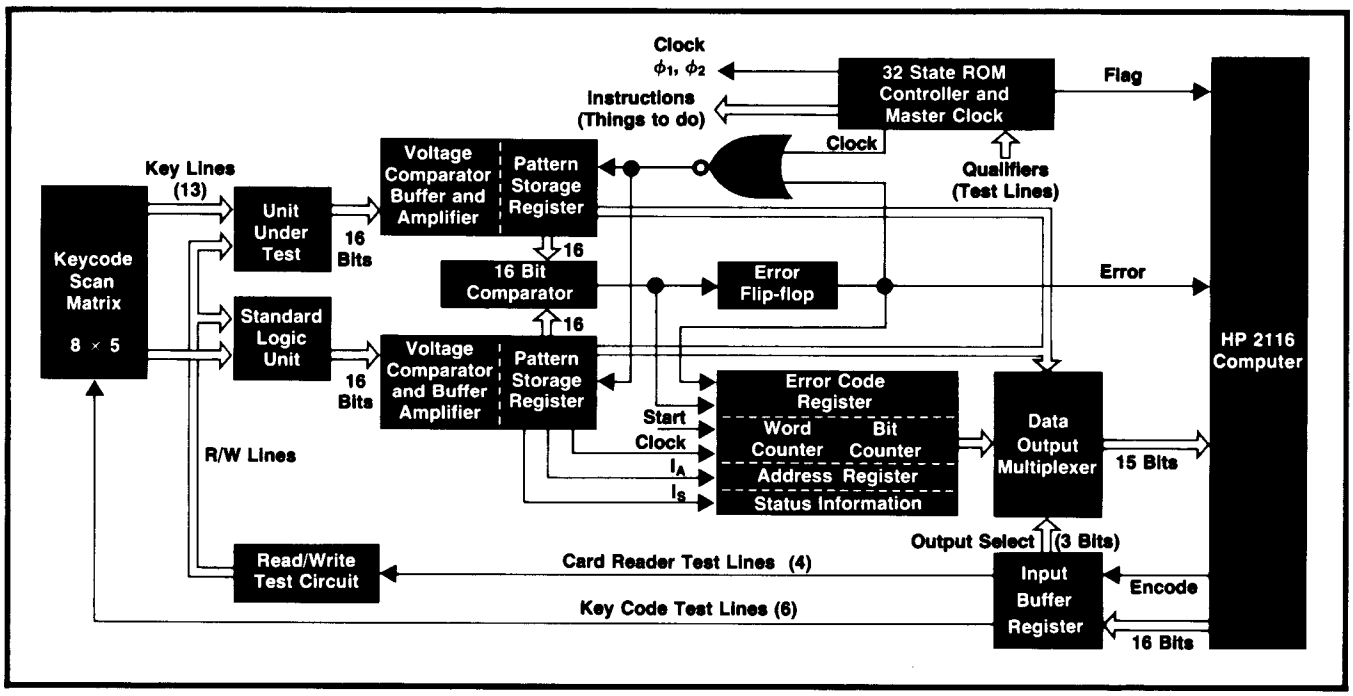


Fig. 2. *Logic board tester compares a reference board with a test board. Failures are analyzed by the computer, which prints out diagnostic messages.*

from the computer specifies the key code. There are four lines to the card reader: learn mode switch, motor switch, head switch, and file protect switch.

The *read/write test circuit* is a 15-state controller with instruction decoding for determining when to shift the read/write data pattern in and out. To simulate the magnetic card, reference write patterns are stored in two 600-bit shift registers. A third 600-bit shift register is used as a counter to keep track of how many shifts have been done.

Card Reader Testing

To make the card reader circuit read or write it is necessary to close the motor switch and the head switch. Then the condition of the learn mode switch line specifies whether the unit will read or write. The file protect line specifies the condition of the write enable line.

The test sequence consists of writing a test pattern from each of the two logic boards and comparing the pattern of the reference board to that of the board under test. The reference patterns are stored in two 600-bit shift registers. The next step is to read the same pattern back into the calculator and then write it back out again. The second write sequence is used to verify that the read circuit is functioning properly. The write enable line is tested by making the file protect signal a logical zero on the first write sequence and a logical one during the second write sequence.

Overall Test Sequence

The steps in the test sequence are as follows:

- Power off, push start button.
- Turn power on, set time delay (this allows the power supply to rise to the proper level).
- Test power supply and clock for proper high and low limits. Also test if power-on pulse was given properly. If not go to error routine.
- If everything is correct give a pseudo power-on (PWO) signal. This sets the starting address to zero.
- Test if both systems are in synchronism and if not, slow the clock to the unit under test until both systems are in synchronism.
- Release power-on pulse and start comparison test. When display has turned on and no error has occurred, flag computer for first key code sequence.
- After receiving first key code, enable key code matrix and continue to enable until display turns off.
- Wait until calculator display has turned back on again.
- Test if calculator is ready for its next test sequence, which can be through the key lines again or through the card reader lines.
- The test sequences are continued until the computer gives an end code. Then the "good" light turns on.
- If an error occurs the tester generates an error code corresponding to that fault. The computer is then flagged with the error line high.
- The computer then sends back a series of output select codes to specify which information is to be sent back to the computer. After the computer has received all information from the tester, an end code is sent to the tester. The tester turns on a "bad" light.
- The computer then prints out diagnostic messages.

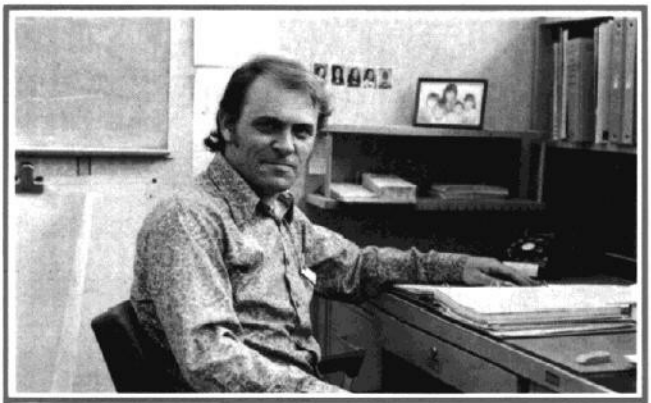
Test Time

The time required for testing a logic board is determined by the number of test sequences and the length of time of each function. Typical test time is about one minute for the complete test sequence, that is, for a good board.

Acknowledgments

I would like to thank Larry Gravelle and Harry Griffin of Advanced Products Division's electrical tooling department for all the support they gave, Marlin Schell for writing the software package, and Rich Whicker for some suggestions and ideas on testing the card reader chip.

Special thanks to Ron Bernard, Sue Gross and Carl Forsyth for their help putting the prototype together.



Ken Peterson

Ken Peterson, a member of HP Laboratories since 1965, has worked on the 2116 Computer, the 9100 Calculator, and all of HP's pocket calculators. Ken was a radar technician in the U.S. Air Force and later studied electronics for two years at a community college. His home is in Fremont, California, just south of his native Oakland, and he frequently attends Oakland Raiders' professional football games, especially enjoying it when the Raiders beat their rivals, the Kansas City Chiefs. He's a softball player and a skier, too. He and his wife Carmen have five daughters ranging in age from four months to fourteen years.