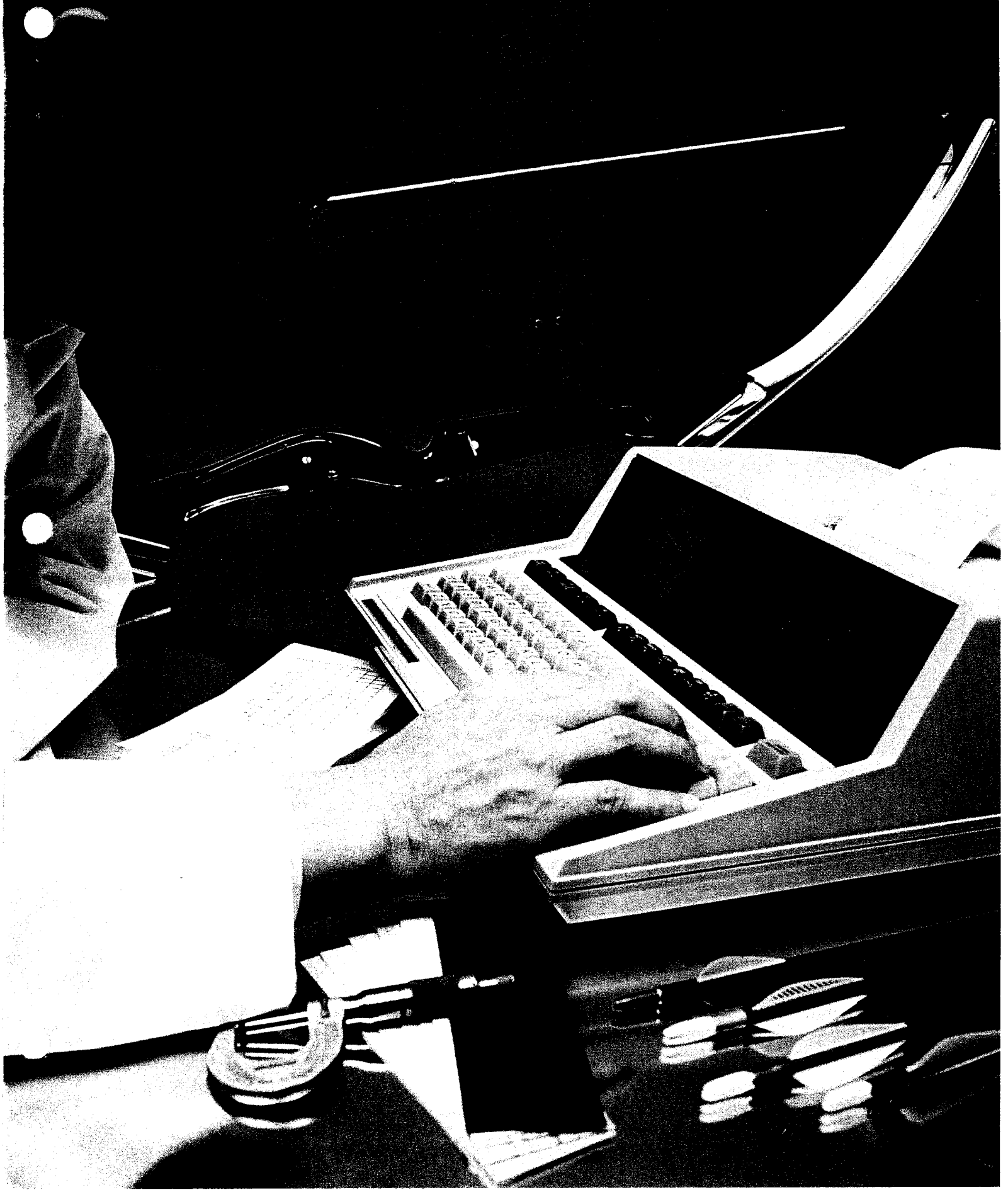
JULY 1980

HEWLETT-PACKARD JOURNAL



HEWLETT-PACKARD JOURNAL
Technical Information from the Laboratories of Hewlett-Packard Company

JULY 1980 Volume 31 • Number 7

Contents:

In this Issue:

The six articles in this issue deal with the design of Hewlett-Packard's first personal computer, the HP-85. What makes a computer "personal"? Mainly, the cost has to be low enough that the computer is likely to be bought either by an individual or by a company for the exclusive use of one individual. Compactness is also important, because the computer's owner may want to use it sometimes at the office, sometimes at home, and sometimes at a job site.

Unlike personal computers designed primarily for the home hobbyist, the HP-85 is designed for personal use in business and industry by professionals such as engineers, scientists, accountants, and investment analysts. Of course, a serious hobbyist might also buy it for home use, and schools might find it a good computer for students of the computer arts to practice on. Our cover photo depicts a bow designer using the HP-85 to plot force-draw curves and compute energy storage. Alert readers may remember that our October 1976 cover also dealt with bow design. That's not a coincidence; Art Director Arvid Danielson is one of the top amateur archers in the U.S.A. (The red bow in the cover photo is a compound bow supplied by Jennings Compound Bow, Inc., and we are grateful for their help.)

Certain of its features set the HP-85 apart from other personal computers. First, everything is built in—keyboard, CRT display (see page 19), printer (page 22), and magnetic tape drive (page 14). In other words, it's an *integrated* computer. There's no need for a bunch of cables to hook up a lot of optional capability; you get everything you need in one package. Second, the HP-85 is ready to go when you turn it on. The operating system and the BASIC language (see page 26) are permanently stored in the machine and don't have to be loaded from a magnetic tape or a flexible disc before you can get started. Third, the HP-85 displays not only letters and numbers, but also graphs and pictures. And anything you see on the CRT display can be copied on the printer just by pressing a single button. Finally, if you do want to expand the system, you can. Four slots in the back of the machine accept input/output modules that connect the computer to disc drives, plotters, printers, instruments, and the like (see page 7). The HP-85 is the lowest-priced instrument controller currently available, a distinction that will no doubt account for a major portion of its sales.

-R.P. Dolan

Editor, Richard P. Dolan • Associate Editor, Kenneth A. Shaw • Art Director, Photographer, Arvid A. Danielson
Illustrator, Nancy S. Vanderbloom • Administrative Services, Typography, Anne S. LoPresti • European Production Manager, Dick Leeksma

A New World of Personal/Professional Computation

Now, an inexpensive computer system with integral display, mass storage, hard copy, and graphics capability is available for personal use by the technical professional or first-time computer user.

by Todd R. Lynch

IN JANUARY 1980, Hewlett-Packard made its first entry into the personal computer marketplace with a product called the HP-85 (Fig. 1), a totally self-contained computer system for the technical professional or first system for the beginning small-computer user. It is based on a custom eight-bit microprocessor, with an operating system and BASIC* language commands stored in 32K bytes of ROM (read-only memory). There are 16K bytes of RAM (random-access memory) in the machine, with the capability of adding another 16K bytes. A cathode ray tube (CRT) display with high-resolution graphics, a thermal printer, a tape cartridge drive, a keyboard, and four plug-in input/output (I/O) slots come as standard features.

The HP-85 represents a significant contribution to the personal computer field. Its characteristics reflect the objectives that guided its design and development:

- An integrated rather than component system. The central processor, keyboard, display, printer, and mass storage are all contained in a single package. This eliminates the maze of cables and connectors that would be required to interconnect a system made up of individual components.
- Portability. The machine is small and lightweight so that it may be easily relocated.
- Suitable for home use. It was envisioned that the HP-85 would accompany its owner to work during the day and to the owner's home at night. To provide the customer with a computer that can be used at home, the design had to include barriers to the annoying electromagnetic interference (EMI) generated by the electronics in the machine. Therefore, the plastic top and bottom cases are metallized with a thin coat of aluminum. This conductive shield is tied to the sheet metal back panel, which is grounded. It is expected that the HP-85 will meet applicable U.S. FCC (Federal Communications Commission)

*Beginner's All-purpose Symbolic Instruction Code

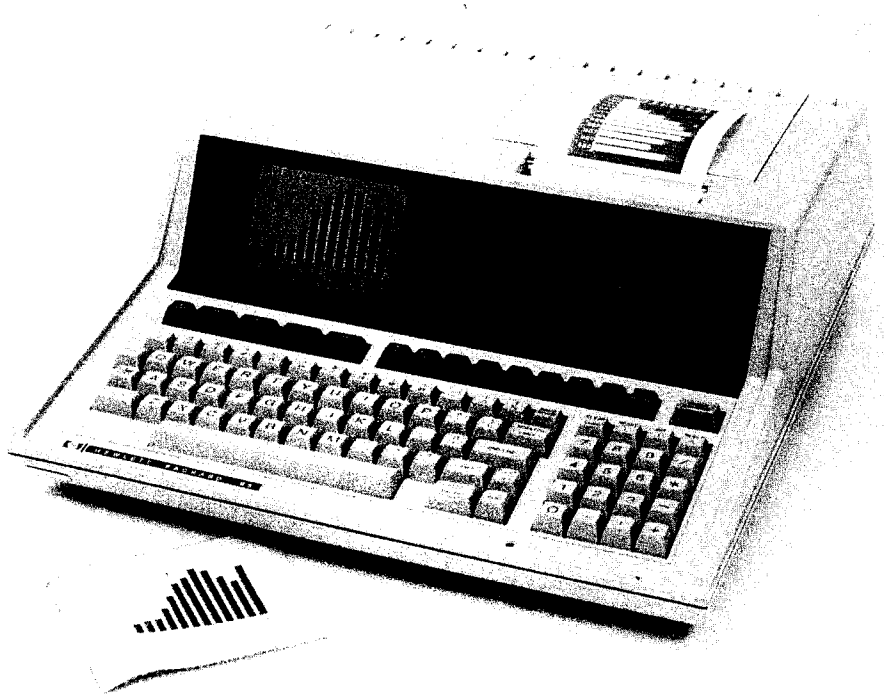


Fig. 1. *The HP-85 Personal Computer uses BASIC language programming and operating commands that are permanently resident in the system firmware. Contained in a single package are all of the elements for a small computer system—keyboard entry, tape mass storage, CRT display, and a thermal printer. The inverse video display mode is shown with the associated printer copy at the upper right. In the left foreground is the printer copy for the normal white-on-black display mode.*

and German VDE (Verband Deutsches Elektrotechniker) EMI requirements. However, the certification process has not yet been completed.

- Ac line operation. The intended customer for the HP-85 is assumed to be the scientist or engineer, and the normal location for operating the machine is assumed to be the laboratory or office, where the presence of an ac line outlet is implied. The machine operates on either 115 or 230 Vac at a frequency from 50 to 60 Hz.
- Quiet operation. Since use in an office environment requires minimal noise generation, a cooling fan is eliminated by designing a highly efficient switching power supply and by writing firmware that allows only one of the input/output devices to operate at a time. By turning off the CRT when printing or using the tape drive, much power is saved. Total system power is nominally 25 watts. Using only convection cooling, dissipation of this power level generates only a 5 to 10°C temperature rise inside the package.
- BASIC language programming. The language is capable of complex computations, but is still friendly and easy to use. Forty-two built-in functions (e.g., SIN, MIN, ABS, SQR, etc.) are incorporated into the machine to facilitate the user's programming task. Creating a computer that is friendly and easy to use is accomplished, in part, by building the BASIC language into ROM. As soon as the system is turned on, it is ready to go. The operating system is always resident in the machine and cannot be altered or destroyed by an errant user program.

  Error messages are given to the user in the form of an error number plus a short description of the type of error. Warnings are also given for certain arithmetic conditions (e.g., division by zero). A warning differs from an error message in that a warning allows execution to continue. In the case of a divide-by-zero warning, the computer assumes a default result of ±9.99999999999E499. The user can override this assumption and cause the machine to halt on mathematical errors by inserting the DEFAULT OFF statement in the program.
- Expandable system capability. Extra memory and command sets can be easily added. The ability to connect interfaces for control of instruments and other input/output peripherals is provided by four I/O slots in the rear of the machine.
- Low cost. The system is affordable by an individual or by a company buying for individual workers. The cost of the electronics was minimized by the design of nine custom LSI (large-scale integrated) circuits. These nine packages require a fraction of the space and power that their off-the-shelf equivalents would require. Printed circuit boards in the product are all two-layer boards to save cost. The package top and bottom cases are injection-molded plastic.
- Reliability and serviceability. The system electronic design emphasizes the use of integrated circuits and modular construction. The integrated circuits in the system were carefully characterized over their entire operating region. They are packaged in ceramic to achieve maximum reliability. All of the NMOS* parts are burned-in for 24 hours at 125°C to weed out infant mortality failures.

  The HP-85 is designed with subassemblies that are easy to manufacture and service. The CRT, tape drive, logic board, printer/power supply, keyboard, and back panel make up the major subassemblies. These are indicated in Fig. 2. After a subassembly is manufactured, it is independently tested. When the HP-85 goes through final assembly, it is exhaustively tested by a diagnostic program resident in a special plug-in ROM. Good systems are burned-in for 48 hours to screen out infant mortality failures and marginally operating units. This period is also used to age the CRT phosphor coating, which prevents dull spots in the display.

*N-channel metal-oxide semiconductor integrated-circuit process.

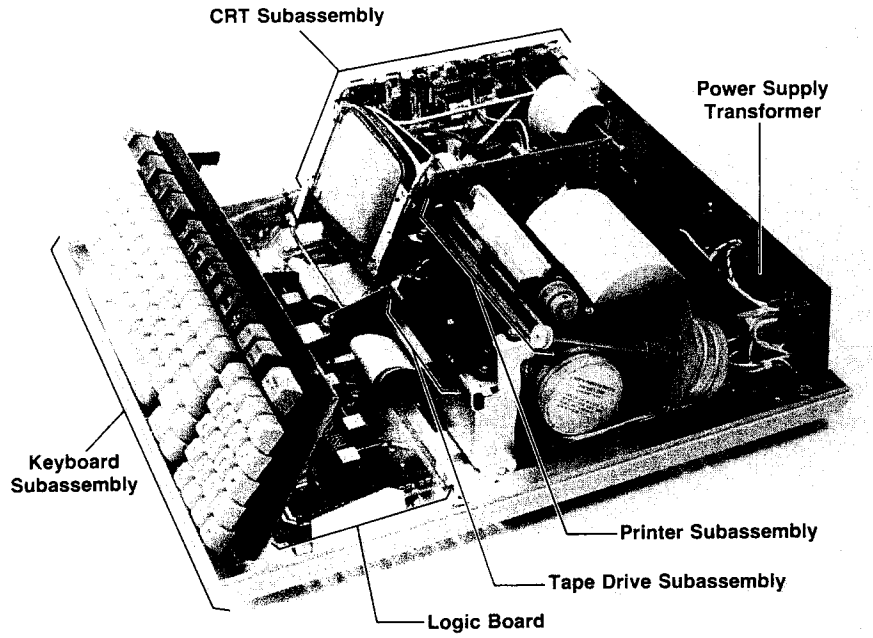


Fig. 2. *Internal view of the HP-85 with the major subassemblies identified. The compact designs of these subassemblies make it possible to provide a complete computer system in a single portable package.*

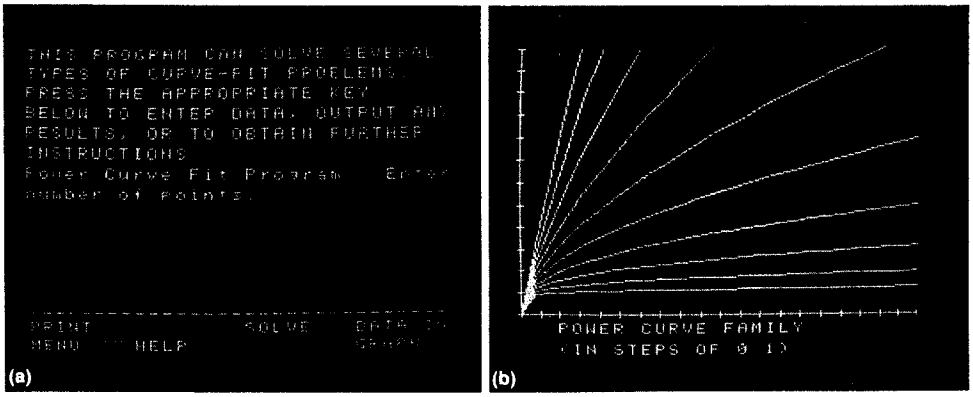


Fig. 3. *(a) Alphanumeric display mode. A full set of 128 ASCII characters can be used to provide up to 16 lines of 32 characters each for alphanumeric data. Labels to identify selected special key functions can be displayed at the bottom of the display as shown. (b) Graphics display mode. An array 256 dots wide by 192 dots high is available for plotting and labeling data on the 127-mm CRT.*

- An internal self-test capability. Should the user suspect a problem with the machine, the user can run a self-contained diagnostic program by pressing the **TEST** key. This program interrogates all of the machine electronics. If everything checks out, the ASCII* character set will be displayed and printed. If a problem is encountered, the message ERROR 23; SELF TEST will appear on the CRT screen. In case of trouble, the customer is advised to seek service from an authorized HP service center or dealer.

Display

The display is a 127-mm (5-in) diagonal black-and-white electromagnetic-deflection CRT. The design of this subassembly is discussed in the article on page 19. The memory that holds the display data contains over 65,000 bits and is completely independent of the user's programming memory. It is partitioned into alpha and graphics buffers. In alpha mode the display shows 16 lines of 32 characters per line (Fig. 3a). Scrolling keys are provided to view an additional 48 lines of alphanumeric data. The graphics display mode is entered by pressing the **GRAPH** key. Now the display provides a resolution of 256 dots wide by 192 dots high. Each dot can be individually accessed by the user with the BPLOT command. In the graphics mode, the SCALE, AXIS, MOVE, DRAW and PLOT commands make it easy to create plots (Fig. 3b) and line drawings. Labels and other alphanumeric information can be added to the graphs, either horizontally or vertically. In addition, HP-85 users can adjust the brightness of the display to adapt to changing lighting conditions.

Printer

A 32-character-per-line thermal printer (see article, page 22) is provided for hard copy output. A set of 128 ASCII characters is available for printing when outputting alphanumeric copy (e.g., program listings). The printer prints alphanumeric copy bidirectionally at a nominal rate of two lines per second. Short lines are printed faster than long lines.

A **COPY** key lets the user obtain a copy of the CRT display on thermal paper. If the CRT is in the graphics mode, the printer prints in one direction only and the print output is rotated 90° to allow for strip-chart plotting.

To compensate for variations in paper, printheads, and the type of printing being done, the HP-85 gives the user the ability to lighten or darken the intensity of the output. A print intensity switch located under the paper door provides for selection of eight levels of print intensity.

*American Standard Code for Information Interchange

Tape Cartridge

Mass storage for the HP-85 is provided by a tape cartridge system (see article on page 14) using the 98200A data cartridge. Each cartridge holds about 200K bytes of information. This information may be any combination of source program files, binary program files and data files. The directory stored on the tape will hold 42 six-character file names. By using the CATalog command, the user can easily review the contents of any data cartridge. A portion (256 bytes) of the directory is automatically stored in the system RAM. This significantly reduces the time required to access a file.

The HP-85 tape electronics can drive the tape at two different speeds. When searching for a file the tape speed is 1524 mm/s (60 in/s). Once the file is found, the speed for read/write is slowed to 254 mm/s (10 in/s). This corresponds to a data transfer rate of 650 bytes per second.

An *autostart* capability is provided by the system firmware. When power is applied to the HP-85, it automatically rewinds the tape and reads the tape directory, looking for a program named Autost. If the file exists, the computer will find it on the tape, load it into memory, and immediately start running the program. Autost programs are supplied with some HP software pacs or can be generated by users for their particular needs.

Security is available for the files stored on the tape cartridge. When storing a file the user can specify one of four types of security for that file. Also, the entire cartridge can be manually protected against inadvertent overwriting by setting the write-protect switch that exists on each cartridge.

Keyboard

The keyboard is partitioned into four sections as shown in Fig. 4. A typewriter keyboard is available for entering alphanumeric data. A numeric pad is provided for fast numeric entries or calculator applications. Directly under the CRT display are four special function keys. Combined with the **SHIFT** key, these provide eight keys that can interrupt a running program. The **KEY LABEL** key causes a description (up to eight characters specified by the programmer) of each programmed key to appear on the bottom two lines of the display (Fig. 3a). This serves to remind the user what each key is programmed to do. The top row of keys on the right provides a convenient means to scroll or move the cursor on the display, to operate the printer or tape

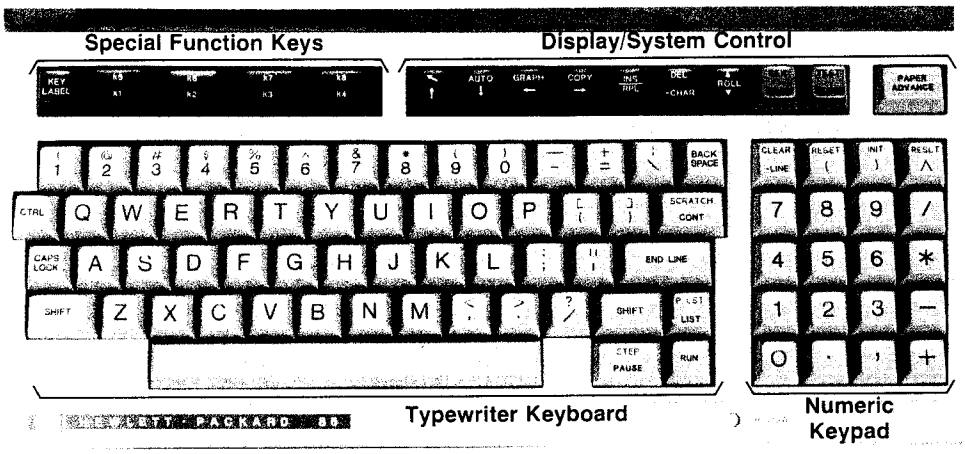


Fig. 4. *The HP-85 keyboard is organized into four sections. Besides the normal typewriter section there is a numeric keypad for easy entry of data, a group of special function keys located close to the CRT display where their respective assigned labels may be displayed, and a set of control keys for frequently used system commands and editing/display control.*

unit, or to edit previous input with the **INS/RPL** and **DEL-CHAR** keys.

Plug-Ins

Four I/O slots are provided in the rear of the unit. An additional 16K bytes of RAM can be added to the HP-85 by plugging a RAM module into any one of the slots. The programming power of the machine can be enhanced by plugging in a ROM drawer. This drawer can contain from one to six 8K-byte ROM modules. The HP-85 will soon have ROMs available for matrix operations, mass memory interface, printer and plotter interface and general I/O capability.

Four interface plug-ins have been designed to provide interfacing to four common bus structures. The 82937A HP-IB Interface is currently available for connection and control of devices designed for the IEEE Standard 488-1978 instrumentation bus. An RS-232C serial interface and BCD and GPIO interfaces will also become available soon. For a more extensive discussion of the I/O design and capabilities of the HP-85, refer to the article on page 7.

Enhanced BASIC

The HP-85 provides a very powerful BASIC language. The development and capabilities of this firmware are reviewed in the article on page 26. One feature of the language is its mathematical accuracy. Numbers are carried internally in decimal form with 15-digit precision and are rounded to 12 digits for output to the user. They can range in size from 10^{-499} to $9.99999999999\times10^{499}$.

The string handling ability of the machine is extensive. A string may be any length, limited only by the amount of available read/write memory.

The CHAIN and COMmon commands allow long programs to be segmented and stored separately on the data cartridge. Different segments are brought into memory when needed and executed by the CHAIN statement. COMmon allows current variable values to be passed from one segment to another.

A programmable tone speaker is provided. The user can specify both the frequency and duration of the tone by using the BEEP command. This provides the capability for custom audio error indication, input prompting, or playing Bach.

When debugging a program, the user can take advantage of a number of tools provided by the machine firmware. The user can either single-step through the program operations or run the program. Using the TRACE, TRACE ALL and TRACE VARiable commands, changing variables and changes in the sequence of execution can be viewed. This makes it much easier to find where and why a program does not execute as expected.

In all, there are 42 predefined BASIC functions, 65 BASIC statements and 20 commands available in the standard HP-85 system ROMs. Of the standard 16K bytes of user RAM in the computer, only 10% is reserved for system overhead. The rest is available to the user.

Software

To provide the user with solutions for particular problems, additional software packages are available for use by the HP-85. Each machine is sold with a Standard Pac consisting of a preprogrammed tape cartridge and user's manual. On this tape there are programs for general mathematics and business. There is also a game and a music composing program. This pac is intended to acquaint the customer with the power of the machine as well as offer solutions to some common problems.

There are other software pacs currently available. A BASIC training pac is intended as a self-teaching course for novice users. Statistics, finance, waveform analysis, mathematics, circuit analysis, games, linear programming and text editing packages are also available.

Todd R. Lynch

Todd Lynch is a native of Rochester, New York. He joined HP Laboratories in 1972 after working for a year on process control logic designs. Since coming to HP he has worked on computer architectures, and since transferring to what is now HP's Corvallis Division, Todd designed the architecture for the HP-85 CPU. He was the project manager for the integrated circuits in the HP-85 and is now project manager for high-end mainframes. He received a BS degree from Grove City College, Pennsylvania in 1970 and an MSEE from the University of Wisconsin in 1971. Todd is co-chairman for the Willamette Valley Junior Tennis Tournament and in addition to tennis enjoys woodworking, softball, and skiing. He lives in Albany, Oregon with his wife and new son.

SPECIFICATIONS
HP-85 Personal Computer

READ/WRITE MEMORY
STANDARD: 16,384 bytes (10% for system overhead)
OPTION: 32,764 bytes, using 16,384 byte plug-in RAM
CRT DISPLAY
SCREEN SIZE: 127 mm (5 in) diagonal
INTENSITY: Adjustable
CHARACTER GENERATION: 5 × 7 dot character font in 8 × 12 dot cell
SCREEN CAPACITY: 16 lines × 32 characters full display, 64 line scrolling
CURSOR: Underline
PLOTTING AREA: 256 wide by 192 high dot matrix.
THERMAL MOVING-HEAD PRINTER
INTENSITY: Adjustable
PRINT SPEED: Two 32-character lines per second
CHARACTERS: 5 × 7 font in 7 × 10 cell.
GRAPHICS OUTPUT: Dot for dot reproduction of CRT display.
PAPER SIZE: 122m (400 ft) long by 10.8 cm (4.3 in) wide roll.
TAPE CARTRIDGE DRIVE
PROGRAM CAPACITY: 195K bytes
DATA CAPACITY: 210K bytes
ACCESS: Directory, file-by-name, 42 named files, max
ERROR RATE: 1 bit in 10^8.
SEARCH SPEED: 152 cm/s (60 in/s).
READ/WRITE SPEED: 25.4 cm/s (10 in/s).
REWIND TIME: 29 s, max.
CARTRIDGE SIZE: 98200A, 63.5 × 82.5 × 12.7 mm (2.5 × 3.25 × 0.5 in)
QUARTZ CRYSTAL TIMER
TIME KEEPING: Seconds since midnight with year and day-in-year
TIME SET: Must be reset each time power is turned on.
PROGRAMMABLE TIMERS
NUMBER: Three
OPERATION: Independently programmable interrupt
RANGE: 1 ms to 99,999,999 ms (27.7 hours)
BEEPER
TONE: Programmable from approximately 0 to 4575 Hz.
DURATION: Programmable
DEFAULT TONE AND DURATION: 2000 Hz, 100 ms.
I/O SLOTS
NUMBER: Four
INTERFACE OPTIONS: HP-IB (82937A), RS-232, GP-IO, BCD, 16K byte RAM module (82903A), ROM drawer (82936A) for up to six ROM modules.
BASIC LANGUAGE
PREDEFINED FUNCTIONS: 42, of which 10 are trigonometric.
STATEMENTS: 81, 16 of which are for graphics.
COMMANDS: 20, 10 of which are programmable.
DYNAMIC RANGE
REAL: $-9.99999999999E499$ to $-1E-499$, 0, $1E-499$ to $9.99999999999E499$.
SHORT: $-9.9999E99$ to $-1E-99$, 0, $1E-99$ to $9.9999E99$.
INTEGER: −99999 to 99999
TEMPERATURE ENVIRONMENT
OPERATING: 5° to 40°C (41° to 104°F).
STORAGE: −40° to 65°C (−40 to 149°F).
POWER REQUIREMENTS
SIZE: HWD 15.8 cm × 41.9 cm × 45.2 cm (6.3 in × 16.5 in × 17.8 in)
WEIGHT: 9.06 kg (20 lb)
VOLTAGE: 90-127 Vac (115 Vac line) or 200-254 Vac (230 Vac line) switch selected on rear panel
FREQUENCY: 50-60 Hz
CONSUMPTION: 25W.
PRICE IN U.S.A.: HP-85 base price: $3,250.
MANUFACTURING DIVISION: CORVALLIS DIVISION
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330 U.S.A.

Acknowledgments

A great deal of credit must be given to the management team for the HP-85. Ernst Erni was the section manager for the project. His foresight and guidance contributed enormously to the success of the product. Kent Stockwell was the project manager responsible for the mechanical designs and overall system integrity. Chung Tung's support as lab manager was very beneficial. Tim Williams and John Nairn served as project managers responsible for the plug-in modules.

Also, credit goes to Marv Higgins, who engineered the subassembly interconnect scheme as well as the EMI shielding design, and to Chuck Dodge, who did the excellent job of industrial design on the product. Norm Glaeser was responsible for product QA. Deme Clainos and Carmen West contributed their skills in the product marketing area.

Adding I/O Capability to the HP-85

With the implementation of I/O features, the capabilities of a self-contained personal computer system are expandable to control instruments, add on more powerful peripherals, and even talk to other computers.

by John H. Nairn, Tim I. Mikkelsen, and David J. Sweetser

THE HP-85 is an integrated, stand-alone personal computer. This means that within this single package are contained all of the elements of a small computer system. Part of Fig. 1 shows a block diagram of the HP-85 mainframe. The box labeled CPU contains the main processor, ROM and RAM memory, operating system, power supply, and all of the other elements that in a large mainframe or minicomputer would constitute the computer itself. The remaining elements provide the human interface, letting a programmer and/or operator communicate with the computer. Besides the obvious advantage of having a complete, functional computer in a single portable unit, the integrated computer architecture provides certain operational advantages.

For example, in the HP-85, a graphics display created on the CRT can be dumped to the built-in printer by executing a single COPY statement. This feature would be much more difficult if not impossible to provide if the printer were an external device with unknown operational characteristics.

The internal peripherals chosen to be incorporated into the HP-85 make it a complete tool for solving a large variety of computational problems. However, there are two classes of tasks that require that the system be expandable.

The first class consists of tasks for which the human interface elements need to have more powerful characteristics than those provided by the internal peripherals. A particular application may require a printer capable of full page width printing, or of filling out standard forms in multiple copies. For graphics displays, a full-size plotter with multicolor capability might be required. Or the use of a human interface element not even provided in the basic system, such as a digitizer, may be necessary. The ability to support external as well as internal peripherals greatly expands the class of problems the system can solve.

The second class consists of tasks in which it is desirable to eliminate as much of the human interface as possible. In data acquisition applications, measurement instruments are available that can communicate their results directly to the computer, eliminating the need for manual entry of data by an operator. When the computer is capable not only of acquiring and/or performing analyses on the data, but also of making decisions based on that data and providing feed-

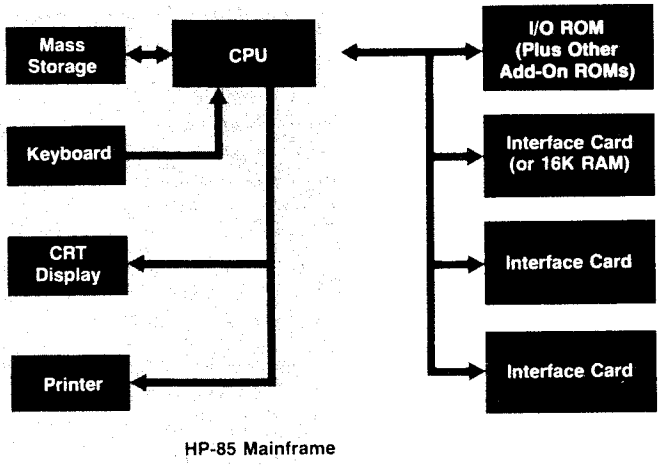


Fig. 1. *The basic functional blocks for the HP-85 personal computer are integrated into a single mainframe for a self-contained computer system. Four I/O slots are provided to expand system capability. Various combinations (such as the one shown) of the add-on ROM drawer, additional 16K RAM, and peripheral interfaces can be installed in these four slots.*

back that modifies the operation of the system to which it is connected, the computer functions as a controller.

Two elements are required to add I/O capability to the HP-85. First, a piece of firmware (I/O ROM) is required to give the operator access to the interfaced device. Second, a piece of hardware (interface card) is required to provide electrical, mechanical, and timing compatibility for the device to be interfaced to the computer.

The HP-85 is a BASIC language computer. Many of the language features are similar or identical in syntax and semantics to the interfacing statements on the 9835 and 9845 Computers.[1,2] The design of the HP-85 interfacing capability also draws heavily from the 9825 Computer's I/O architecture.[3]

The HP-85 contains both read-only memory (ROM) and read-write or random-access memory (RAM). The RAM contains the user's BASIC language program (software) and data. The ROM contains the machine language program (firmware) which recognizes and executes the statements provided by the BASIC language. Thus, the operating system ROM in the HP-85 provides such statements as PRINT, DISP, and INPUT for accessing the internal peripherals. When external peripherals are added, their wider range of capabilities requires more extensive BASIC language statements to fully use these capabilities. Additional plug-in modules, called add-on ROMs, merely enrich the BASIC language by increasing the number of statements and functions that can be recognized and executed.

Some of these add-on ROMs may be dedicated to a particular device. For example, a ROM may add statements to the BASIC language which are designed only to provide complete control of a floppy disc or of a full-size plotter. The I/O ROM, on the other hand, is designed to be as much as possible a general-purpose tool for communicating with the wide variety of peripherals and instruments available.

Almost all computers provide language extensions for outputting data to a device or entering data from a device. The OUTPUT and ENTER statements are usually sufficient for communicating with external peripherals. In controller applications, however, where timing may be critical and the speed of the external devices and instruments may not be well matched to the speed of the computer, other methods of transferring data and control information can increase the performance of the system. In many cases this can make the difference in whether the application can be done at all. For example, in taking data from very slow devices, the ability to transfer data under interrupt allows the HP-85 to perform other operations instead of waiting on the slow device, thus increasing system throughput. In the other extreme, being able to capture data in a burst transfer from a very fast device and then process the data into internal format at a later time can be a "make-or-break" capability in fast data acquisition applications.

In the back of the HP-85 are four slots that allow add-on ROMs, add-on RAM, and interface cards to be connected to the internal memory bus. A single ROM drawer containing the I/O ROM and up to five other add-on ROMs can be plugged into any one of these slots. Thus a typical interfacing configuration can include the ROM drawer and three interface cards; or the ROM drawer, the add-on RAM module and two interface cards, as shown in Fig. 1.

On each interface card is a custom integrated circuit called the translator chip (TC), which translates between the timing and protocol requirements of the internal memory bus and a microcomputer bus (Fig. 2). Each card also contains an 8049 microcomputer and a set of discrete drivers that implement the electrical and protocol requirements

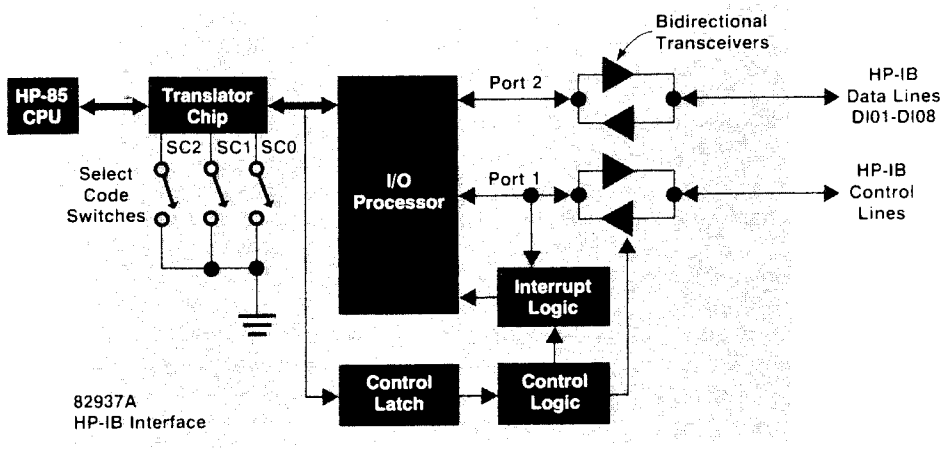


Fig. 2. *Typical interface card design. The translator chip permits communication between the system CPU and the microprocessor on board the interface. The I/O processor then implements the electrical requirements for the particular interface functions.*

for that card.

The most general-purpose interface card is the sixteen-bit parallel card (GPIO). Sixteen TTL compatible bidirectional lines and sixteen open-collector output lines, plus a variety of status, control, and handshake lines make this card a versatile workhorse. A variation of this card, with the input lines organized into four-bit nibbles, is also available for interfacing devices that operate in binary-coded-decimal (BCD) format.

A third type of interface card is designed to connect the HP-85 to serial I/O devices using either RS-232C or 20-mA current loop configurations. This card will also allow terminals to be connected to the HP-85, or allow the HP-85 to be connected to a modem or a host computer.

The fourth card is the 82937A HP-IB Interface, the HP-85's implementation of IEEE Standard 488-1978. Because of the microprocessor included on this card, a very powerful implementation of both controller and noncontroller capabilities is provided.

The HP-IB Interface Card is available now and the others (GPIO, BCD, and Serial I/O) will be available later.

I/O ROM

There are three areas that any interfacing capability in a computing system must connect—the programmer, the operating system, and the I/O hardware. On HP's interpretive computers the user interface is through keyword execution of the language extensions provided by the I/O ROM. An important aspect of the friendliness of these computers is the choice between calculator and program mode execution of the keywords. The program mode means the statement has a line number and is to be stored for later program execution (parsed,* stored in the program memory, executed at **RUN** time). The calculator mode means the statement does not have a line number and is to be immediately executed (parsed, stored in temporary memory, and executed).

In the HP-85, the I/O hardware—the interface card—is a type of channel processor. A channel processor is a computer with limited resources that performs many of the central computer's interfacing tasks. The interface to the I/O hardware is the I/O drivers. These implement the protocol that the central computer uses to tell the I/O card what it is supposed to be doing. The I/O drivers include data passage routines, command passage routines, status and control routines, and an interrupt handler.

The interface to the operating system is what makes interfacing capabilities a natural extension of the computing system. For example, I/O events can cause changes in the BASIC program flow. The interface to the operating system is achieved primarily in three ways—routine linkages, read/write memory use, and register use. An example of a routine linkage is the print driver—when the programmer specifies PRINTER IS 4, all PRINT commands cause a routine in read/write memory to be called. When the I/O ROM is in the system it replaces this read/write memory routine with a routine of its own to handle the PRINT. If the I/O ROM is not in the system the default system routine generates an error.

Read/write memory is used when the I/O ROM passes information to or from a system-defined location. For example, when an error occurs in an I/O ROM statement, the I/O ROM puts the error number in a common location that the system knows.

The register interface to the operating system consists of some CPU registers that the system reserves for system status and control. For example, register sixteen indicates the machine state—idle, running, and so forth.

There are several basic groups of interfacing and related capabilities provided by the HP-85 I/O ROM to the user.

- Formatted transfers—input and output of data with formatting and conversions
- Buffered transfers—fast handshake and interrupt data transfers through explicit buffering
- Register access—ability to set and interrogate the card configuration
- Interrupt access—ability to receive hardware interrupts
- Timeouts—ability to detect an inoperative card or peripheral
- Miscellaneous—keyboard masking, base conversion and binary operations
- HP-IB control—high-level access to HP-IB capabilities.

The ability to input and output data (*formatted transfers*) is probably the most commonly used of all interfacing capabilities. In some systems input and output are the only interface functions provided. The HP-85 allows for free-field or formatted input (via ENTER) and output (via OUTPUT). These statements are the tools that a programmer uses to get arbitrary string and numeric data into and out of the computer. Many times the peripheral has a non-ASCII character set, so a character conversion capability is provided to allow the programmer to set up a conversion table and deal with data as if it were normal ASCII data.

OUTPUT and ENTER are medium-performance data transfer mechanisms that consume the machine (program execution stops until the current statement is completed). *Buffered transfers* provide two additional methods to get data into and out of the HP-85—interrupt and fast handshake. The interrupt mode of data transfer is lower in performance than ENTER and OUTPUT, but does not tie up the computer, because it can be doing many other things in addition to the interrupt transfer. For example, while transferring data in the interrupt mode, the HP-85 can plot to the CRT, do arithmetic computation, print to the internal printer, or transfer data through other interrupt transfers. The fast handshake mode is much better in performance than ENTER and OUTPUT, but still ties up the computer. Neither of these buffered transfer modes performs any formatting. The data is placed into or taken out of an I/O buffer—hence the name "buffered" transfers. The buffer is a string variable that has been modified for use by the I/O ROM (via IOBUFFER). The buffered transfers are performed via the TRANSFER statement.

The programmer needs to access specific attributes of each interface card (*register access*). These attributes vary from card to card. For example, an RS-232 user would like to get at the clear-to-send bit and change its state; a sixteen-bit parallel interface user would like to change the logic sense from positive-true to negative-true logic. These features would become cumbersome if all of them were provided at the BASIC language level. Hundreds of keywords

*Parsing is the operation of taking a statement line and decomposing it into its elementary parts.

would be required. Instead, there can be up to sixteen card status registers and twenty-four card control registers on any interface card (accessed via STATUS and CONTROL). These registers allow the programmer access to the various low-level capabilities of the interface: parity, end-of-transmission character sequences, interface control and data lines, error indicators, and so on.

A programmer often needs to perform some operation when an interface condition occurs (*interrupt access*). Rather than constantly checking for these conditions, the interface card can check for the programmer (via ENABLE INTR), and when the condition happens, the end-of-line branch is taken to a BASIC service routine. The location of the service routine is set up with the ON INTR statement.

The *timeout* capability lets a programmer set an arbitrary length of time (from 1 to 32767 milliseconds) to wait for a response from the I/O hardware. Once a timeout has occurred, there is an end-of-line branch taken to a BASIC service routine specified by the programmer.

The *miscellaneous* statements and functions do not perform any interfacing capabilities. They are included to make the job of the interface programmer easier. There is the capability to mask out sections of the keyboard so that the operator does not inadvertently disrupt program flow. Base conversions for decimal numbers to and from binary, octal, and hexadecimal allow the programmer to display interfacing information in a convenient form. Boolean functions (and, or, exclusive or, complement, bit test) let the programmer test and modify interfacing information easily.

HP-IB is a mnemonic for Hewlett-Packard's implementation of IEEE Standard 488-1978. Since HP-IB is such a pervasive interfacing standard for instrumentation, printers, plotters and peripherals of all descriptions, there are several high-level HP-IB control statements in the HP-85 I/O ROM. There are many commands that send multiline messages on the HP-IB (such as TRIGGER, PASS CONTROL,

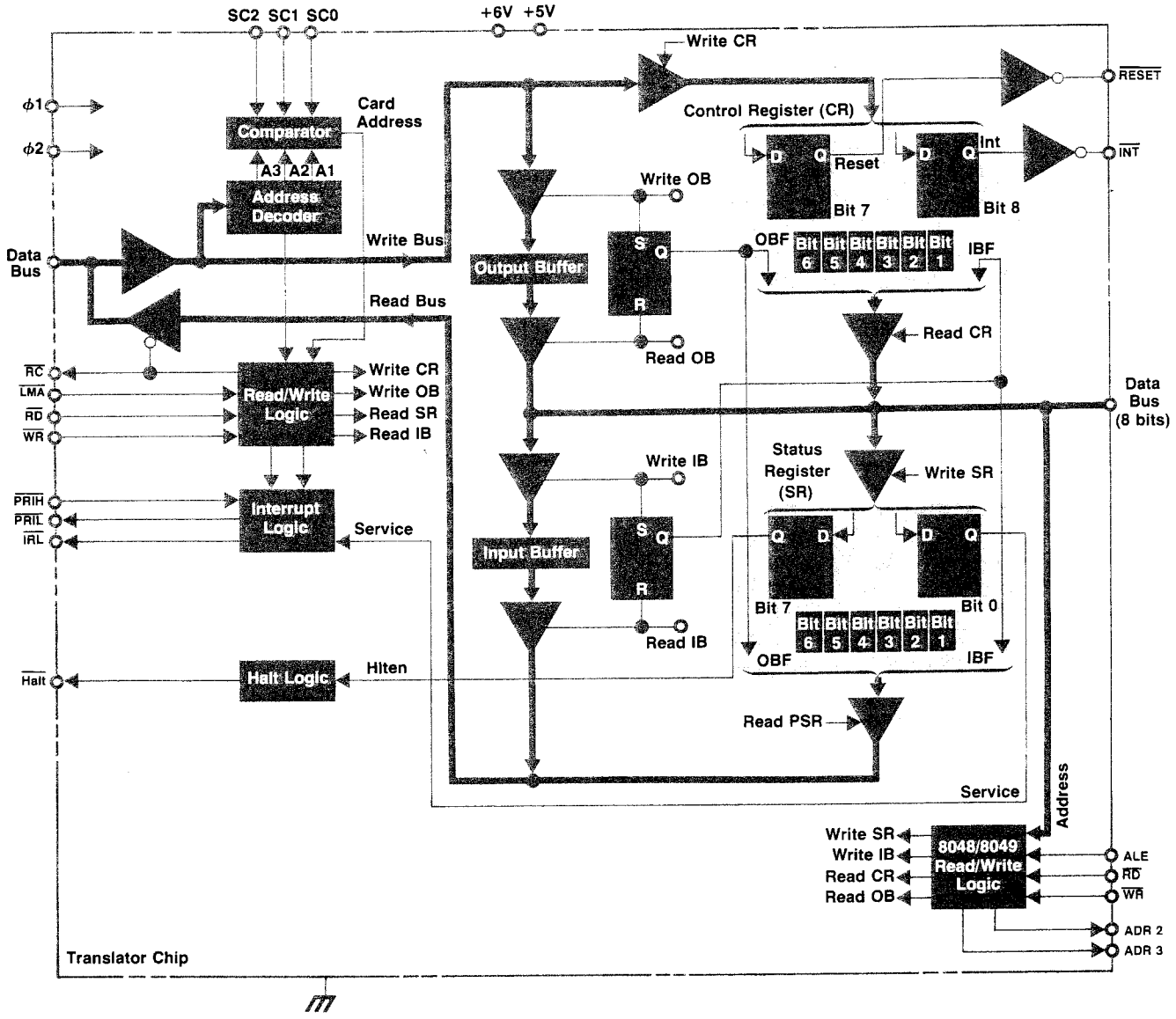


Fig. 3. *Block diagram of the translator chip that translates timing and protocol requirements between the HP-85 CPU and the interface card I/O processor.*

CLEAR, etc.). Multiline messages are those commands that the controller handshakes to the various devices on the bus in bit-parallel, byte-serial fashion. There are also commands that send uni-line messages on the HP-IB (such as ABORTIO, LOCAL, REQUEST, etc.), and there are functions that return status information (PPOLL and SPOLL). The other cards (Serial I/O, GPIO, BCD) use some of these statements for their own functions when appropriate.

Design of the 82937A HP-IB Interface

To implement all the capabilities planned for the I/O ROM, it was necessary to design intelligent I/O cards. Each I/O card contains a single-chip microcomputer to (1) communicate with the HP-85 CPU and (2) control and respond to the signals on the interface. The result of this is highly significant—the I/O ROM is not required to know what type of interface is being accessed. Communication between the I/O ROM and an I/O card is totally independent of the type of card; it is the card's responsibility to perform the appropriate I/O function over the interface.

This enhanced capability of the I/O cards permits more capability to be put in the I/O ROM than would have otherwise been possible. Also, this design approach permits overlapped processing; that is, the HP-85 CPU can be running a BASIC program while an interface card is performing I/O operations. An excellent example of this design concept is the HP-85 interface to the HP-IB. The principal elements of the HP-IB card are shown in Fig. 3. They are:

1. Translator chip: The translator chip interfaces between the HP-85 CPU and the I/O processor on the card. It provides electrical and timing compatibility between the buses and contains several registers to implement the communications protocol between the CPU and the I/O processor.
2. I/O processor: The I/O processor is a single-chip microcomputer containing 2048 bytes of program ROM and 128 bytes of RAM. It performs two tasks: it communicates with the CPU through the translator chip to determine what I/O operation is desired, and it implements the I/O operation over the bus using the bus transceivers.
3. Bus transceivers and control logic: Bidirectional transceivers are used by the I/O processor to control and to respond to signals on the bus. A latch, written to by the I/O processor, controls the direction of the transceivers and also controls I/O processor interrupts.

The translator chip (Fig. 3), hereafter called the TC, was designed to achieve several goals:

- Support eight peripheral select codes
- Let the HP-85 processor interrupt the I/O processor
- Let the I/O processor interrupt the HP-85 CPU
- Let the HP-85 CPU do a hardware reset of the I/O processor
- Provide a means for the I/O processor to halt the HP-85 CPU
- Provide general-purpose data registers for bidirectional communications between the two processors.

I/O for the HP-85 is memory-mapped. To support eight select codes, the TC's address is switch-settable to one of eight different addresses. Three switches reside in the I/O card. In the process of setting the card's select code, the user is actually setting the card's address. A mapping is done in the I/O ROM to translate from the select code specified in the program to the appropriate address.

Each TC actually occupies a pair of addresses. The lower address is used to access the control register (write-only by the HP-85 CPU and read-only by the I/O processor) and the status register (read-only by the HP-85 CPU and write-only by the I/O processor). The upper address of the pair accesses the output buffer (write-only by the HP-85 CPU and read-only by the I/O processor) and the input buffer (read-only by the HP-85 CPU and write-only by the I/O processor).

The input and output buffers are used for general-purpose communications between the two processors. Bits in the control and status registers are used to qualify data in these buffers as well as report the status of various events. To synchronize the flow of data, each processor can ascertain the condition of these buffers via flags in the status and control registers. These flags are OBF (output buffer full) and IBF (input buffer full). OBF is set when the HP-85 CPU writes to the output buffer and is cleared when the I/O processor reads the output buffer. Similarly, IBF is set when the I/O processor writes to the input buffer and is cleared when the CPU reads the input buffer.

The I/O processor and HP-85 CPU exist in a master-slave relationship, with the CPU being the master. The CPU sends instructions and data to the I/O processor via the output buffer. The software protocol between the CPU and the I/O processor defines a bit in the control register as COM, for command. COM is set high by the CPU before writing a command into the output buffer; COM is set low by the CPU before writing data into the output buffer. While the I/O processor is acting on the data or command, it sets a bit in the status register to 1, which is defined as the BUSY bit. When the I/O processor finishes acting upon the command or data byte in the output buffer, it sets BUSY low, which tells the CPU that it is done.

Several commands received from the CPU require that

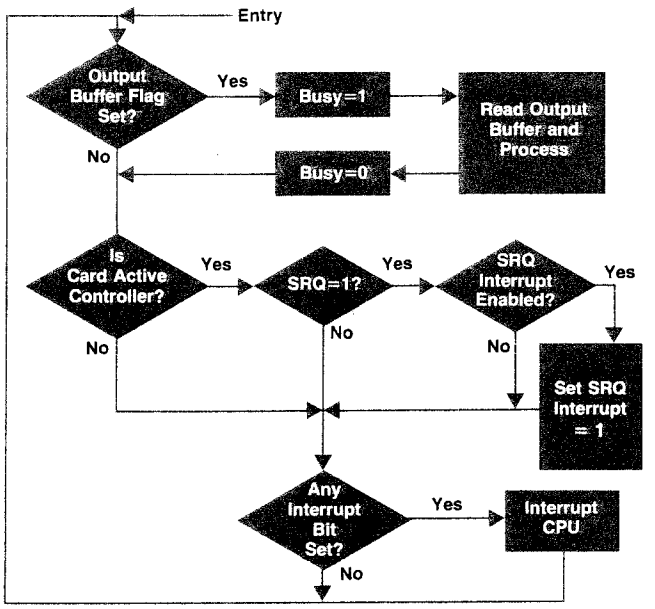


Fig. 4. *Idle loop for the 82937A HP-IB Interface card I/O processor. This loop can be exited at any point upon an interrupt from the CPU or interface bus.*

the I/O processor return data to the CPU (e.g., the command generated by the CPU to execute the user's STATUS statement). The I/O processor returns the data in the input buffer. The CPU monitors IBF to determine when the data is written to the input buffer, while the I/O processor monitors IBF to determine when the data has been read by the CPU.

Associated with each I/O ROM statement that accesses I/O is a set of rules that define the communications protocol between the HP-85 CPU and the I/O processor. The communications protocol not only defines the interaction discussed above, but also covers the interrupt protocol whereby each processor may interrupt the other.

The HP-85 CPU communicates with the I/O processor primarily to convey bus control commands to the I/O processor. The I/O processor then controls the bus as dictated by the CPU and within the bounds specified by IEEE Standard 488-1978. At power-on, the I/O processor reads the five address switches and the system controller switch located on the interface card. These switches are set by the user to configure the I/O processor's HP-IB address and to designate whether or not the I/O processor is to assume the role of system controller at power-on. The user can verify the switch settings by reading the STATUS register.

Fig. 4 shows the idle loop executed by the I/O processor, demonstrating its interaction with the CPU and the bus. In its idle loop, the I/O processor monitors OBF to see if the CPU has written any new information into it; if so, the I/O processor sets BUSY = 1, reads the output buffer and processes it as a command (COM = 1) or data (COM = 0).

If the I/O processor is the active controller of the HP-IB, it polls SRQ. If SRQ is true and the user has enabled an end-of-line interrupt branch on SRQ, then the SRQ end-of-line interrupt bit is set. The I/O processor subsequently examines all eight interrupt cause bits; if any are set, the I/O processor interrupts the CPU. While the SRQ interrupt cause bit is set as the result of polling, the other bits are set as the result of interrupts from the bus as discussed below.

Interrupts of the I/O processor originate from two sources—the CPU (via the TC), and the bus. The CPU interrupts for certain operations, such as STATUS, to guarantee timely operation regardless of the state of the I/O processor. For example, if the I/O processor is busy handshaking data on the bus to a device that is taking an indefinite length of time, then an interrupting STATUS operation guarantees an immediate return of data.

Certain HP-IB signals mandate a response within a time limit. For example, IFC (interface clear) must be responded to within 100 microseconds. To guarantee this, IFC can be enabled to interrupt the I/O processor. Likewise, REN (remote enable) must also be responded to within 100 microseconds, and thus can be used to interrupt the I/O processor. ATN (attention) imposes no timing requirements on the I/O processor but is used as an interrupt input to ensure that ongoing I/O is properly suspended. During an IFC or ATN interrupt, end-of-line interrupt bits may be set, depending upon the interrupt enable mask provided by the user. The interrupt enable bits are examined back in the idle loop; if any are set, the I/O processor interrupts the CPU. The CPU then typically executes the branch specified in the user's BASIC program.

Interface Select Codes and Device Specifiers

The HP-85 can have up to three interface cards plugged into it (see Fig. 5). It also has the internal CRT and printer that I/O ROM users may want to access. How does a programmer tell the I/O ROM which of these devices to interface with? This is done through *interface select codes*.

Every I/O ROM statement that deals with an interface specifies as part of the statement the interface select code. The range of select codes is as follows:

```
 1 : INTERNAL CRT
 2 : INTERNAL PRINTER
 3 : EXTERNAL I/O
 .
 .
10 : EXTERNAL I/O
```

Select codes 1 and 2 are accessible only through OUTPUT. An example of interface select codes is:

```
10 OUTPUT 1; "This line goes to the CRT"
20 OUTPUT 2; "This line goes to the printer"
30 OUTPUT 7; "This line goes to an external
              device"
```

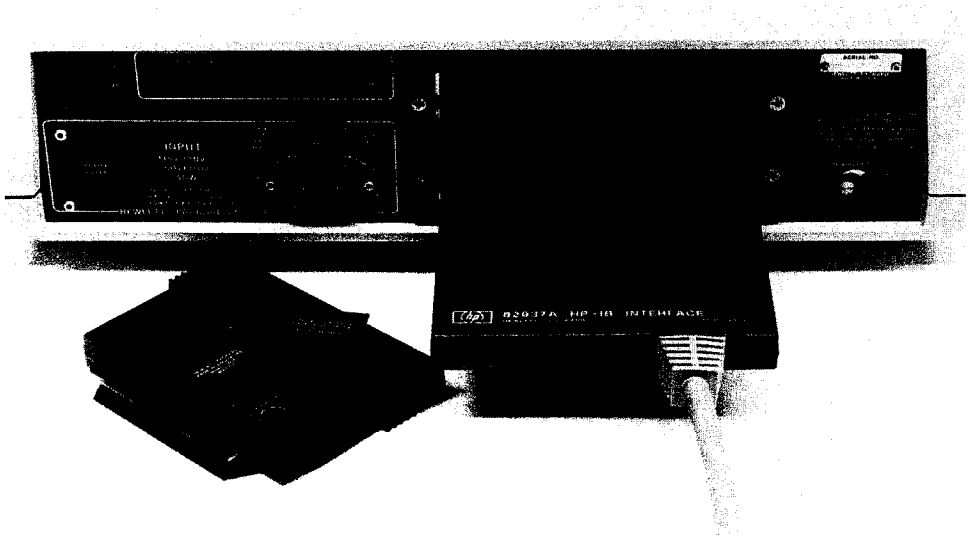


Fig. 5. *Typical HP-85 interface installation. The add-on ROM drawer and the 82937A HP-IB Interface are shown. The unused slots can be used to add two other interfaces or another interface and the 16K RAM module.*

Using HP-85 I/O Capabilities

The HP-85 I/O ROM allows for extreme ease of use. The following examples show a wide range of capabilities of the HP-85 in a simple interfacing application.

Let's assume that you, the user, have an HP-85, an I/O ROM, an HP-IB card, and a 3437A system voltmeter. Let's also assume that these components are hooked together, powered up, and working properly.

You would like to get a reading from the voltmeter. The 3437A's device address is 24 (as shipped from the factory) and the HP-IB interface card is select code 7 (as shipped from the factory). Therefore the address that you should be using is 724. You want to get a reading from the 3437A. To do that you have to execute the following statement:

```
ENTER 724;A
```

After you have executed this statement, the variable A will contain the reading from the voltmeter. That is all there is to do.

A program to get one reading and display it, would be like this:

```
20 ENTER 724;A
30 DISP A
50 END
```

Now suppose you want to take in a hundred readings and display them on the internal HP-85 CRT. The following program will do this:

```
10 FOR I=1 TO 100
20 ENTER 724;A
30 DISP A
40 NEXT I
50 END
```

Now suppose you want to plot these 100 readings. The following program does this:

```
 1 SCALE 1,100, -20,20
 2 GRAPH
 3 GCLEAR
 4 MOVE 1,0
10 FOR I=1 TO 100
20 ENTER 724;A
30 PLOT I,A
40 NEXT I
50 END
```

Now, say that this is not fast enough. Your application requires less time between voltmeter readings. First of all, you know that the 3437A is capable of much better performance. To get the HP-85 to take readings at a faster rate will require a type of data exchange known as a buffered transfer. This means that you tell the HP-85 to take the data into a buffer as fast as it can and then at a later time take the data out of the buffer and turn it into numeric data that the computer can use computationally. The following program does this fast transfer and plots the data as before.

```
 1 SCALE 1,100,-20,20
 2 GRAPH
 3 GCLEAR
 4 MOVE 1,0
 5 DIM B$[709]
 6 IOBUFFER B$
 7 OUTPUT 724;"N100S"
 8 ! THE PREVIOUS LINE CONFIGURES THE VOLTMETER
 9 TRANSFER 724 TO B$ FHS,COUNT 701
10 FOR I=1 TO 100
20 ENTER B$ USING "#,K";A
30 PLOT I,A
40 NEXT I
50 END
```

-Tim Mikkelsen

The select code of an I/O card is set by switches mounted on the card. These switches are preset at the factory, depending on the type of the interface (e.g., HP-IB's select code is preset to 7). An example where the programmer might want to change the select code would be when there are two HP-IB interfaces in one HP-85. The programmer might do this to extend the number of HP-IB devices on one computer, or if the computer is to be a controller on one HP-IB and just another device on the other. In such a case, the programmer would have to change one of the select codes to prevent a hardware conflict.

This, however, is not enough. The HP-IB standard allows for up to 31 device addresses on a single interface (14 is the physical limit that any HP-IB interface can support). The 16-bit general-purpose interface allows the programmer access to four eight-bit ports in various ways (using a total of sixteen logical addresses). The BCD interface allows for one or two channels of information consisting of data fields, function codes, and error indications which are accessible in various combinations (using a total of seven logical addresses). How does the programmer tell the I/O ROM which device address to talk to? This is done through *device specifiers.*

Tim I. Mikkelsen

Tim Mikkelsen came to HP in 1977 after completing BS and MS degrees in computer science at Iowa State University in 1975 and 1977, respectively. Currently a product marketing engineer, he has worked on I/O ROMs for the HP-85 and the 9835 and 9845 Computers. From Missouri Valley, Iowa, Tim is a member of IEEE, and now lives in Fort Collins, Colorado. He is married with one daughter. When not involved with woodworking, downhill skiing, personal computing, and photography, Tim is busy trying to restore a 1964 TR4 sportscar.

John H. Nairn

Born in Pittsburgh, Kansas, John Nairn went to school in Denver, Colorado, earning a BS in chemistry at Regis College in 1967. He then received the MS and PhD degrees in mathematical physics from Rice University, Houston, Texas in 1971. After working as a regional analyst John joined HP in 1972 and has had various responsibilities since then. Initially he worked in marketing on the 9821 and 9830 Calculators and then did R&D work for the 9825 I/O. In 1976, John returned to marketing as interfacing product manager. Later, in 1978, he moved to his current position as R&D project manager for the HP-85 I/O. John is a member of ACM, a co-inventor for a patent related to the 9825 Calculator, and author of a regular series of articles for HP's Keyboard magazine. He lives in Fort Collins, Colorado and is interested in photography, hiking, recreational mathematics and folk guitar.

David J. Sweetser
Born in Woodland, California, Dave Sweetser attended Harvey Mudd College, where he earned BS and MS degrees in engineering in 1971 and 1972, respectively. After five years with an aerospace company, Dave joined HP in 1977 and has designed an interface for the HP-85 in addition to writing I/O software. He and his wife, who is an engineer at HP's Loveland Instrument Division, designed their own house, which is presently being built on 3 acres of land between Loveland and Fort Collins, Colorado. Dave enjoys rafting, bicycling, backpacking, cross-country skiing and snow camping.

The device specifier is like an interface select code, except that it includes both the select code and the device address. Notice that the range of device addresses is from 00 to 31. To build a device specifier—multiply the select code by 100 and add the device address. Hence select code 7, device 24 would be $7\times100+24 = 724$. Device addresses are generally preset by the factory, but can be changed in a manner similar to changing the interface select code.

References

1. S.L. Chumbley, "Extending Possibilities in Desktop Computing," Hewlett-Packard Journal, May 1979.
2. W.D. Eads and J.M. Walden, "A Highly Integrated Desktop Computer System," Hewlett-Packard Journal, April 1978.
3. D.E. Morris, C.J. Christopher, G.W. Chance, and D.B. Barney, "Third Generation Programmable Calculator Has Computer-Like Capabilities," Hewlett-Packard Journal, June 1976.

A Compact Tape Transport Subassembly Designed for Reliability and Low Cost

by Douglas J. Collins and Brian G. Spreadbury

OVER THE LAST FOUR YEARS Hewlett-Packard has developed a series of magnetic transports[1,2,3] for the 98200A data cartridge shown in Fig. 1. The use of this small data cartridge was selected because of proven reliability, large data capacity, ease of use, and low system cost.

The tape system integral to the HP-85 is in many ways a refinement of previous designs. However, the total integration of the transport with a single printed circuit board into one small package called for new techniques in electrical and mechanical design.

Transport Electronics Design

To incorporate a tape system into the compact HP-85 package required the cartridge transport to be small and to consume a minimum of power. Unfortunately, a relatively powerful motor is needed to drive the tape. For best efficiency, a pulse-width-modulated 20-kHz signal controls the motor drive transistors to keep them either cut off or fully saturated (turned on). These large power pulses can generate severe noise levels on the ten-millivolt signal lines from the tape head. To prevent this, the entire read/write circuitry is located within three centimetres of the magnetic head. Also, the motor drive circuitry has a separate power supply and is independently grounded.

The heart of the tape system electronics is the custom NMOS tape controller IC. It performs the tasks of interfacing to the HP-85 CPU, controlling motor speed and direction, and encoding/decoding data for the tape. Two Schmitt-trigger inputs provide direct sensing of the analog signals

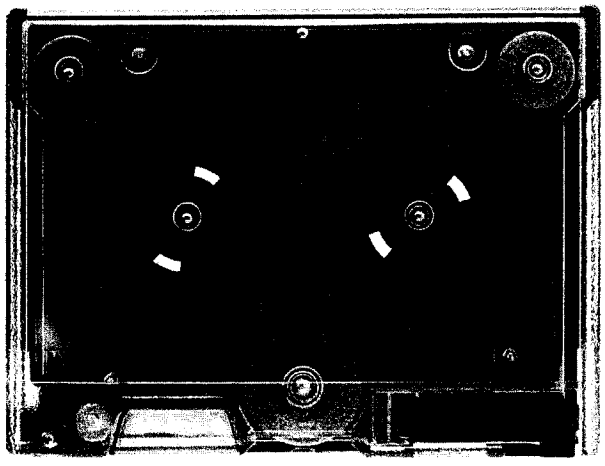


Fig. 1. *The 98200A mini data cartridge is a compact storage medium for data and programs, and has proven reliability and flexibility.*

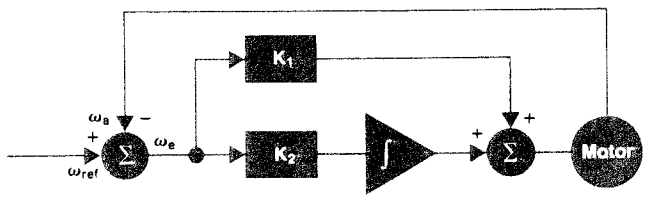


Fig. 2. *Flow diagram for a traditional speed control design.*

from the phototransistors that are used for optical sensing of the end-of-tape hole and the motor tachometer disc. Despite all of the functions included on this IC chip, it remains remarkably small (approximately 14 square millimetres in area) and typically dissipates about fifty milliwatts of power.

To keep the size of this IC small, the simple calculations required for tape operation are delegated to the HP-85 central processor. Functions such as gap length measurement, checksum generation, and end-of-tape detection are performed by the CPU. These computations are trivial for the CPU, but would require the addition of multibyte adders and counters to the tape controller IC if it were required to do such functions.

Digital Servo Design

To help understand the digital servo system used here, consider the traditional position control system shown in Fig. 2. For this case, the cumulative error ω_e is held to zero by integration, thereby providing no long-term speed error. The system is given stability by feed-forward compensation K_1. To transform this system into a digital system, the integrator and its associated summing junction are replaced by an up/down counter C1 and a binary adder as shown in Fig. 3.

An optical tachometer generates a series of pulses that are fed into down counter C2. On each tachometer pulse the counter is loaded with the desired reference period (the reciprocal of the desired tachometer frequency). Counter C2 counts down until the next tachometer pulse is detected and then its output is evaluated. If the sign of the output is negative, the time between successive tachometer pulses is too long, indicating a low motor speed. This causes C1 to count up, which generates a longer duty cycle for the motor drive pulse-width output. If the output of C2 is positive, C1 counts down and the motor slows down. The value of the C2 output is scaled and fed forward through the adder to stabilize the system. Since the servo output is recalculated for each tachometer pulse, the overall gain of the system varies with the speed of the motor. However, if the servo is operating properly, the speed variation will be small and the gain modulation will be negligible.

There are only two speeds required for the HP-85 tape system—254 and 1524 mm/s (10 and 60 in/s). To change the speed, the reference period is modified and scaling adjustments are made to the servo timing to compensate for the change in gain.

The servo is run open-loop to start or stop the motor. The acceleration or deceleration is controlled by incrementing or decrementing C1 and feeding its output directly to the pulse-width modulator. No additional counters are required.

The digital pulse-width modulator directly interfaces to four Darlington-transistor motor drivers. To protect the motor and drivers during high load conditions, a table lookup ROM is used to solve the defining equation for a direct-current, permanent-magnet motor.

$$V = R_a \times I_a + K_e \times \omega_a$$

where V is the voltage applied to the motor (proportional to duty cycle), R_a is the armature resistance, I_a is the armature current, K_e is the back EMF constant, and ω_a is the motor's angular velocity.

Given the maximum armature current and motor constants R_a and K_e, the ROM is programmed to solve for the maximum duty cycle allowed to drive the motor. The output of C2 in Fig. 3 is used as the input to this ROM because its reciprocal is proportional to the motor's angular velocity. For each tachometer pulse, the maximum allowable duty cycle is calculated and compared to the servo output. The lesser of the two values is output to the motor drive circuitry. A delay is added between the time of excessive load detection and the time the armature current is actually

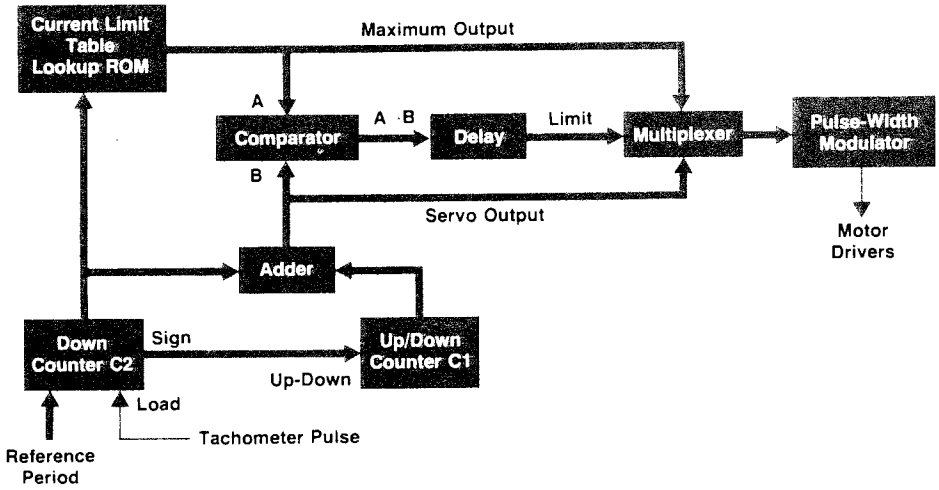


Fig. 3. *Block diagram of digital servo electronics for monitoring and controlling the tape drive motor speed.*

limited to prevent short-term speed perturbations from affecting the motor speed. This eliminates the need for a current-sense resistor and associated detection circuitry that would sacrifice motor efficiency.

Additionally, the motor is protected against a stall condition by shutting the motor down if the tachometer frequency becomes excessively low. Both the stall and current-limit conditions are reflected in the tape controller's status, which is monitored by the HP-85 system firmware.

Data Coding

Information is transferred to and from the tape controller IC on a byte-by-byte basis. Data written onto the tape is first precompensated (lengthened or shortened, dependent on the value of adjacent bits) to prevent magnetic pulse crowding on the tape. A delta-distance code with a ratio of 1.75:1 is used.[4] That is, the distance between magnetic flux reversals on the tape determines the value of the bit, with the distance for a one being 1.75 times greater than that for a zero.

The decoder diagrammed in Fig. 4 is a speed tolerant design that compares the current bit pulse width with a continually updated nominal pulse width contained in the up/down counter. The down counter is essentially used as a variable modulus counter set by the contents of the up/down counter. At each magnetic flux reversal, the down counter is loaded with the nominal period for a zero bit. When this quantity is counted down to zero, the state counter is incremented and the down counter is reloaded. With the decoder in state one now, one-fourth of the nominal zero-bit period is counted down and the down counter is reloaded again. The next decoder state counts down one half of the nominal zero-bit period. The value of the pulse being decoded is determined by the current state of the decoder when the next flux reversal occurs. State zero indicates a shorter-than-nominal zero-bit period and so decrements the current count for the nominal period. State one is for a long zero-bit period and increments the nominal count. States two and three correspond to short and long one-bit periods, respectively. The transition from state two to state three happens after the last flux reversal at exactly 1.75 times the nominal zero-bit period. In this manner, the 1.75:1 delta-distance code can be decoded with a minimum of logic circuitry.

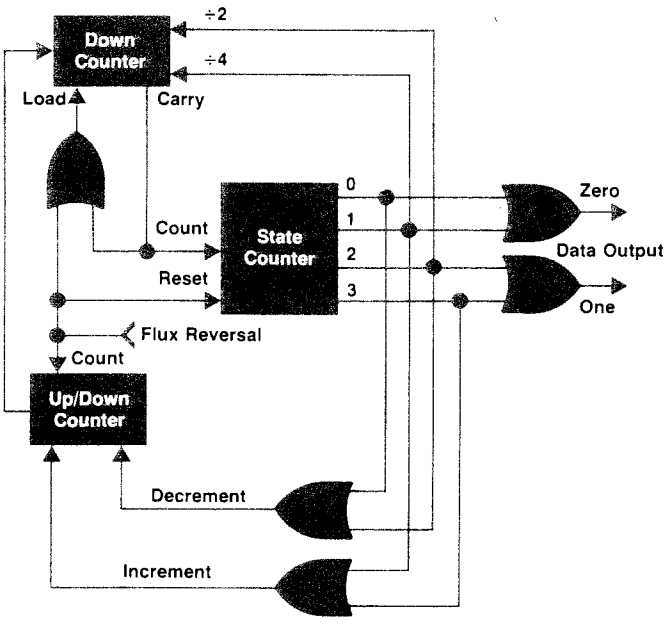


Fig. 4. *Logic diagram for the decoder circuit that converts the time intervals between successive flux reversals on the tape into the equivalent binary data.*

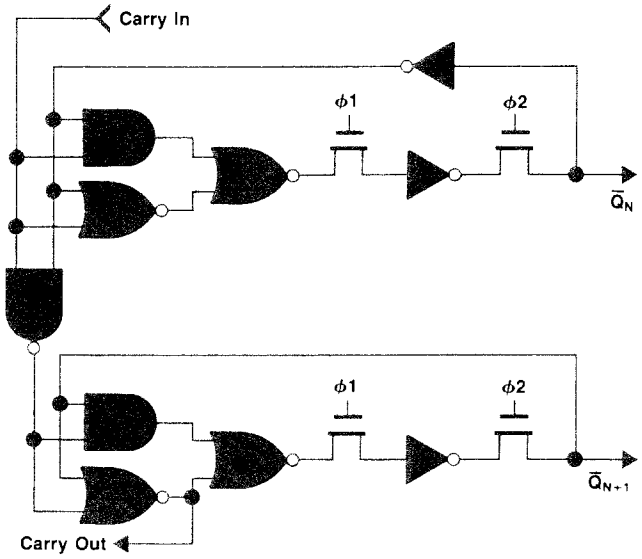


Fig. 5. *The dynamically clocked logic design for counter stages which achieve optimum speed for minimum IC chip area.*

Logic Design

To implement all this with a custom NMOS LSI circuit design, the various circuits must be optimized for the required speed with the lowest number of transistors. Gate level minimization does not apply here. However, the number of gate delays for any circuit path should not be too many, otherwise chip area and power will be wasted.

The counters in the tape controller IC are implemented using an alternating positive/negative logic scheme that is dynamically clocked as drawn in Fig. 5. The sense of the carry is inverted for each bit into the counter. This counter has one gate delay per stage (from the carry), which is optimum for the number of stages required.

Data is stored by the NMOS capacitances in the counter and is refreshed by two nonoverlapping clocks. The use of two clocks prevents possible race conditions.

Read/Write Amplifier

A single bipolar read/write amplifier IC transforms coded digital information from the tape controller IC into magnetic pulses on the tape, and back again. The basic read circuit is diagrammed in Fig. 6. The configuration of the two magnetic-head read coils permits the use of shared input and feedback components for both tape tracks, thus reducing the number of external parts needed. Each track has its own differential transistor pair, but shares a common voltage amplifier. Track selection is accomplished by gating the respective current source for the desired input pair.

After preamplification, the head signal is processed by a peak detector and a bilateral threshold detector. Signal peak detection is done with a differentiator and a zero-crossing detector. Because of the low signal-to-noise ratio for the flux reversals on the tape, the peak detection method can generate false transitions between true signal peaks. This problem is eliminated by gating the peak detector's output with the threshold sensing circuit. The relationship between the various signals is illustrated in Fig. 7.

The current sources and switches necessary for writing data onto the tape use only a single external resistor to accurately set the write-current level.

Two incandescent lamps are used as light sources for the optical sensing of the end-of-tape hole and the motor tachometer disc by the phototransistors. To save power, the lamps are turned on only while the tape transport is being used. The bipolar read/write amplifier IC also contains a turn-on current surge limit circuit for these lamps. This circuit assures good lamp reliability, even after thousands of on-off cycles.

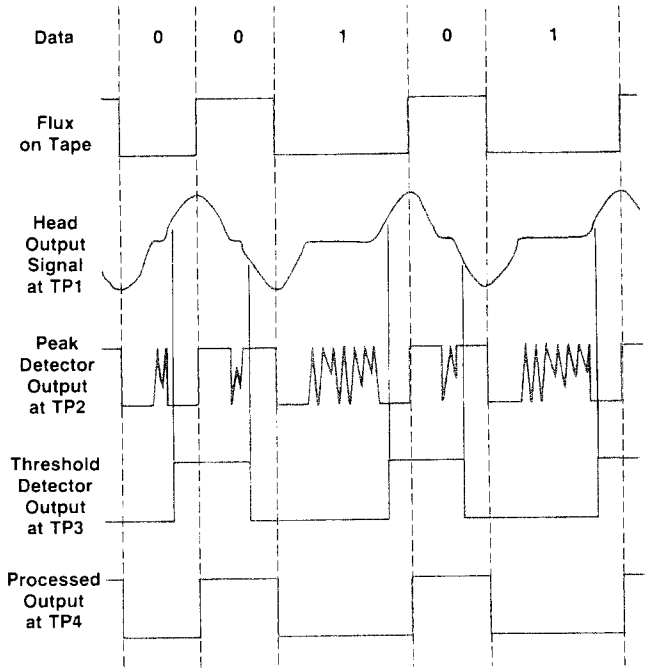


Fig. 7. *Time relationship between the various signals involved in reading data from the tape. See Fig. 6 for the location of these signals in the read/write amplifier and detector circuit.*

Mechanical Design

The design objectives for the tape transport emphasized flexibility, reliability, and low cost. Cost reduction was a must, and the flexibility and reliability were to be improved if possible.

The initial observation of the existing drive designs and an analysis of costs indicated that any cost reduction would be hard-won indeed, and that only some basic rethinking of the total design would be productive. The resulting design integrates the mechanical and electrical elements to a greater degree than previously achieved, reduces the number and complexity of components, and considerably lowers the labor content.

The primary requirement for the transport, from a

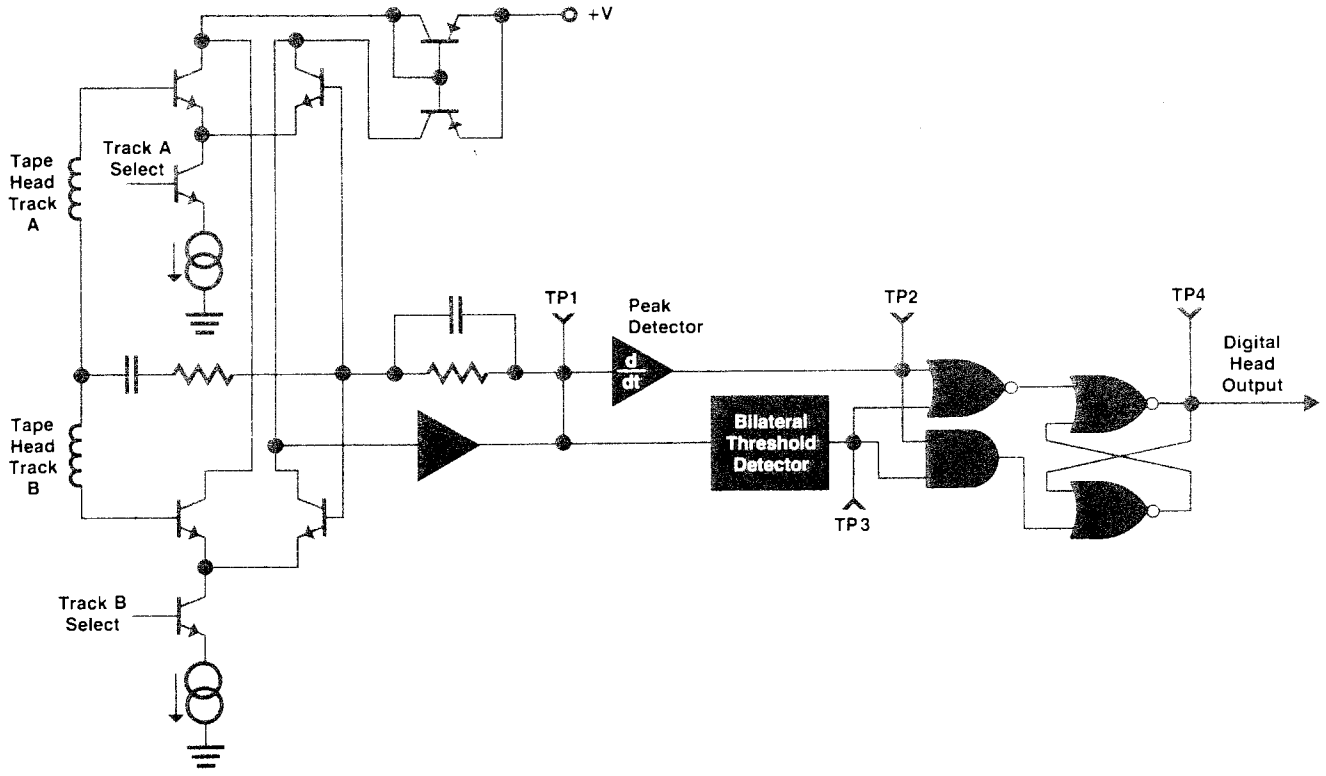


Fig. 6. *Schematic for the tape read/write amplifier and detector. Some components are shared for both tape tracks. Tracks are selected by gating the current source for the differential pair associated with the track selected.*

mechanical point of view, is to positively maintain the cartridge position with respect to the magnetic head, while driving the tape roller at the desired speed under varying conditions of voltage, temperature, humidity, and loading. From this basic premise, an assembly evolved that is installed in the mainframe with only two mounting screws, and has only two flat-ribbon cables for subsequent connection to the logic board assembly. All other control functions are contained within the transport subassembly (Fig. 8).

The transport unit consists of three field-replaceable modular subassemblies that require only simple tools for installation. They are the baseplate assembly, the printed circuit board assembly, and the motor/capstan assembly.

The baseplate assembly consists of the cartridge ejection mechanism, the baseplate (which acts as the support for the ejector), the motor, and the head bridge. The head bridge performs the multiple functions of cartridge guidance, magnetic head location, and housing of the leaf switches that sense the presence of a cartridge and the condition for record protect.[5] To maintain tape-to-head contact and tracking and azimuth alignment within very close tolerances, it was decided that optical alignment of the head with respect to the head bridge guide rails was necessary. Tape-to-head wrap must be held closely to prevent either loss of intimate contact because of insufficient wrap, or excessive tape or head wear caused by too much wrap. This is controlled by using locating pins in the head bridge that mate with corresponding holes in the baseplate. The motor assembly is similarly located on the baseplate. The head alignment process positions the head within a pocket in the head bridge, using fixtures that engage the locating pins to control head penetration. After adjustment for azimuth and tracking, the head is bonded in place with fast-set acrylic adhesive. This enables fast turnover of the alignment fixtures in production.

The loaded printed circuit board is directly mounted to the rear of the transport mechanism and engages the magnetic head leads, which are 24 AWG solid-copper tinned wire. This greatly facilitates their alignment with the six oversize holes in the circuit board and eliminates the need for a head wiring harness.

Functions such as cartridge in/out and write enable are sensed by plunger-operated leaf switches mounted in the head bridge. The switches pick up contact pads on the circuit board. By careful insertion of phototransistors and lamps into the board, using jigs, end-of-tape detection and tachometer speed control are achieved without any wiring since the components align exactly with physical locating holes molded into the baseplate and head bridge.

The motor/capstan assembly consists of a conventional direct-current motor with a polyurethane-on-aluminum capstan and attached mylar tachometer disc. The disc is used for servo feedback of motor angular velocity as discussed earlier in this article. The disc is mounted together with a miniature incandescent lamp within a plastic motor mount that locates the motor relative to the baseplate. The motor leads plug directly into the printed circuit board that contains all of the motor drive circuitry.

Acknowledgments

Mike Barbour completed the read/write amplifier, using a fixed array IC. The final custom design was carried out by Mike Moore. The test program was developed by Tran Lung and processing assistance was given by Javid Bajwa, Al Wang, and Irene Rhodes. The tool design and head alignment fixtures for the tape unit were provided through the

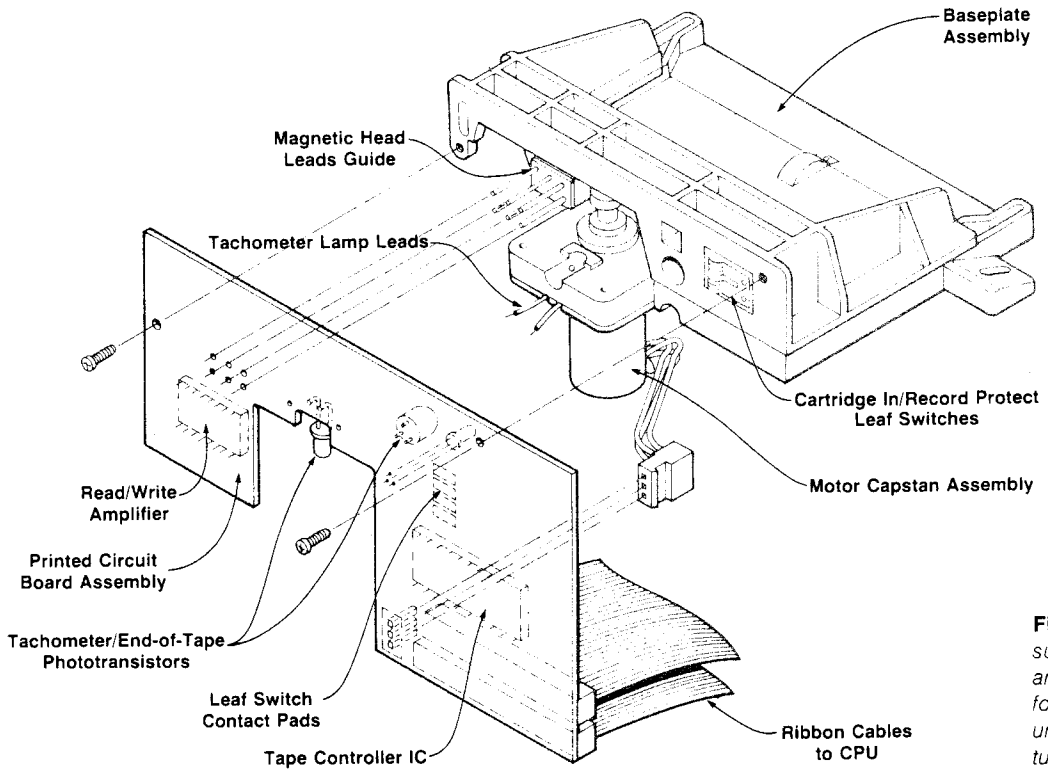


Fig. 8. *Tape cartridge transport subassembly. The mechanical and electrical designs combine to form a compact, highly reliable unit that can be easily manufactured.*

Douglas J. Collins

Doug Collins graduated from Purdue University with a BSEE degree in 1973 and went on to receive his MSEE from the University of Illinois in 1975. A week after joining HP that same year, he began initial work on the HP-85. Doug helped develop the product configuration, was responsible for the tape transport electronics, designed the tape controller IC, and is currently production engineer for the HP-85. A native of Glen Ellyn, Illinois, he and his wife now live in the countryside near Corvallis, Oregon. He has a private pilot's license and plays on the local volleyball team in addition to his many other interests—sailing, bicycling, hiking, skiing, and woodworking.

Brian G. Spreadbury

A native of Aldershot, England, Brian Spreadbury was educated at Bournemouth College, England where he received the BSc-Mech in 1962. After working for several companies on various aviation system designs, he came to HP in 1976 and is presently a production engineer for the HP-85. Brian earlier worked in R&D on the tape transport design for the HP-85. He now lives in Corvallis, Oregon with his wife and two sons. He helps out at Boy Scout activities, and he enjoys hiking, fishing, and camping when it's warm and cross-country skiing when the weather turns colder.

help of Mark Matsler and Bill Shumate. Ray Steffen helped coordinate the production of the subassembly.

References

1. R.G. Nordman, R.L. Smith, and L.A. Witkin, "New CRT Terminal Has Magnetic Tape Storage for Expanded Capability," Hewlett-Packard Journal, May 1976.
2. D.M. Clifford, F.T. Hickenlooper, and A.C. Mortensen, "Mid-Range Calculator Delivers More Power at Lower Cost," Hewlett-Packard Journal, June 1976.
3. R. Kochis, "System 45 Tape Control System," Hewlett-Packard Journal, April 1978.
4. "9825A Cartridge Tape Unit," Hewlett-Packard Journal, June 1976, page 13.
5. A.J. Richards, "Mini Data Cartridge: A Convincing Alternative for Low-Cost, Removable Storage," Hewlett-Packard Journal, May 1976.

A High-Quality CRT Display for a Portable Computer

by James F. Bausch

ONE OF THE TWO PRIMARY output peripherals integrated into the HP-85 Personal Computer is a high-quality CRT display. The other, a thermal printer, is described elsewhere in this issue. In addition to design goals of display quality and small size, the HP-85 CRT subassembly was to have low power consumption.

The 127-mm diagonal, black-and-white, electromagnetic-deflection CRT can display information in the form of lines of alphanumeric characters or provide a plotting area for graphics output. During tape operation and thermal printing, the display is switched off to conserve power. A photograph of the subassembly is given in Fig. 1, and Fig. 2 shows the block diagram of the CRT subassembly.

CRT Controller

The CRT drive system employs an NMOS controller IC whose function is twofold. Its primary function is to interact with the mainframe memory, CRT, and display refresh memory so that data may be displayed as it is updated by the user. Its secondary function is to provide signals to the CRT drive circuitry for horizontal and vertical timing and character generation.

The alphanumeric display font uses a 5×7 dot matrix in an 8×12 dot array. Characters are displayed 32 per line,

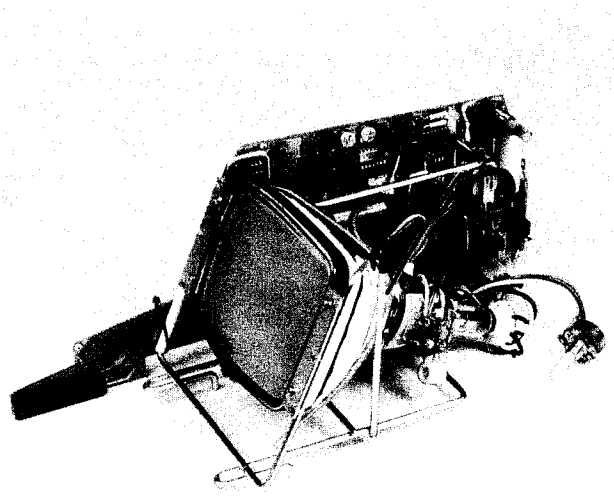


Fig. 1. *Photo of CRT subassembly. This unit contains a 127-mm CRT with associated control electronics and can be independently tested before final assembly into an HP-85 system.*

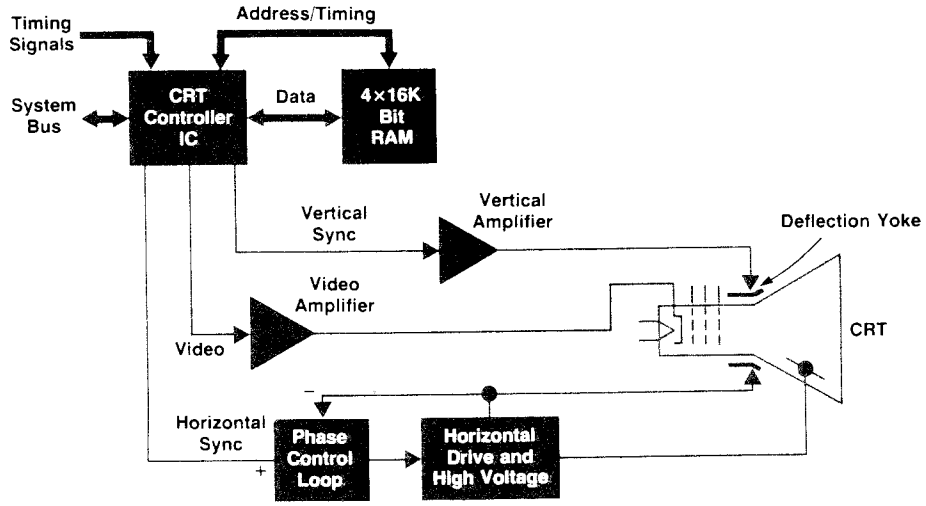


Fig. 2. *Block diagram of the CRT subassembly for the HP-85. The heart of this unit is the CRT controller IC, which generates timing signals, draws characters, and interfaces to the main system.*

with 16 lines in a full display. The graphics display format is the full screen, 256 dots wide by 192 dots high.

The refresh memory stores all of the information necessary to display four screens (64 lines) of alphanumeric data plus one screen of graphics data. The alphanumeric information is stored as the ASCII equivalent of each character. The graphics data maps directly from one bit of RAM to one dot on the screen. Three-quarters of the available refresh RAM is dedicated to graphic information storage.

Vertical Amplifier

In the diagram of the vertical amplifier (Fig. 3), a pair of class AB amplifiers, operating in a push-pull mode, provide the large peak-to-peak excursions required to reverse the currents in the CRT deflection yoke within the allotted vertical frame time interval. The CRT controller IC provides a frame synchronization pulse at the vertical frame rate. The shaping generator provides a current waveform whose profile compensates for the larger writing velocities at the edges of the screen. This waveform is a ramp from zero to eight volts over a period of 12.5 milliseconds. The slope at the beginning and end of this ramp is more gradual (80%) than the slope during the major portion of the ramp. This lets the beam travel at a uniform rate across the CRT.

Resistors R1 to R4 provide current feedback from the ten-ohm current-sense resistor. These resistors affect the output impedance of the amplifier and must be carefully selected. Too large a loop gain causes instability; too small causes excessive rise times. Resistor values are controlled and adequately matched by fabricating them all in a single package, using thick-film technology.

Horizontal Amplifier

In the design of the horizontal amplifier (Fig. 4), most of the circuitry is conventional. However, in the interest of power conservation, and because a 127-mm CRT must have very small border margins, a phase control loop was added. Its function is to assure beam position with respect to dot timing. With this design, the overscanning normally required to compensate for phase errors becomes unnecessary, and drive current is reduced.

The circuit uses a conventional 555 timer that triggers on the leading edge of the negative-going horizontal sync pulse. The timer then outputs a pulse to the driver transformer, turning transistor Q1 on and initiating the scan. Later the phase detector flip-flop is set by the trailing edge of the horizontal sync pulse and is almost instantly cleared by the pulse generated by the ringing of the deflection coil circuit and retrace capacitor C1. The short output spike thus generated at the $\overline{Q}$ output of the phase detector flip flop modulates the timer to vary the trailing edge of its output pulse. The variation anticipates the storage time τ_s of transistor Q1 and turns it off early so that the edges of the horizontal sync and flyback pulses are aligned. Phase errors of less than 200 nanoseconds are achievable for the interval between the horizontal sync pulse and the flyback pulse. Other schemes have used phase-locked voltage-controlled oscillators, but this phase control loop has improved stability because of the absence of a pole at the origin. As with the

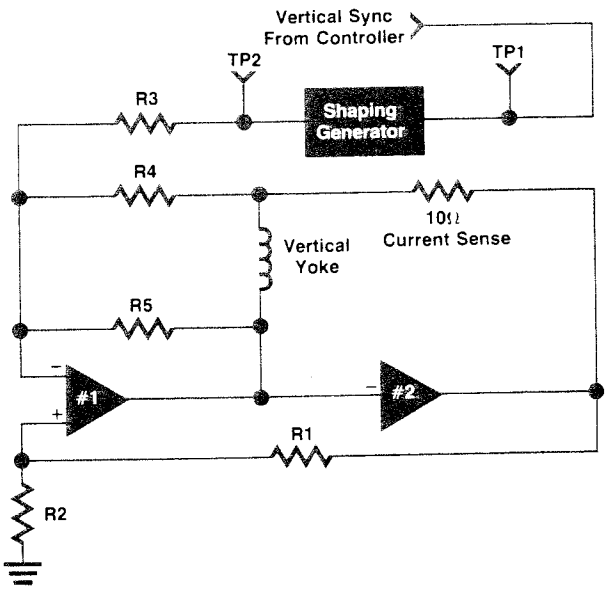


Fig. 3. *Vertical amplifier circuit. The two amplifiers are class AB designs operating in phase opposition, and the resistors are used for feedback from the current sense resistor. The shaping generator compensates for higher writing velocities at the edges of the display.*

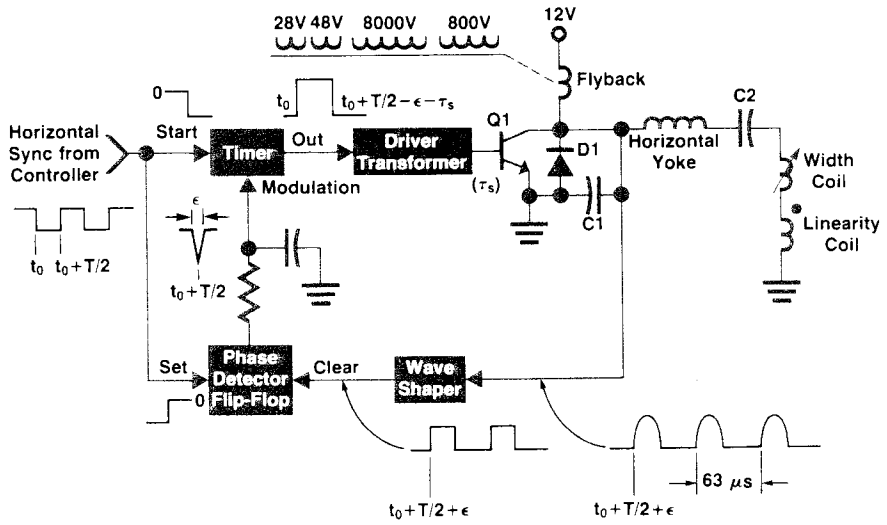


Fig. 4. *Horizontal amplifier circuit. The phase control loop is used to maintain the beam position with respect to dot timing. The flyback autotransformer provides the high potentials required for CRT operation.*

phase-locked loop, however, noise must be carefully controlled since any phase noise can be magnified on succeeding raster lines. The timer's power supply is locally regulated and decoupled.

The shaping capacitor C2 performs the same function as the shaping generator does in the vertical amplifier circuit. The width coil adjusts the overall length of the horizontal track and the linearity coil cancels the effective resistance of the scan circuit path. The flyback transformer secondary windings provide all of the high voltages required for brightness and focus (800 volts), final anode (8000 volts), and grid #1 (48 volts) potentials.

The use of the grid #2 potential as the means for brightness control is strongly recommended by the CRT manufacturer. Small mechanical variations in the grid #1-to-cathode spacing cause large variations in the cutoff voltage (the voltage required to suppress the beam current of the CRT). These variations can be reduced by adjusting the brightness with an adjustment of the voltage on grid #2. A secondary effect of this approach is a decrease in the overall spot size. Spot size is reduced because the lower cutoff voltage results in a smaller cathode emission area. Also, variations in drive voltage are not required for this "constant-cutoff" method.

The disadvantage of this design is the high potential (approximately 800 volts) that must be routed to the brightness control. A 19-mm nylon shaft from the control to the rear panel provides the user with sufficient isolation from this potential.

Video Amplifier

The video amplifier (Fig. 5) achieves low-power opera-

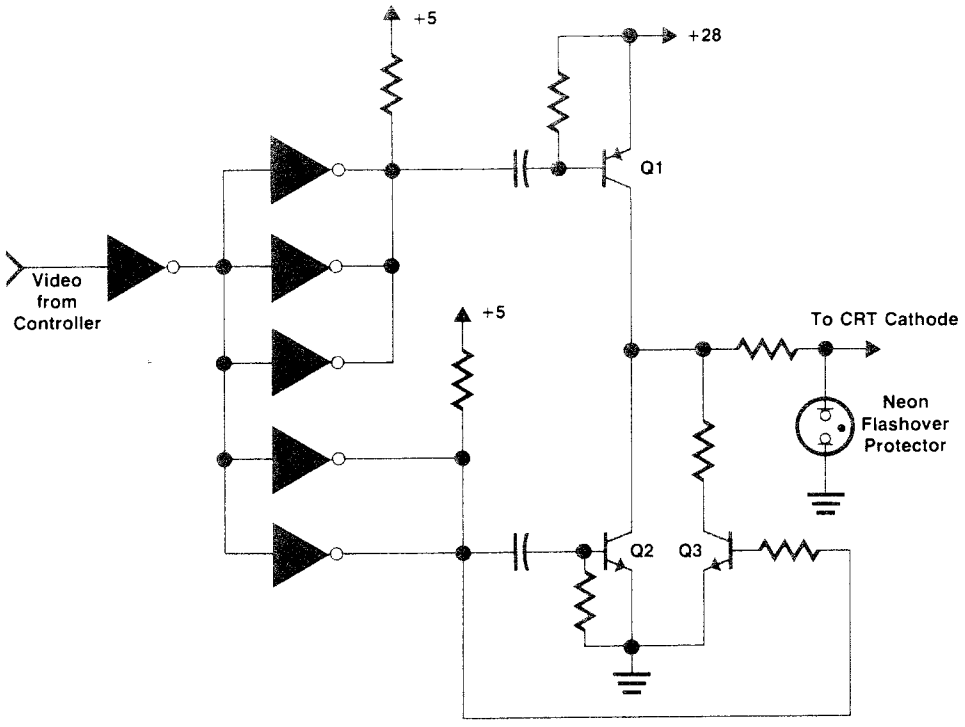


Fig. 5. *Video amplifier circuit. The TTL drivers on the left prevent Q1 and Q2 from going into saturation. Class C operation is used to conserve power. Q3 is used to maintain beam intensity when displaying long lines.*

tion by using a class C design. Class C amplifiers are very efficient since the output stage is either on or off. The power used is primarily CV^2f, where C (here 30 picofarads) is the cathode and lead capacitance, V is the cutoff voltage (28 volts), and f is the transition frequency of each dot (since each dot is 200 nanoseconds wide, the maximum transition frequency is 2.5 MHz). This power would be at most sixty milliwatts if a group of vertical parallel bars were to be displayed on the screen.

A problem inherent in most class C amplifiers is the saturation of the transistors. Saturation limits rise times by increasing the charge storage times for the on transistors. In the HP-85, saturation is prevented during switching by using TTL drivers that are ac coupled to transistors Q1 and Q2. These complementary transistors are always off until transitions occur. There is always some display on the screen (cursor) to provide some beam current. Hence, the CRT cathode normally assumes an off state due to the beam current generating a positive bias, and the complementary pair merely switches the cathode to an on state when required.

The cathode capacitance holds it to a low voltage state in the alphanumeric display mode, where changes between states are more frequent. In the graphics mode, with longer lines required for the display, the beam current required causes the cathode voltage to droop due to increasing bias level. An additional transistor, Q3, operating in saturation but with a large collector resistor, clamps the bias at the dc potential necessary to compensate for the beam current drift.

James F. Bausch

Jim Bausch has investigated displays and small power supplies for handheld computers since coming to HP in 1975. He is one of the co-inventors for a patent application on the HP-85 power supply transformer bobbin design. Jim was born in Cincinnati, Ohio. After receiving a BSEE degree from Georgia Institute of Technology in 1967, he did graduate work at the University of Cincinnati from 1967-1969. He and his wife and two boys now live in Corvallis, Oregon. Jim's interests include playing bridge, fishing, camping, and photography.

Acknowledgments

George Crow, Data Terminals Division, aided in design of the power supply and CRT circuits. Burk Brandt helped on product safety, and Jerry Erickson designed and developed the CRT controller IC.

A Compact Thermal Printer Designed for Integration into a Personal Computer

by Clement C. Lo and Ronald W. Keil

BUILT INTO THE HP-85 Personal Computer is a moving-head, thermal printer/plotter. This unit is a compact, medium-speed, *smart* peripheral that quietly outputs program listings or hard copies of the alphanumeric and graphics data displayed on the CRT. The printer is designed to print two lines per second for alphanumeric copy. Each line can have up to 32 characters. For graphics copy, the plot displayed on the CRT is rotated 90° before it is printed. This provides the capability for forming continuous strip charts by stringing successive plots end-to-end.

The main objectives of the new design were higher printing speed and a lower manufacturing cost than achieved for earlier printers. The former objective was met by using more sophisticated electronics and by printhead and drive development. The latter was met by simplification of assembly, reduction of the number of parts, and by using identical parts in several places. Fig. 1 is a photograph of the printing mechanism with its companion printed circuit board. This board contains all of the printer electronics as well as the HP-85 power supply described on page 24.

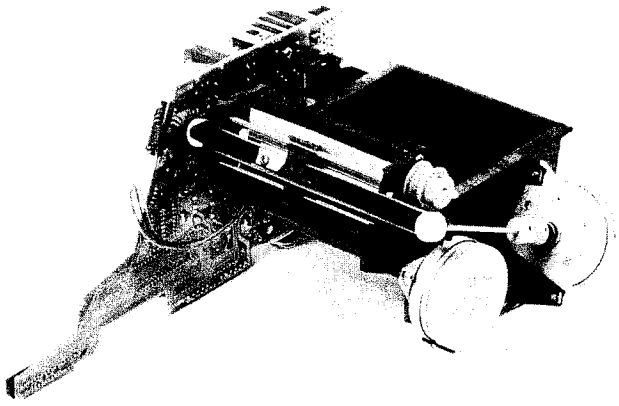


Fig. 1. *HP-85 thermal printer subassembly. In addition to the printing mechanism, this subassembly also contains both the support electronics for printing and the circuitry for the HP-85 power supply.*

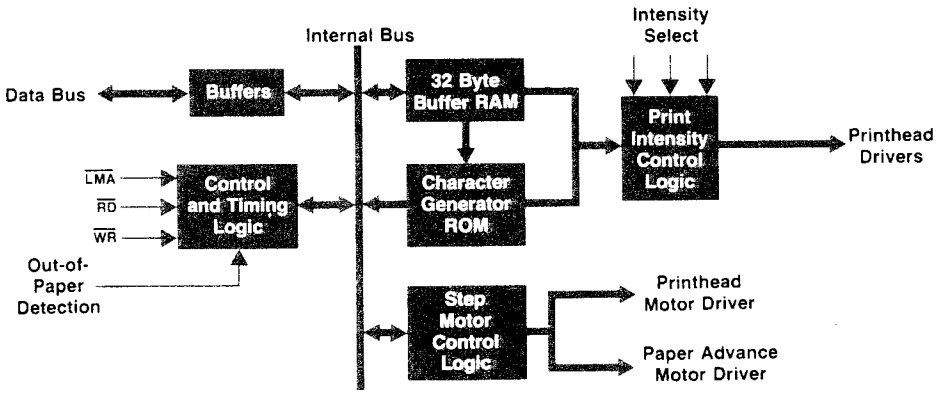


Fig. 2. *Block diagram for the printer controller integrated circuit. This chip controls the printer mechanism motors and print intensity, buffers the data to be printed, and handles control and timing functions.*

Printer Electronics

Consistent with the overall system integration philosophy, all of the electronics required, except for the high-current drivers, were integrated into one NMOS integrated circuit—the printer controller. This IC provides internal buffering of up to 32 characters and a 5×7 dot matrix character generator. The information stored in the internal character ROM can be accessed by the system software and thus the character lookup table in the system can be eliminated. Both the character set and the intensity settings are mask programmable.

The printhead consists of eight thin-film resistors arranged in a vertical column. The entire printhead is moved across the paper so that when the correct resistors are activated at specific times, a character is formed.

An eight-position binary switch is connected to the printer controller chip so the user can select any one of the eight preprogrammed print intensities, with binary zero being the lightest. The chip controls the darkness of the printout by varying the pulse duration of the signals applied to the printhead resistors. To obtain uniform contrast, the controller also adjusts the pulse duration to account for different dot densities. These cover such a wide range that the user can still obtain a printout with acceptable quality under all usual operating conditions.

The controller is capable of printing bidirectionally for alphanumeric copy and unidirectionally for graphics copy. To maximize the print speed, a look-ahead algorithm is built into the chip. The controller decides which direction to print by looking at the present line length and the next line length. A block diagram of the printer controller IC is given in Fig. 2.

The printhead assembly is moved across the paper by the head drive motor. This motion uses a 7.5° (48 steps) step motor that drives the printhead assembly with a toothed belt. The resolution of the motor is doubled by driving it in a half-step mode (3.75°/step). Each step corresponds to one dot column of printout. The motor is driven at 448 steps per second, which is equivalent to a print speed of sixty-four characters per second. Paper feed is accomplished by use of an additional step motor. It operates in a full-step mode (7.5°/step) and steps at one-third the rate of the head drive motor. Each step corresponds to one dot row of paper advanced. When the controller finishes printing a line, it automatically advances the paper. For alphanumeric copy, the paper advances ten dot rows, spacing three dots between lines. For graphics copy, it advances only eight dot rows; thus it is possible to print a continuous vertical line for graphics plots.

There is no home switch designed into the printer. At the first PRINT command after system power-on, the system firmware will send a home command to the printer controller. Knowing that this is the first home command received since power-on, the controller drives the printhead assembly all the way to the left. The assembly moves toward and eventually contacts the left wall of the frame. After the motor has been given 256 pulses, the controller then reverses the direction of the motor and moves the head assembly four steps to the right. Then it turns off the power applied to the motor and resets all internal position keeping logic. In this manner, a home position is established.

The energy stored in the motor inductance is returned to the unregulated 32Vdc supply through the catch diode (See Fig. 3). This not only increases the power efficiency of the system, but also lets us drive the step motor at a higher rate. The field built up in the motor can be collapsed faster by returning the current to a high-voltage supply.

Thermal Printhead

The printhead contains eight resistors, arranged in a single column, with an overall height equal to one character. Each resistor is 0.36 mm (0.014 in) tall by 0.28 mm (0.011 in) wide. In operation, the resistors scan over a thermal-sensitive paper, creating a colored dot on the paper whenever a resistor is energized. A small amount of lateral elongation occurs during printing because of the movement of the printhead acoss the paper before the resistors completely cool down. This widens the dot, making it appear to be square.

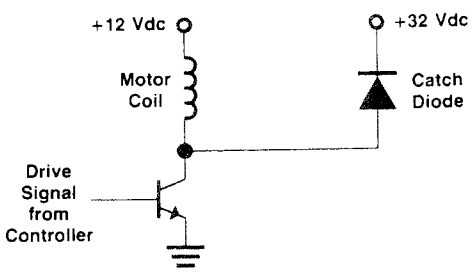


Fig. 3. *Printer motor driver circuit. The energy stored in the motor's inductance is returned to the system power supply through the catch diode. This approach speeds up the stepping rate and conserves power.*

An Efficient Power Supply for the HP-85

As with the rest of the HP-85 system, the efficiency of the power supply must be high. Also, the regulation at low frequencies is important since annoying CRT display modulation results if trace amounts of ac line ripple are present. Switching spikes can also cause ripples in the display.

The solution to these problems is a switching supply synchronized to twice the scan frequency of the CRT (Fig. 1). The heart of this supply is a pulse-width modulator IC that is synchronized by a 32-kHz (twice the CRT horizontal line scan rate) pulse train from the CRT controller. The modulator uses an integrator to provide high loop gain at low frequencies, thus reducing line frequency ripple. The unregulated 32-volt input absorbs the large kickback voltages from the printer drive motors, thus recycling energy.

The output transistor Q1 drives transformer/inductor T1, which acts as a lossless voltage-averaging reactance. The filtered side of T1 provides +12V. During Q1's off time, the flux change in T1 remains constant, so the −12V and +7.5V outputs from T1's secondary winding do not need pass regulation.

The five-volt supply requires higher energy than either the +6V or −12V supplies and cannot be derived from T1 during the off time for Q1. Only energy stored in T1 is available for constant-flux-change regulation. Hence, the five-volt supply uses a separate dc-to-dc converter, which is also synchronized to the CRT display horizontal scan frequency.

The current transformer in the emitter path of Q1 monitors large peaks such as those that occur during printing. High current peaks shut off Q1 in less than two microseconds. This feature prevents problems inherent in driving inductive loads while staying within a safe operating area. If the twelve-volt supply is shorted, the input power will drop, the short acting as a finite current with zero output voltage. This behavior is similar to foldback current limiting in some pass regulator designs.

A +12V overvoltage sensor crowbars the unregulated 32-volt supply if Q1 is shorted. The other supplies do not have an overvoltage failure mode.

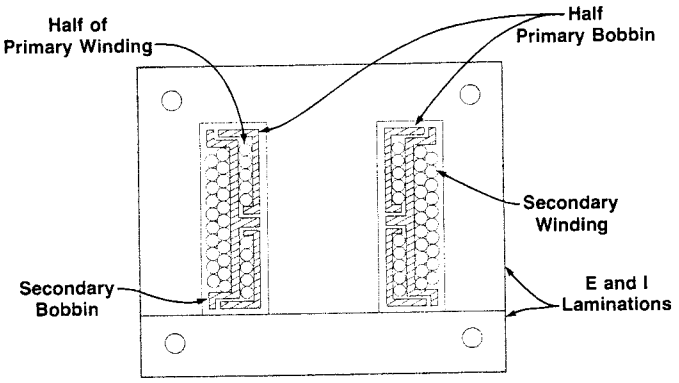


Fig. 2. *Cross-section of the power transformer. This design assures tight coupling between the primary and secondary to reduce unwanted noise and magnetic radiation.*

Line Transformer

The requirements of worldwide safety approval along with the need for minimum electromagnetic radiation to the CRT necessitated development of a special bobbin assembly for the HP-85 power supply transformer.* A cross-section is shown in Fig. 2.

The bobbin has three pieces—two primary windings nested inside the secondary winding to provide tight coupling. The polyester pieces also provide isolation from line voltages as required by safety agencies. The transformer was wound with additional turns to reduce the saturation flux density. Physical placement on the printer subassembly, in the far corner of the HP-85 package away from the CRT, also helped reduce any interference with the display.

Acknowledgments

Floyd Pruitt, Manufacturing Division, provided help on the line transformer. Craig Sanford contributed to the bobbin tooling and Burk Brandt provided help in regard to overall product safety.

*The magnetic radiation from conventional transformers causes waviness in the display because of the lack of primary-to-secondary coupling during core saturation.

-Jim Bausch

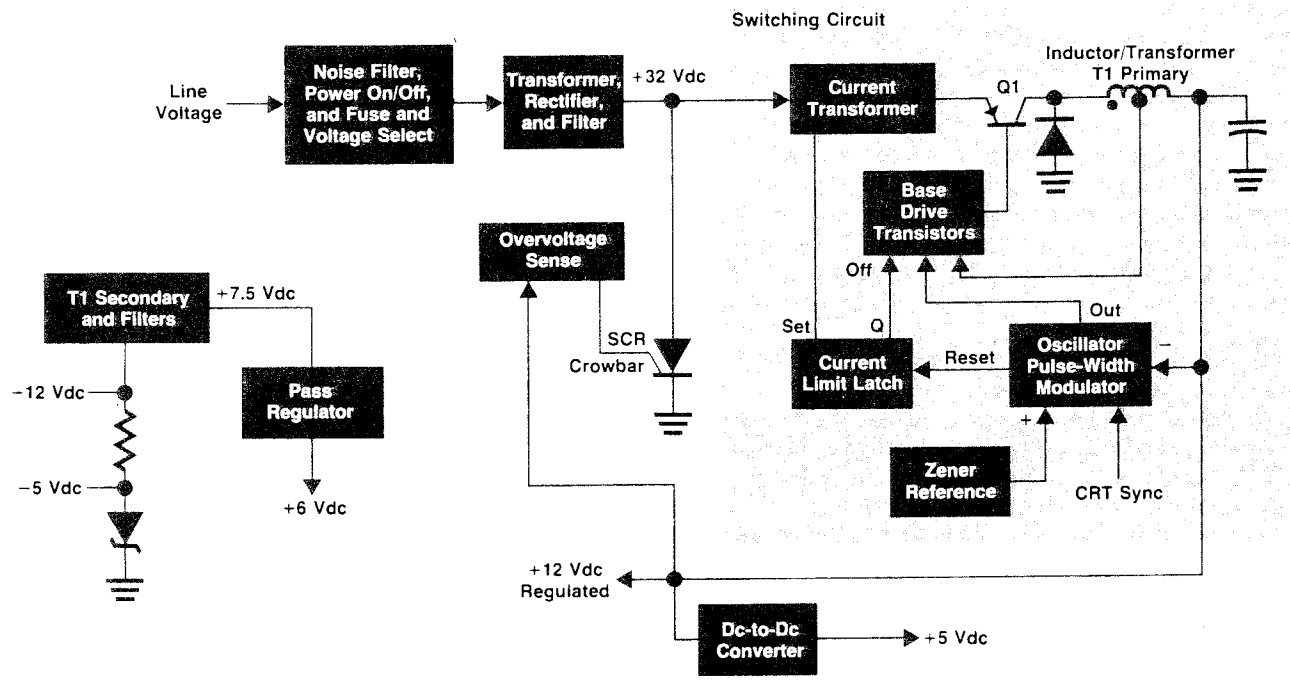


Fig. 1. *HP-85 power supply. The switching rate is at twice the CRT horizontal scan rate and is synchronized with it to minimize effects on display quality.*

To have the resistor heat up quickly and to reduce power consumption, the resistors are formed on a thermal barrier of glass, which is deposited on a ceramic substrate. Making the thermal barrier too thick slows the cool-down time of the resistor, smearing the trailing edge of a formed dot. Accurately controlling this barrier balances both power use and print clarity.

An objective in the HP-85 printer design was to give the user the ability to control the darkness of the print by varying the energy dissipated in the printhead. Through reliability testing, it was discovered that a change of only 10% in the printhead energy dissipation changed the printhead life by a factor of two. Ordinarily, a six-micrometre-thick layer of aluminum oxide is deposited over the resistor to protect it from wear and chemical attack caused by the paper. Inspection of failed printheads indicated that chemical attack was accelerated with increased energy dissipation. It was our observation that the increased thermal cycling created cracks in the brittle protective layer, thus letting the paper chemicals penetrate through the layer to destroy the resistor film. To inhibit crack growth and/or act as a chemical or ion barrier, the idea of depositing an additional film of tantalum-aluminum within the protective overcoat was evaluated. The initial results were substantially improved printhead lifetimes. Fig. 4 is a cross-sectional view of the printhead. Production implementation of this design has yielded two to three-fold increases in printhead life, providing a very acceptable printhead reliability of greater than 85% after five years of use.

Printer Mechanism

The printer mainframe consists of five principal parts that are snapped and screwed together. In addition to locating the mechanical parts, they form a paper bale that holds a generous 120-m (400-ft) roll of 10.8-cm (4.25-in) heat sensitive paper.

The low-cost objective of the printer design precluded use of a high-cost, high-performance motor. Therefore, two identical permanent-magnet step motors are used to advance paper and to move the thermal printhead. Each is a stamped-frame motor with forty-eight steps per revolution.

A toothed belt transfers motion from one of the motors to the paper advance shaft. Narrow, rubber-faced rollers at the ends of the shaft advance the paper by grasping it at the edges. The paper is backed up by two pinch rollers on each side, spaced at nearly 90° to take maximum advantage of the friction of the paper wrap around the rubber roller. A center-pivoted leaf spring at each side provides equal clamping force on each set of pinch rollers. A pair of spring-loaded roll guides is inserted into the ends of the paper roll shaft. These provide a small friction moment to keep the paper under tension, so that the paper unrolls straight.

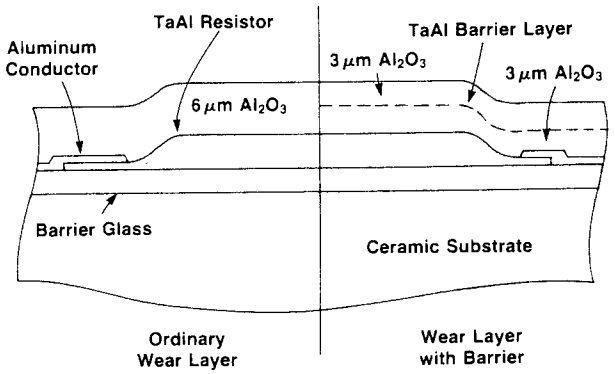


Fig. 4. *Cross-sectional view of a single printhead resistor. The addition of the tantalum aluminum barrier on the right increases printhead lifetime by retarding chemical attack on the resistor.*

The printhead motor is mounted at the rear, transferring rotation forward through use of a shaft extension. A molded pulley on the end of this shaft drives another toothed belt which passes across the front of the printer to an identical idler pulley.

The printhead mechanism consists of a slider and a printhead holder. The slider moves laterally on two identical, parallel, circular shafts held in the mainframe. The printhead holder is pivoted on the slider about an axis parallel to those two shafts. The slider engages the upper shaft with a 60° V-groove, and the lower end of the slider has a small, raised pad that rests against the lower shaft. Easy disassembly for servicing is possible since neither end of the slider encircles its respective shaft. A slot with teeth molded into the forward face of the slider captures the upper run of the toothed belt so it can drive the slider.

The printhead is fixture-bonded to the slider/printhead carrier assembly to assure that the working face (with resistors) is held in the correct position relative to the upper cross shaft previously mentioned. The resistors are located near the right edge of the substrate, and the printhead is fixed with the left edge lifted very slightly from the paper. To prevent the right edge from scraping the paper, this edge and the upper edge are lightly beveled.

A compression spring rotates the printhead holder away from the slider until it contacts the paper, providing the necessary contact force. The reactions from this force hold the V-groove and lower pad of the slider in contact with their respective support shafts. A balancing of spring, friction, and driving forces and the geometry of the assembly assure stability. The spring chosen has a low spring constant and requires considerable initial compression. This prevents major variations in the nominal printhead contact pressure because of a buildup of other tolerances. A molded peg on the slider fits inside one end of the spring, which itself slips inside a molded-in sleeve on the printhead holder. The peg and sleeve almost overlap in the assembled position, countering the lateral instability of the long, slender spring.

Behind the paper lies a full-width platen. The printhead contact force holds cylindrical ribs on the back side of the platen in a flat-bottomed groove in the platen holder. These constrain the platen movement to a rotation about an axis parallel to the upper printhead support shaft so that the platen automatically lines up with the printhead under the applied force. The molded plastic platen is faced with a thin layer of silicone rubber. The whole assembly is overlaid by a flap of plastic-impregnated glass cloth, which is fastened below the platen and serves as a low-friction wear layer. The net effect is a long-wearing, heat-resistant, self-aligning, resilient platen for optimum print quality.

Current is carried to the moving printhead by a flexible

Ronald W. Keil

After earning his BSME degree from San Jose State College in 1963, Ron Keil worked on rocket motor R&D. He returned to school at the University of California at Davis, and earned his MS and PhD degrees in 1969 and 1972, respectively. He then taught at California State Polytechnic University for four years before joining Hewlett-Packard in 1976. He has worked on the HP-85 printer, I/O ports, and plug-in modules and is co-inventor for a patent pending on the HP-85 printhead mechanism. Born in San Francisco, he now lives in Corvallis, Oregon with his wife and two boys. Ron is a registered mechanical engineer in California and a member of SAE. His interests include reading, automotive engineering, building furniture and toys, and serving on the board of the Corvallis Community Club.

Clement C. Lo

Clement Lo has worked on various IC designs since joining Hewlett-Packard in 1975. His responsibilities have included HP-27 product engineering and the printer controller IC for the HP-85. He is a native of Hong Kong and received his BSEE degree from the University of California at Berkeley in 1974 and the MSEE degree from UCLA in 1975. Clement is a computer hobbyist in addition to his other interests—music, gardening, and photography. He and his wife live in Corvallis, Oregon.

cable of copper foil encased in a Kapton™ sheath. This cable makes one 180° bend of substantial radius, rolling and unrolling against a flat panel that forms a crossmember between the printer side plates. By this design, the radius of curvature is held nearly constant, and as the printhead shuttles back-and-forth, the curved or stressed portion of the cable shifts. Thus, no part of the cable is subjected to maximum stress all the time, considerably lengthening the fatigue life of the cable.

Acknowledgments

Printer mechanism contributions came from Tom Hender and his knowledge of step motors, Horst Irmscher's aid in mold design and detail improvements, Bill Schafer's platen mounting idea, and Jeff Forsythe's efforts on the detail design. Printhead development and reliability improvements were made by Ken Trueba with help from Mike Moore.

Enhanced BASIC Language for a Personal Computer

by Nelson A. Mills, Homer C. Russell, and Kent R. Henscheid

HP-85 BASIC is an enhanced implementation of the American National Standard for Minimal BASIC (ANSI X3.60-1978). It is designed to provide the user with a friendly, easy-to-use, and powerful programming tool. The BASIC language interpreter and the HP-85 operating system are implemented entirely in firmware to relieve the user of any responsibility for getting the operating system and/or interpreter loaded and executing.

For the novice user, the HP-85 features an autostart mode of operation. When HP-85 power is turned on, the system will automatically search for a stored program named Autost on the tape cartridge. If a program by this name is found, the HP-85 will load it into memory and begin its execution. This program can then prompt the user for inputs, give instructions, do selected computations, or load other programs selected by the user from a menu.

Several commands are included in the HP-85 BASIC language instruction set to simplify the plotting of data and the presentation of graphical information on the CRT. The CRT display can subsequently be output in hard copy form on the system's internal printer.

Language Enhancements

ANSI Minimal BASIC includes the LET and implied LET statements for variable assignment, the FOR-NEXT statements for looping, the GOTO, GOSUB, computed GOTO, and IF THEN statements for branching, INPUT, DATA, READ, and RESTORE statements for data manipulation, the PRINT statement for output, and several arithmetic operations and functions. ANSI Minimal BASIC contains a total of twenty-one statements and twelve predefined functions.

By contrast, HP-85 BASIC includes 65 statements, 20 commands, and 42 predefined functions. The IF THEN statement has been extended to include the IF THEN ELSE statement, which, when combined with a multistatement line, provides a very useful programming tool. For example,

```
100 IF A> 10 THEN A = A - 10 @ GOTO
    200 ELSE DISP A @ GOTO 300
```

The @ symbol separates individual executable statements for multistatement lines.

HP-85 BASIC allows formatted output via the PRINT USING or DISP USING statements and the IMAGE statement. Their use is illustrated as follows.

```
10 IMAGE 8(" "), "EXCHANGE RATE"/
     /,8(" "),"MARCH 3, 1980"//,
   "$",DC3D.DD," = £",DP3DRDD
20 PRINT USING 10; 1065.43,2198.13
30 STOP
```

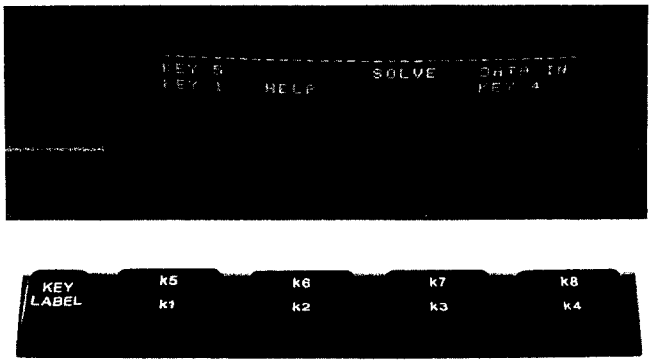


Fig.1. *Up to eight characters can be used for displayed labels for each of the eight user-defined keys in the HP-85 system. A sample display of such labels is shown above.*

Executing these statements results in the output:

EXCHANGE RATE

MARCH 3, 1980

$1,065.43 = £2.198,13

The keyboard includes eight special function keys (k1-k8) which can be programmed using the ON KEY # statement and KEY LABEL. When the KEY LABEL statement appears in a program or when the **KEY LABEL** key is pressed, the labels of the respective keys will be displayed on the bottom three lines of the CRT display. For example, execution of the following statements results in the display shown in Fig. 1.

```
10 ON KEY#1,"KEY 1" GOSUB 100
20 ON KEY#2,"HELP" GOSUB 200
30 ON KEY#4,"KEY 4" GOSUB 300
40 ON KEY#5,"KEY 5" GOSUB 400
50 ON KEY#7,"SOLVE" GOSUB 500
60 ON KEY#8,"DATA IN" GOSUB 600
70 CLEAR
80 KEY LABEL @ GOTO 80 ! WAIT
   FOR KEY PRESS
```

Another important feature of HP-85 BASIC is the ability to solve virtually any size problem using the CHAIN and COMmon statements. CHAIN replaces the current program in memory with a new program that is loaded from mass storage, and retains the current values of variables passed by the COMmon statement.

Debugging of BASIC programs on the HP-85 is simplified by the TRACE, TRACEALL, TRACE VARiable and STEP commands, and the ability to inspect and modify any program variable under keyboard control.

Screen Editor

One of the main design goals in creating the HP-85 Personal Computer was to make it easy to use. The keyboard is tightly integrated with the HP-85 screen editor, giving the user unprecedented editing capabilities in a personal computer. The user has complete cursor control over the entire 64 lines of alphanumeric memory and can operate on anything in this CRT buffer by simply positioning the cursor anywhere within the desired information and pressing the appropriate key.

Inserting characters in a line is made simple by the **INS/RPL** key which switches between the insert and replace character modes. In the insert mode, a double cursor identifies the position in the display where the inserted information will appear. The delete character (**DEL-CHAR**) key removes the character currently underscored by the cursor. Several typing aids and selective screen and line-clear operations add to the strong editing capabilities of the HP-85. Most keys repeat their function when held down.

Modifications made on the CRT with these powerful editing features only take effect when the **ENDLINE** key is pressed. The HP-85 will process up to three lines, or 96 characters, at that time. From the current cursor position, the editor firmware looks up and down the right-hand edge of the CRT until it reaches either a non-displaying carriage return character, the top or bottom line of the current display, or the 96-character limit for editing. With these bounds established, the edited information is then read from the CRT buffer into the system RAM where it can be processed by the operating system.

To correct a problem on some personal computers, the user must retype the corrected problem and execute it:

```
20000 * PI
  62831. 8530718
32000 * PI                    Corrected line
  100530.964915               Corrected result.
-
```

On the HP-85, the user can also use the editor and cursor control keys to make the modification in the original problem:

```
32000 * PI
  62831.8530718
```

which, when executed by pressing the **ENDLINE** key becomes

```
32000 * PI
  100530.964915
-
```

Thus users can easily modify what they see, including scrolling an additional 48 lines onto the screen for modification, and re-executing those lines for quick results. The 64-line alphanumeric memory can also be scrolled for review of previous operations and results.

Mathematical Functions

Historically, BASIC implementations have had a rather limited function set. Fortunately, this situation is changing and the user should expect to be provided not only TAN, for example, but additionally SIN and COS, with all of these functions operable in both degrees and radians. The HP-85 mathematical function set exceeds many other BASIC implementations in quantity, capability, and accuracy. In another sense, most of the HP-85 mathematical functions are now standard fare, having appeared on previous Hewlett-Packard computers. New to HP personal computing products and appearing on the HP-85 is a random

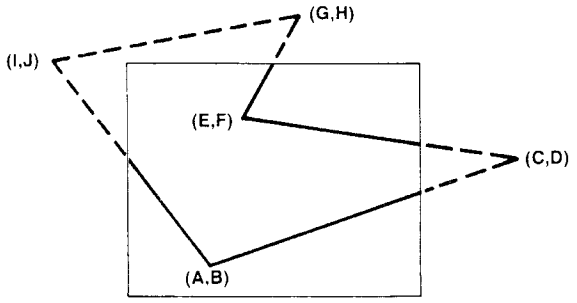


Fig. 2 *Example of clipping performed by the HP-85 system when the plot exceeds the boundary of the screen display. First, a line is drawn from (A,B) toward point (C,D) off-screen. A line drawn now from (C,D) to (E,F) will intersect the edge of the CRT display at the proper point. Thus, the CRT acts as a window on a much larger plotting space. The size of the window is specified by the* SCALE *statement.*

number generator whose implementation has been singled out for discussion at right.

High accuracy has been obtained for the HP-85 mathematical functions by using essentially those algorithms developed first for the HP-67/97 and subsequently improved for the HP-32E, the HP-34C, and the HP-41C as discussed in previous articles.[1-5] These algorithms are adjusted for the larger HP-85 word size to retain accuracy. Real number calculations in the HP-85 are performed internally to fifteen significant decimal digits and rounded to twelve digits for presentation to the user.

For all the algebraic functions (+, −, ×, ÷, and SQR), the error is no bigger than one-half count in the twelfth significant digit (with correct rounding in all situations). For some rational operations such as RMD and MOD there are no errors, regardless of the magnitude of the arguments. For example, 10^{499} MOD 3 = 1 exactly. The transcendental functions are accurate to well within one unit in the twelfth significant digit, except where such specification would be impractical for any machine.

There are just two such cases:

1. When y^x is bigger than 10^{200} or tinier than 10^{-200} (but not 0), its error may exceed one count in the last place but is always smaller than two counts.
2. For trigonometric functions of large radian arguments, TRIG(X) is really TRIG (πX/3.14159265358979) with an additional error rather less than one count in the twelfth significant digit. The use of only fifteen significant digits of π contributes less error than if the given argument X had been changed in its fifteenth significant digit. Despite this error, which is significant only when X is huge, both sides of the trigonometric identity SIN(2X)= 2SIN(X) COS(X) agree to all twelve significant digits when, say, X=52,174 radians. In general, every trigonometric identity that doesn't explicitly involve π will be satisfied to within a rounding error in each trigonometric function that appears in it. Large angles in degrees or grads suffer no such complications (e.g., TAN (2×10^k)= 0.363970234266 for all k=1,2,3,...,499 just as it should).

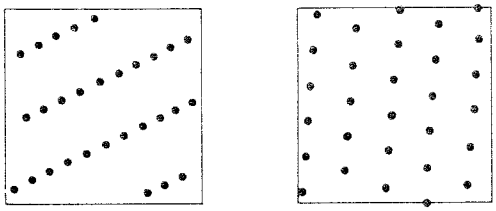
Graphics

The integrated graphics of the HP-85 gives the user sixteen powerful and easy-to-use commands that provide scaling, line generation, labels, axes with interval marks,

Random Number Generation

The random number generator in the HP-85 uses the multiplicative congruential method. Briefly, the method generates the next random integer x_{i+1} by multiplying the current random integer x_i by a constant multiplier α (fixed within the HP-85) and keeping only the right-hand-most fifteen decimal digits: $x_{i+1}=\alpha x_i$ (modulo 10^{15}). The user may supply the starting seed, use a default seed, or call for a random seed using the current clock setting. The routine scales x_{i+1} to get random numbers that lie within the unit interval, and rounds them to twelve significant decimal digits before output. However, the routine continues generating successive random numbers treating α and the x_i's as fifteen-digit integers internally.

The crucial factor for a generator of this kind is the selection of the multiplier α. An inherent characteristic of the multiplicative congruential method is that a plot of consecutive numbers $(x_{i+1}, x_{i+2},\ldots, x_{i+m})$, regarded as points in a space of m dimensions, shows that all of these "random" points lie on a relatively small number of parallel hyperplanes in the m-dimensional unit cube.[1] This presents an appearance similar to that of an orchard (see Fig. 1).



Bad Selection of Multiplier α **Better Selection of Multiplier α**

Fig. 1. (a) Bad selection of multiplier α. (b) Better selection of multiplier α.

Bad selections of the multiplier α force all the m-tuples onto a small number of widely spaced hyperplanes, consequently giving a less uniform distribution of the m-tuples throughout the unit cube. Therefore, the m-tuples must crowd onto the fewer parallel hyperplanes leaving large unoccupied gaps between them.[2] A measure of the widest distance between the hyperplanes is then a measure of the nonuniformity of the generator and characterizes the quality of the selected multiplier α. The determination of this distance is known as the spectral test. Passing this test means selecting a multiplier α that gives rise to minimal hyperplane distance separation according to criteria described in Knuth.[3] Knuth underlines the importance of this test by stating that "not only do all good random-number generators pass it, but also all linear congruential sequences now known to be bad actually fail it! Thus, it is by far the most powerful test known..." The random number generator for the HP-85 was designed to pass the spectral test by developing other computer programs to assist in the proper selection of the multiplier α. The article by Harada[4] outlines the criteria and technique for choosing optimal multipliers for the spectral test.

References

1. G. Marsaglia, "Random Numbers Fall Mainly in the Planes," Proc. N.A.S., Vol. 61, 1968, pp 25-28.
2. W.R. Gosper, "Numerical Experiments with the Spectral Test," Computer Science Department, Stanford University, STAN-CS-75-490, May 1975, pp 1-3.
3. D.E. Knuth, "The Art of Computer Programming," Vol. 2: Semi-Numerical Algorithms, Addison-Wesley, Reading, Mass., 1969.
4. N. Harada, "Optimal Multipliers for the Spectral Test of Random Number Generators," Joho Shori, Journal of the Information Processing Society of Japan, Vol. 15, No. 3, March 1974, pp 180-188.

-Homer Russell

user-generated characters, and hard-copy output. With the SCALE statement the user has complete control of the 256-by-192-dot resolution within a coordinate system specified by the user. The graphics firmware does all conversions from the user's coordinate system to the CRT coordinate system. A user point (X, Y) is mapped into the CRT system (X_c, Y_c) by

$$X_c = \frac{X - X_{min}}{X_{max} - X_{min}} * 255$$

$$Y_c = \frac{Y_{max} - Y}{Y_{max} - Y_{min}} * 191$$

where X_{min}, X_{max}, Y_{min}, Y_{max} are the parameters of the SCALE statement. The factors

$$\frac{255}{X_{max} - X_{min}} \text{ and } \frac{191}{Y_{max} - Y_{min}}$$

are computed by the SCALE statement, so for any one plotting operation, only one subtraction and one multiplication must be performed for each coordinate.

For plots that exceed the boundaries established by the SCALE statement, a complex clipping algorithm handles the intersection of lines to virtual points off the screen, as shown by Fig. 2.

The following program shows the ease of generating high-quality graphics on the HP-85.

```
10 RAD @ GCLEAR
20 SCALE -6*PI,6*PI,-0.1,1
30 XAXIS 0,PI/2 @ YAXIS 0,.1
40 PENUP
50 FOR I=-6*PI TO 6*PI STEP PI/16
60 PLOT I,SIN(I)/I
70 NEXT I
80 MOVE PI, 0.5
90 LABEL "SIN(X)/X"
100 COPY @ STOP
```

Executing this program results in the plot shown in Fig. 3a. The user, after noting the clipping of the bottom part of the graph, can make the following change in line 20.

```
20 SCALE -6*PI,6*PI,-0.4,1
```

which results in the corrected plot shown in Fig. 3b. Point-to-point line generation, character generation with the BPLOT command, labeling commands, and the other statements discussed above give the HP-85 graphics software capability not found in many other personal computers.

Timing, Copying, and Audio Commands

There are three built-in, programmable timers, which may be used to control a program via the ON TIMER # statement. For example, the statement

```
ON TIMER#1,10000 GOSUB 300
```

will start timer number one so that every ten seconds it will interrupt the computer's operation and cause program control to be transferred to line 300.

The contents of the alphanumeric or graphic CRT display may be transferred to the internal printer by executing the COPY statement or by pressing the **COPY** key on the keyboard.

Finally, the HP-85 has an internal programmable beeper that is controlled by specifying the frequency and duration of the desired tone via the BEEP statement. This statement can be used with other HP-85 BASIC commands in a program to enable the user to compose music, print out the musical score, and play the tune, using the beeper.

Program Internal Formats

HP-85 BASIC programs are executed by an interpreter that is part of the firmware operating system. However, the code that is interpreted is vastly different from the BASIC commands as they were originally entered. As the statements are entered, they are compiled to a form of RPN (reverse Polish notation), which can be interpreted more efficiently than the BASIC source statements. As part of the compilation process, all BASIC reserved words are converted to single-byte tokens. This makes the internal form of the code somewhat more compact than the original form, and also makes interpretation easier and faster. The single-byte tokens are actually ordinals for the runtime table, the parse table, and the ASCII table. Thus, token 142 is associated with the 142nd entry in the runtime table, the parse table, and the ASCII table. Following is an example of the internal form of the HP-85 BASIC assignment statement.

The statement line

```
10 LET A = B + SIN (C*D)
```

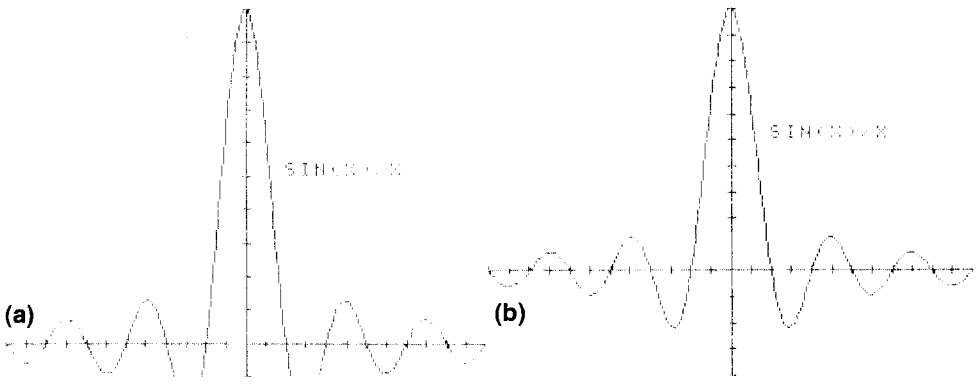


Fig. 3. *(a) An initial plot shows that the graph of a waveform is clipped at the bottom. (b) By modifying one parameter in the SCALE statement, the graph now can be easily relocated within the display boundaries.*

is converted to single-byte tokens as listed below.

20	BCD line #10
0	
22	# bytes in the line
142	Token for LET
21	Token for fetch variable address
40	ASCII blank
101	ASCII A
1	Token for fetch variable
40	ASCII blank
102	ASCII B
1	Token for fetch variable
40	ASCII blank
103	ASCII C
1	Token for fetch variable
40	ASCII blank
104	ASCII D
52	Token for *
330	Token for SIN
53	Token for +
10	Token for store numeric variable
16	End of line token

Preallocation

One of the things that has traditionally made interpreters slow is that they maintain a table of variables, which must be searched at runtime for each variable reference. In the HP-85 we have attempted to solve this problem by preallocation of all variable references. During the allocation process, the variable names that occur in the internal RPN form of the program are replaced by the relative addresses of the variables. Thus, at runtime, the interpreter has only to read an address and add it to the base address of the program to determine the absolute location of the variable being referenced. For traditional interpreters the time required to access any variable is dependent upon its position within the variable table. On the HP-85 all variables may be accessed in exactly the same amount of time.

User-defined functions are like variables in that they have names that look like variable names, and they have associated with them a value, much like a variable. Because of these similarities and the obvious advantages in runtime execution, user-defined functions are also preallocated.

In traditional interpreters, it is also necessary to search for all line number references. Thus execution speed is affected by how far the referenced line is from the beginning of the program or from the current line, depending on the implementation. In the HP-85 all line number references are replaced by the relative address of the referenced line during the allocation process, thus eliminating the need to search for referenced lines at runtime.

Output Control

Flexible vectored control of output is provided in the HP-85 operating system by an output control routine that acts like a police officer directing traffic. This routine is accessed by the user via the PRINTER IS and CRT IS statements. Output information that normally goes to the CRT (output device 1) can be redirected to the internal printer

Fast Integer Processing

The HP-85 implementation of BASIC provides INTEGER, SHORT, and REAL variable type declarations (undeclared variables are considered to be REAL) with this feature—many calculations are transparent to the user and proceed internally faster than might be expected when the values involved are integers, even though INTEGER type may or may not have been specified.

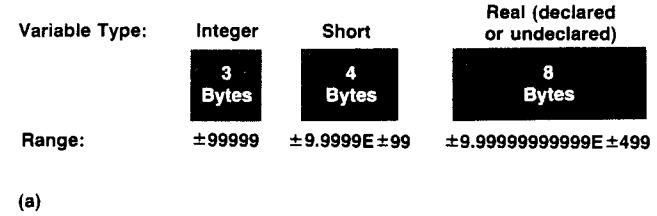
Let's see how and when this occurs. Memory space is conserved by allocating for each variable no more space than is needed to represent the range of values over which that type of variable can run (see Fig. 1a). In each of these cases, the variable is initially tagged using one of the bytes to flag it as "null data" (a warning is issued to the user if processing is attempted using null data). Now suppose, for example, a program initializes a variable I=0 so it is no longer null and subsequently performs the calculation I=I+1, assuming no prior type declaration for I. By default, I is assumed REAL. It is easiest for the user to ignore type declaration when storage space is not at a premium. The penalty for this in usual BASIC implementations is to force slower floating point calculations, even though only integer values may be involved. The HP-85 implementation avoids this penalty, when it can, by using an additional special format—tagged integers.

When a value is assigned to a REAL variable, it is first examined to see if it is actually an integer (up to three bytes). If so, it is stored as shown in Fig. 1b and the subsequent calculation I=I+1 will proceed much faster since only small integers are involved. Should the integer become too large (more then five digits) or should some other portion of the program force I to a wider REAL format, I will remain REAL and subsequent calculations of I=I+1 will proceed accurately, but at a slower pace.

The fetch routine presents eight bytes (a REAL or tagged integer) to each math routine. Routines not interested specifically in integers, such as SIN and COS, call a conversion routine which checks for the tag byte, and, when necessary, converts the integer to the normalized REAL format. The add, subtract, and multiply routines act differently. If both arguments are tagged integers, fast integer routines are employed, providing fast counting in FOR-NEXT loops and other instances as mentioned.

When processing of a math routine is completed, control is relinquished to the system. If the result is to be stored in memory, the recipient variable is checked and the eight-byte result is converted, if necessary, to the appropriate one of the formats shown in Fig. 1 before storage. Should the answer exceed the range for that variable type, a default value is stored and a range warning is issued.

-Homer Russell



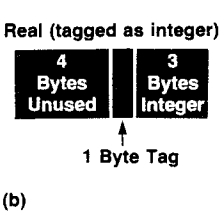


Fig. 1. *(a) Each variable is allocated space in the memory according to its type specification. (b) If a REAL variable is perceived to be actually an integer of up to three bytes, it is stored in memory as shown and processed internally as an integer.*

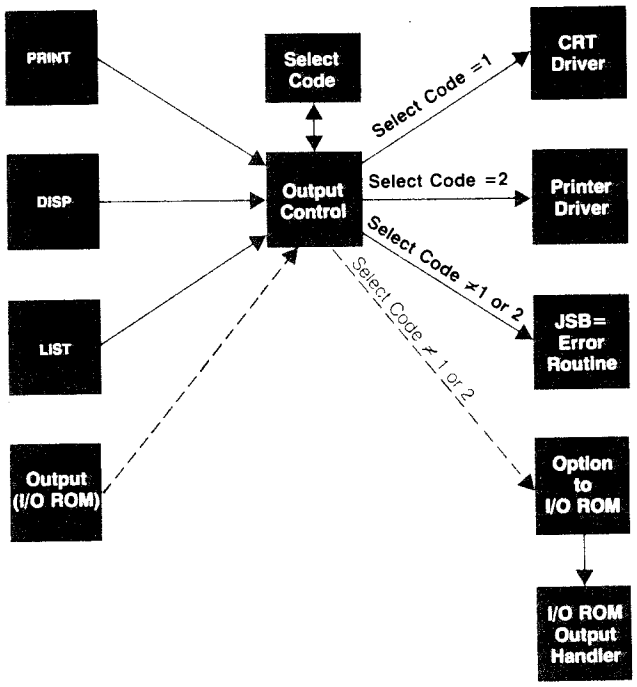


Fig. 4. *An output control routine directs the output from PRINT, DISP, and LIST statements to the selected peripheral. If the peripheral is not present, the routine calls an appropriate error routine. When additional peripherals are added to the HP-85 system the I/O ROM supplies an additional OUTPUT command and redirects the normal error subroutine call from the output control routine to the appropriate subroutine in the I/O ROM.*

(output device 2) by the statement CRT IS 2. Similarly, output information normally going to the internal printer can be redirected to the CRT by a PRINTER IS 1 statement. Normal CRT output information comes from DISP, LIST, and CAT statements and ERROR messages. Print output comes from PRINT, PLIST, and TRACE statements. Output device specifications other than 1 or 2 normally result in an error message. All output-producing statements set a select code, which the output control routine uses to direct the output from the producer to the target device (see Fig. 4). The operating system uses a select code of 0 as a flag for external I/O to increase throughput speed of multiple output calls to the output control routine, such as LISTings or OUTPUT statements with multiple parameters in the output list.

System Extensions

Option locations or "hooks" in the operating system allow extension of the HP-85 capabilities. An option location is typically several RAM locations that the operating system accesses with a subroutine call or a single byte in RAM containing a flag with special meaning to the system. In the case of a subroutine call option, the RAM option location would be initialized to a subroutine return or an error message subroutine call. A plug-in ROM or binary program can "hook" into the operating system and take over some process in the system by inserting a subroutine call to itself in place of the default option location contents. For example, the output control routine discussed earlier attempts to send output for a select code other than 1 or 2 to one of these locations. A plug-in ROM that handles the output to a peripheral device replaces the normal error call at the respective option location with a subroutine call pointing to its own output handling routine (see Fig. 4). Other locations in the operating system provide interception of the keyboard processing routine, the system parser, the main idle loop, and external I/O interrupts. Special flags in the RAM option locations permit image and interrupt process extensions.

The HP-85 option locations thus provide an easy means to expand the capabilities and extend the power of the basic computer.

Mass Storage

The 98200A tape cartridge provides a digital-quality storage medium for data and program storage and retrieval. A directory of file information at the front of the tape immediately provides the location and length of any of up to 42 files on the tape.[6] A high-speed search technique gives the user fast access to files, with the six-character file name providing a friendly reference not commonly found in personal computers. This directory can be readily displayed using the CATalog command. Four levels of file security provide protection of sensitive or important information on the tape. The user can prevent duplication and listing of program files and access to data files that contain private information. A soft write-protect security prohibits accidental replacement of valuable files.

The data files can be structured by the user to be any number of records, each of a specified size up to 32767 bytes long, limited only by the mass storage space available (210K bytes for each tape cartridge). The data files can be accessed both serially and randomly, giving the user complete control over retrieval techniques and access speed for data. The records within a file are treated as logical length records and can be addressed directly by specifying the numbers of their positions within the file. The firmware transparently handles the physical record divisions for the user.

Language Extensions

The BASIC language as implemented on the HP-85 can be extended by the addition of either plug-in ROMs or binary programs. HP-85 memory consists of 32K bytes of ROM and 16K bytes of RAM. The last 8K bytes of the ROM is bank-selectable. A plug-in module allows expansion of the RAM to 32K bytes.

Plug-in ROMs map into the bank-selectable portion of system ROM, and binary programs load into the RAM. There may be up to six plug-in ROMs and one binary program in the system at any one time. Binary programs are loaded by the LOADBIN command and stored by the STORE-BIN command, and reside in the highest memory locations of the RAM. They have a somewhat rigid structure in that all relocatable addresses are contained in a block at the beginning of the program and are relocated by the LOADBIN command. The remainder of the binary program must be written as "run anywhere code."

The structures for ROMs and binary programs are similar in that they require the same table structure at the beginning. The first two bytes of each ROM contain the ROM number and its complement. The first two bytes of a binary program contain the address at which the binary is loaded.

Nelson A. Mills

Joining HP in 1976, Nelson Mills worked on the HP-85 operating system and interpreter. He is now the project manager for high-end firmware at the Corvallis Division. After receiving a BS in mathematics from Albion College, Michigan in 1961, Nelson spent five years in the U.S. Navy and then worked for ten years as a systems programmer. He and his family—wife and two children—live in Corvallis, Oregon. Outside of working hours Nelson enjoys photography, hiking, camping, and coaching basketball and AYSO soccer.

Kent R. Henscheid

After earning a BS in Mathematical Sciences from Stanford University in 1973, Kent Henscheid joined HP as an applications engineer. In late 1975 he went back to Stanford and received the MSEE degree in 1976. Since then he has worked on the firmware for the HP-85 and recently has become involved with product marketing. Kent is a co-inventor for a patent related to the HP-22 pocket calculator and is a native of Pocatello, Idaho. He and his wife live in Corvallis, Oregon, have one son, and are expecting a new addition to the family. Outside of working hours, Kent enjoys music, hunting, fishing, camping and skiing.

Homer C. Russell

At HP since 1969, Homer Russell initially worked on marketing applications and later became a marketing product support manager. Subsequently, he has worked on HP-85 mathematics algorithm development and implementation and now is a project leader for the HP-85 Matrix ROM. Before joining HP, he worked on orbital analysis for the Apollo program and as a computer systems engineer. Homer is a co-inventor for a patent on the 9820 Computer. Born in Phelps, Kentucky, he earned a BS in physics from the University of Texas (El Paso) in 1961 and an MA in mathematics from the University of California at Berkeley in 1966. Homer is married with two children, a son and a daughter, and lives in Corvallis, Oregon. When he isn't home with his family working out in their exercise room or relaxing in their sauna and hot tub, he likes to play racquetball, look for Indian relics, or read about mathematics, investment analysis, and nutritional studies.

The next ten bytes of both ROMs and binary programs contain, in order, the runtime table address, the ASCII table address, the parse table address, the error message table address, and the address of the initialization routine.

For binary programs, this block of table addresses is followed by the runtime table and the parse table, since they also contain addresses that must be relocated.

Acknowledgments

We would like to thank George Fichter who was project manager in charge of software development for the HP-85, and who did much of the design and implementation of the operating system and interpreter. Wan Cheng Chan developed the self-test function and the diagnostic ROM. Dennis Harms and Professor W. Kahan contributed to the implementation of the mathematical functions. Bill Kemper, Dan Harrington, and Peter Bock were responsible for the development of the applications software, and Constance Jackson wrote the HP-85 Owners Manual.

References

1. D.W. Harms, "The New Accuracy: Making $2^3=8$", Hewlett-Packard Journal, November 1976.
2. W.E. Egbert, "Personal Calculator Algorithms I: Square Roots", Hewlett-Packard Journal, May 1977.
3. W.E. Egbert, "Personal Calculator Algorithms II: Trigonometric Functions". Hewlett-Packard Journal, June 1977.
4. W.E. Egbert, "Personal Calculator Algorithms III: Inverse Trigonometric Functions", Hewlett-Packard Journal, November 1977.
5. W.E. Egbert, "Personal Calculator Algorithms IV: Logarithmic Functions", Hewlett-Packard Journal, April 1978.
6. W.A. Hanna, "Mass Storage Management—A Unified Approach," Hewlett-Packard Journal, June 1980.

Address Correction Requested
Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

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

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, 1501 Page Mill Road, Palo Alto, California 94304 U.S.A. Allow 60 days.

Takagi and Minoru Niizaki designed the voltage sources, and Fumiroh Tsuruda and Hisao Yoshino designed the digital section. Mechanical design was by Yoshimasa Shibata and industrial design by Kazunori Shibata. Yoshio Sato designed the accessories. We would also like to thank Takuo Banno, who gave much useful advice on design and evaluation of the prototypes, and the many other people who made significant contributions to the project.

Hitoshi Noguchi

Hitoshi Noguchi graduated from Akita University in 1961 and joined Yokogawa Electric Works that same year, working as and R and D engineer on signal generator development. He transferred to Yokogawa-Hewlett-Packard in 1964 where he worked on the 4260A Universal Bridge, the 4270A Capacitance Bridge, the 4271A LCR Meter, among others, before becoming project leader for the 4140A. Outside of working hours, Hitoshi likes to go hiking and cycling, or listening to classical music.

ABBREVIATED SPECIFICATIONS

HP Model 4140A pA Meter/DC Voltage Source

MEASUREMENT FUNCTIONS: I, I-V and C-V.
I: Independent picoammeter and programmable voltage source.
I-V: I-V characteristic measurements.
C-V: Quasi-static C-V characteristic measurement.

VOLTAGE SOURCES: V_A and V_B:

Function	V_A	V_B
I		
I-V		
C-V		

VOLTAGE SWEEP: Auto or manual (pause).

DISPLAYS:
CURRENT: 3½ digits with 2-digit annunciator.
VOLTAGE: 3½ digits.

Current Measurements

RANGE: $\pm 1.000 \times 10^{-12}$ A to $\pm 1.000 \times 10^{-2}$ A full scale in 11 ranges, auto or manual ranging, 90% overrange.

ACCURACY/INTEGRATION TIME:

Range	Accuracy ±(% of rdg. + counts)	Integration Time (ms) Short	Medium	Long
10^{-2}-10^{-9}	0.5 + 2	20	80	320
10^{-10}	2 + 2			
10^{-11}	5 + 3	80	320	1280
10^{-12}	5 + 8	160	640	2560

VOLTAGE BURDEN: ≤10μV at full scale.
ZERO OFFSET RANGE: 0 to $\pm 100 \times 10^{-15}$ A.
TRIGGER (Output I Data): INT, EXT and HOLD MAN.
HIGH-SPEED I DATA OUTPUT: Available with HP-IB option. Maximum rate: 2.5 ms intervals.

Capacitance-Voltage (C-V) Measurement

RANGE: 0.0 pF to 1900 pF, auto-ranging.
ZERO OFFSET RANGE: 0 to 100 pF.
%C RANGE: 0.0% to 199.9% (Capacitance change in device under test is displayed as a percent of the set value of the oxide capacitance; Cox = 100%).

DC Voltage Sources

RANGES (V_A AND V_B): 0 to ±100.0 V.
MAXIMUM CURRENT: 10 mA, both sources.
VOLTAGE SWEEP: Auto and manual (pause), up/down step in manual (pause) mode. Sweep abort enables reset.
PARAMETER SETTING RANGES:
START/STOP V: 0 to ±10.00V, 0.01V steps; 0 to ±100.0V, 0.1V steps.
STEP V: 0 to ±10.00V, 0.01V steps; 0 to ±100.0V, 0.1V steps.
HOLD TIME: 0 to 199.9 s, 0.1-s steps; 0 to 1999 s, 1-s steps.
STEP DELAY TIME: 0 to 10.00 s, 0.01-s steps; 0 to 100.0 s, 0.1-s steps.
dV/dt (ramp rate): 0.001 V/s to 1.000 V/s, 0.001-V/s steps.
CURRENT LIMITING: 100 μA, 1 mA, and 10 mA, ±10%(V_A and V_B).

General

OPERATING TEMPERATURE: 0°C to 40°C.
RELATIVE HUMIDITY: ≤70% at 40°C.
POWER: 100, 120, 220V, ±10%; 240V + 5% −10%; 48-66 Hz, 135 V_A maximum with any option.
DIMENSIONS: 426 mm W × 177 mm H × 498 mm D (16.5 × 7 × 19.6 in).
WEIGHT: 14.2 kg (31.2 lb).
ACCESSORY FURNISHED: 16053A Test Leads. Triaxial cable, two each BNC-BNC cables and one connection plate.
OPTIONS: 001 Analog Output (I, C and V) with pushbutton scaling.
101 HP-IB Interface.
ACCESSORIES AVAILABLE:
16053A Test leads.
16054A Connection selector.
16055A Test fixture, general-purpose.
16056A Current divider (10:1).
PRICES IN U.S.A.: 4140A, $7360; Opt 001, $325; Opt 101, $220; 16053A, $320; 16054A, $275; 16055A, $1250; 16056A, $140.
MANUFACTURING DIVISION: YOKOGAWA-HEWLETT-PACKARD LTD.
9-1, Takakura-cho, Hachioji-shi
Tokyo, Japan, 192

Personal Calculator Has Key to Solve Any Equation f(x) = 0

The HP-34C is the first handheld calculator to have a built-in numerical equation solver. That's why one of its keys is labeled SOLVE.

by William M. Kahan

BUILT INTO HEWLETT-PACKARD'S new handheld calculator, the HP-34C, is an automatic numerical equation solver. It is invoked by pressing the **SOLVE** key (see Fig. 1). For an illustration of how it finds a root x of an equation f(x) = 0 take the function

$$f(x) \equiv e^x - C_1x - C_2$$

with constants C_1 and C_2. Equations f(x) = 0 involving functions like this one have to be solved in connection with certain transistor circuits, black-body radiation, and stability margins of delay-differential equations. If the equation f(x) = 0 has a real root x three steps will find it:

Step 1. Program f(x) into the calculator under, say, label **A** (see Fig. 2).

Step 2. Enter one or two guesses at the desired root:
(first guess) **ENTER** (second guess if any)
Any x will do as a guess provided f(x) is defined at that value of x, but the closer a guess falls to a desired root the sooner that root will be found.

Step 3. Press **SOLVE A** and wait a little while to see what turns up.

Figs. 3a-3d show what turns up for a typical assortment of constants C_1 and C_2 and first guesses.

When a root is found it is displayed. But is it correct? When no root exists, or when **SOLVE** can't find one, ERROR 6 is displayed. But how does the calculator know when to abandon its search? Why does it not search forever? And if it fails to find a root, what should be done next? These questions and some others are addressed in the sections that follow.

What does SOLVE Do, and When Does It Work?

Neither **SOLVE** nor any other numerical equation solver can understand the program that defines f(x). Instead, equation solvers blindly execute that program repeatedly. Successive arguments x supplied to the f(x) program by **SOLVE** are successive guesses at the desired root, starting with the user's guess(es). If all goes well, successive guesses will get closer to the desired root until, ideally, f(x)=0 at the last guess x, which must then be the root. **SOLVE** is distinguished from other equation solvers by its guessing strategy, a relatively simple procedure that will surely find a root, provided one exists, in an astonishingly wide range of circumstances. The three simplest circumstances are the ones that predominate in practice:

1. f(x) is strictly monotonic, regardless of initial guesses, or
2. ±f(x) is strictly convex, regardless of initial guesses, or
3. Initial guesses x and y straddle an odd number of roots, i.e., f(x) and f(y) have opposite signs, regardless of the shape of the graph of f.

In these cases **SOLVE** always finds a root of f(x)=0 if a root exists.

About as often as not, **SOLVE** must be declared to have found a root even though f(x) never vanishes. For example, take the function:

$$g(x) \equiv x+2\cdot(x-5)$$

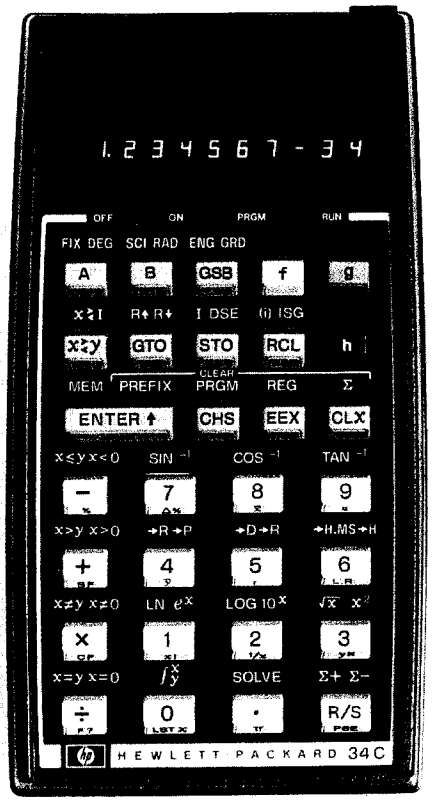


Fig. 1. *The HP-34C, a new handheld programmable calculator, has two keys that are new to handheld calculators– ∫(integrate) and* SOLVE *SOLVE, a numerical equation solver, is described in this article.*

PRGM▮▯RUN	Switch to Program Mode
CL PRGM	Clear Program Memory
LBL A	x Is in the X Register
e^x	e^x
LST x	Get x Back
RCL 1	C_1
×	C_1x
−	$e^x - C_1x$
RCL 2	C_2
−	$f(x) = e^x - C_1x - C_2$
RTN	Return f(x) in the X register
PRGM▯▮RUN	Switch to Run Mode
... C_1 ...	
STO 1	Store C_1 in Register 1
... C_2 ...	
STO 2	Store C_2 in Register 2

Fig. 2. *This is an HP-34C program for the function* $f(x) = e^x - C_1x - C_2$. *It replaces x by f(x) in the HP-34C's X register (display). It is labeled A, but labels B, 0, 1, 2, or 3 would serve as well.*

Of course g(x) = 3x−10, and when calculated as prescribed above (don't omit the parentheses!) it is calculated exactly (without roundoff) throughout $1 \leq x \leq 6.666666666$. Consequently, the calculated value of g(x) cannot vanish because the obvious candidate x=10/3=3.333... cannot be supplied as an argument on an ordinary calculator. **SOLVE** does the sensible thing when asked to solve g(x)=0; it delivers final guesses 3.333333333 and 3.333333334 in the X and Y registers in a few seconds. In general, when **SOLVE** finds a root of f(x)=0 it returns two final guesses x and y in the X and Y registers respectively; either x=y and f(x)=0, or else x and y differ in their last (10th) significant decimal digit and f(x) and f(y) have opposite signs. In both cases the Z register will contain f(x).

On the other hand, **SOLVE** may fail to find a place where f(x) vanishes or changes sign, possibly because no such place exists. Rather than search forever, the calculator will stop where |f(x)| appears to be stationary, near either a local positive minimum of |f(x)| as illustrated in Fig. 3d or where f(x) appears to be constant. Then the calculator displays ERROR 6 while holding a value x in the X register and f(x) in the Z register for which $f(y)/f(x) \geq 1$ at every other guess y that was tried, usually at least four guesses on each side of x. (One of those guesses is in the Y register.) When this happens the calculator user can explore the behavior of f(x) in the neighborhood of x, possibly by pressing **SOLVE** again, to see whether |f| really is minimal near x, as it is in Fig. 3d, or whether the calculator has been misled by unlucky guesses. More about this later.

So **SOLVE** is not foolproof. Neither is any other equation solver, as explained on page 23.

How Does SOLVE Compare with Other Root-Finders?

Program libraries for large and small computers and calculators usually contain root-finding programs, but none of them works over so wide a range of problems or so conveniently as does the HP-34C's **SOLVE** key. Other root-finders are hampered by at least some of the following limitations:

1. They insist upon *two* initial guesses that straddle an odd number of roots. **SOLVE** accepts any guess or two and does what it can to find a root nearby, if possible, or else farther away.
2. They may have to be told in advance how long they are permitted to search lest they search forever. Consequently their search permit may expire after a long search, but just moments before they would have found a root. **SOLVE** knows when to quit; it can't go on forever, but it can go on for a long time (e.g., when f(x)=1/x).
3. They may require that you prescribe a tolerance and then oblige you to accept as a root any estimate closer than that tolerance to some previous estimate, even if both estimates are silly. **SOLVE** will claim to have found a root x only when either f(x)=0 or $f(x)\cdot f(y)<0$ for some y differing from x only in their last (10th) significant decimal digit.
4. They may claim that no root exists when they should admit that no root was found. **SOLVE** will not abandon its search unless it stumbles into a local minimum of |f|, namely an argument x for which $f(y)/f(x) \geq 1$ at all other (usually at least nine) sampled arguments y on both sides of x.
5. They may deny to the program that calculates f(x) certain of the calculator's resources; for instance
   "begin with no label other than A' "
   "do not use storage registers 0 through 8"
   "do not use certain operations like CLR or =".
   SOLVE allows the f(x) program to use everything in the calculator except the **SOLVE** key. Moreover, **SOLVE** may be invoked from another program just like any other key on the calculator; and f(x) can use the HP-34C's powerful $\int_y^x$ key.

A lot of thought has gone into making **SOLVE** conform to Albert Einstein's dictum: "As simple as possible, but no simpler."

How Does SOLVE Work?

The **SOLVE** key's microprogram uses very little of the HP-34C's resources. Reserved for **SOLVE**'s exclusive use are just five memory registers for data and a handful of other bits. Those five memory registers hold three sample arguments α, β, and γ and two previously calculated sample values $f(\alpha)$ and $f(\beta)$ while the user's f(x) program is calculating f(x) from the argument $x = \gamma$, which it found in the stack. How does **SOLVE** choose that argument γ?

Suppose α and β both lie close to a root $x = \zeta$ of the equation f(x)=0. Then a secant (straight line) that cuts the graph of f at the points $[x=\alpha\ ,\ y=f(\alpha)]$ and $[x=\beta\ ,\ y=f(\beta)]$ must cut the x-axis at a point $[x=\gamma\ ,\ y=0]$ given by

$$\gamma = \beta - (\beta-\alpha)\cdot f(\beta)/(f(\beta)-f(\alpha)) \qquad (1)$$

Provided the graph of f is smooth and provided ζ is a simple root, i.e., $f(\zeta) = 0 \neq f'(\zeta)$, then as Fig. 4 suggests, γ must approximate ζ much more closely than do α and β. In fact the new error $\gamma-\zeta$ can be expressed as

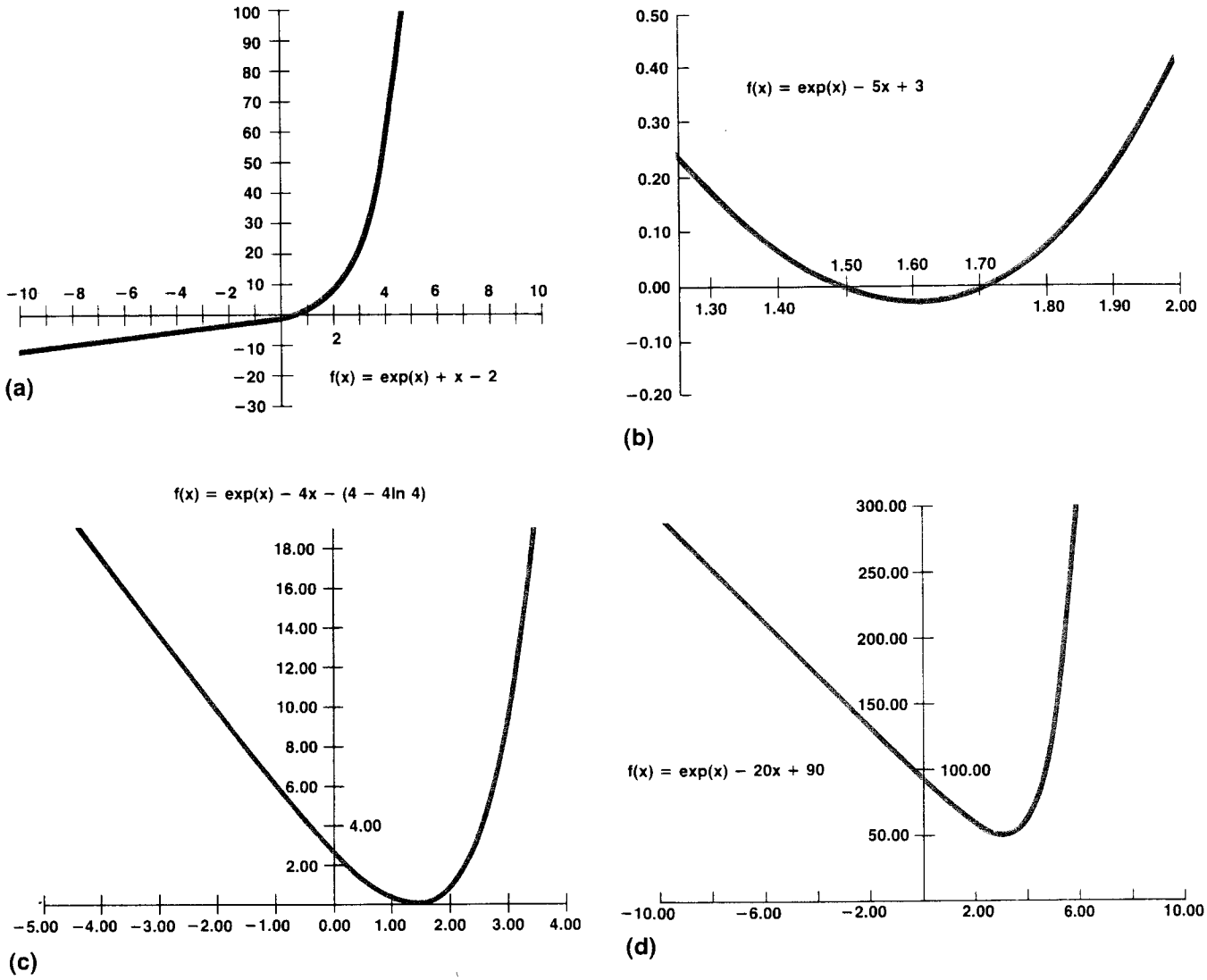


Fig. 3. *Examples of* SOLVE *results for different values of C_1 and C_2 and different first guesses for the root x in the program of Fig. 2. (a) If the first guess is −99 the root x = 0.442854401 is found in 25 seconds. The graph of f(x) on the negative-x side is relatively straight, so* SOLVE *works quickly. If the first guess is 99 the root is found in 190 seconds.* SOLVE *takes longer to get around a sharp bend. (b) With first guesses 0 and 2 the root 1.468829255 is found in 30 seconds. With first guesses 2 and 4 the root x = 1.74375199 is found in 20 seconds. Many root finders have trouble finding nearby roots. (c) With first guesses 0 and 2 the double root 1.386277368 is found in 50 seconds. Many root finders cannot find a double root at all. (d) Since no root exists,* SOLVE *displays* ERROR 6. *With first guesses of 0 and 10,* SOLVE *displays* ERROR 6 *in 25 seconds. After the error is cleared* SOLVE *displays 2.32677...., which approximates the place x = 2.99573.... where f(x) takes its minimum value 50.085.....*

$$\gamma - \zeta = K \cdot (\alpha - \zeta) \cdot (\beta - \zeta)$$

where K is complicated but very nearly constant when α and β both lie close enough to ζ. Consequently the secant formula, equation 1, improves good approximations to ζ dramatically, and it may be iterated (repeated): after $f(\gamma)$ has been calculated α and $f(\alpha)$ may be discarded and a new and better guess δ calculated from a formula just like equation 1:

$$\delta = \gamma - (\gamma - \beta) \cdot f(\gamma) / (f(\gamma) - f(\beta)) \qquad (2)$$

This process repeated constitutes the secant iteration and is the foundation underlying the operation of the **SOLVE** key.

A lot could be said about the secant iteration's ultimately rapid convergence, but for two reasons the theory hardly ever matters. First, the theory shows how strongly the secant formula (equation 1) improves good estimates of a root without explaining how to find them, even though the search for these estimates generally consumes far more time than their improvement. Second, after good estimates have been found, the secant iteration usually improves them so quickly that, after half a dozen iterations or so, the tiny calculated values of f(x) fall into the realm of rounding error noise. Subsequent applications of equation 1 are confounded by *relatively* inaccurate values $f(\alpha)$ and $f(\beta)$ that

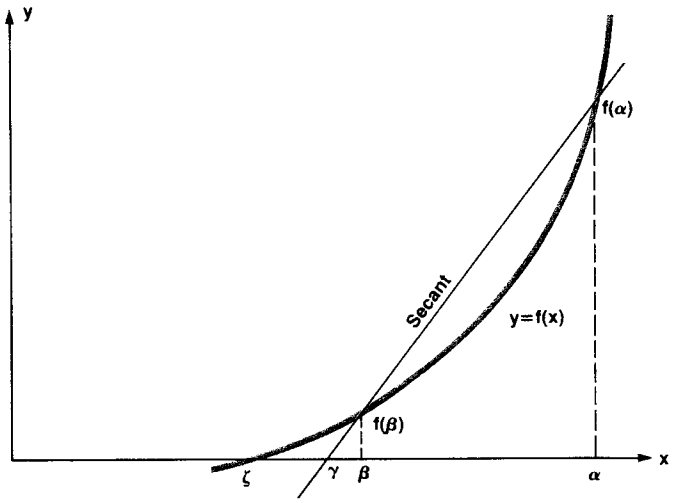


Fig. 4. *Given guesses α and β with corresponding function values f(α) and f(β) the secant iteration produces a new guess γ by the formula* $\gamma = \beta - (\beta-\alpha)\cdot f(\beta)/(f(\beta)-f(\alpha))$.

produce a spurious value for the quotient $f(\beta)/(f(\beta)-f(\alpha))$. For these reasons the secant iteration is capable of dithering interminably (or until the calculator's battery runs down). Figs. 5a-5b show examples where the secant iteration cycles endlessly through estimates $\alpha, \beta, \gamma, \delta, \alpha, \beta, \gamma, \delta, \ldots$

Therefore, the secant iteration must be amended before it can serve the **SOLVE** key satisfactorily.

SOLVE cannot dither as shown in Fig. 5a because, having discovered two samples of f(x) with opposite signs, it constrains each successive new guess to lie strictly between every two previous guesses at which f(x) took opposite signs, thereby forcing successive guesses to converge to a place where f vanishes or reverses sign. That constraint is accomplished by modifying equation 2 slightly to bend the

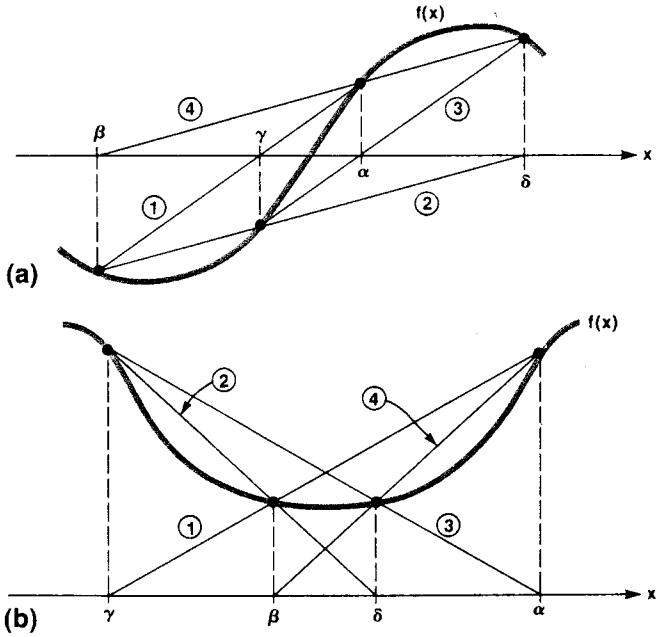


Fig. 5. *Examples of how the secant iteration can cycle endlessly through the values α, β, γ, δ. (1) α, β→γ (2) β, γ→δ (3) γ, δ→α (4) δ, α→β and so forth.*

Why Is Equation Solving Provably Impossible?

"The merely Difficult, we do immediately; the Impossible will take slightly longer." Old British naval maxim.

What makes equation solving merely difficult is the proper calculation of f(x) when the equation f(x) = 0 has to be solved. Sometimes the calculated values of f(x) can simultaneously be correct and yet utterly misleading. For example, let $g(x) \equiv x+2\cdot(x-5)$; this is the function whose calculated values change sign but never vanish. Next let the constant c be the calculated value of $(g(10/3))^2$; this amounts to $c = 10^{-18}$ on an HP handheld calculator, but another calculator may get some other positive value. Finally, let $f(x) \equiv 1-2\exp(-g^2(x)/c^2)$. The graph of f crosses the x-axis despite the fact that the correctly rounded value calculated for f(x) is always 1. None of the arguments x for which f(x) differs significantly from 1 can be keyed into the calculator, so it has no way to discover that f(x) vanishes twice very near 10/3, namely at

$$x = 10/3 \pm c\sqrt{\ln 2}/3$$

No numerical equation solver could discover those roots.

Worse, perhaps, than roots that can't be found are roots that aren't roots. Here is an example where the calculator cannot know whether it has solved f(x) = 0 or f(x) = ∞. Consider the two functions

$$f(x) \equiv 1/g(x) \text{ and } f(x) \equiv 1/(g(x) + c^2/g(x))$$

where g(x) and c are defined above. These two functions have identical calculated values, after rounding, for every x that can be keyed into the calculator, which consequently can't tell one from the other despite the fact that at x = 10/3 the first has a pole, f(10/3) = ∞, and the second a zero, f(10/3) = 0. Starting from straddling initial guesses x = 1 and x = 10 the SOLVE key finds a "root" of both equations f(x) = 0 to lie between 3.333333333 and 3.333333334 after only 49 samples. The user, not the calculator, must decide whether the place where f(x) changes sign is a root of f(x) = 0 or not. A similar decision arises when both initial guesses lie on the same side of 10/3, in which case SOLVE ultimately finds a "root" of f(x) at some huge x with $|x| > 3.33 \times 10^{98}$, where the calculated value of f(x) underflows to zero. That huge x must be regarded as an approximation to $x = \pm\infty$ where both functions $f(\pm\infty) = 0$.

The foregoing examples illustrate how our inability to perform calculations with infinitely many figures makes equation solving difficult. What makes equation solving impossible, even if rounding errors never happened, is our natural desire to decide after only finitely many samples of f(x) whether it never vanishes. Any procedure that claims to accomplish this task in all cases can be exposed as a fraud as follows:

> First apply the procedure to "solve" f(x) = 0 when $f(x) \equiv -1$ everywhere, and record the finitely many sample arguments $x_1, x_2, x_3, \ldots, x_n$ at which f(x) was calculated to reach the decision that f(x) never vanishes. Then apply the procedure again to $f(x) \equiv (x-x_1)\cdot(x-x_2)\cdot(x-x_3)\cdot(\ldots)\cdot(x-x_n)-1$. Since both functions f(x) take exactly the same value, −1, at every sample argument, the procedure must decide the same way for both: both equations f(x) = 0 have no real root. But that is visibly not so.

So equation solving is impossible in general, however necessary it may be in particular cases of practical interest. Therefore, ask not whether SOLVE can fail; rather ask, "When will it succeed?"

Answer: Usually.

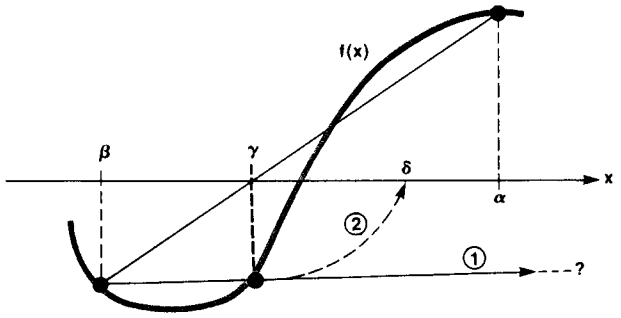


Fig. 6. *In the HP-34C, once two samples of f(x) with opposite signs have been discovered, the secant line (1) is bent to (2) whenever necessary to prevent an iterate δ from escaping out of the shortest interval known to contain a place where f(x) reverses sign.*

secant occasionally as illustrated in Fig. 6. Another small modification to compensate for roundoff in the secant formula (equation 1) protects it from the premature termination illustrated in Fig. 7. Although **SOLVE** can now guarantee convergence ultimately, that might not be soon enough since ultimately we all lose patience. Fortunately, convergence cannot be arbitrarily slow. At most six and normally fewer iterations suffice to diminish either successive errors $|x-\zeta|$ or successive values $|f(x)|$ by an order of magnitude, and rarely are more than a dozen or two iterations needed to achieve full ten-significant-digit accuracy. So fierce is the bent-secant iteration's urge to converge that it will converge to a pole (where $f(x)=\infty$) if no zero (where $f(x)=0$) is available, and this is just as well because poles and zeros cannot be distinguished by numerical means alone (see page 23).

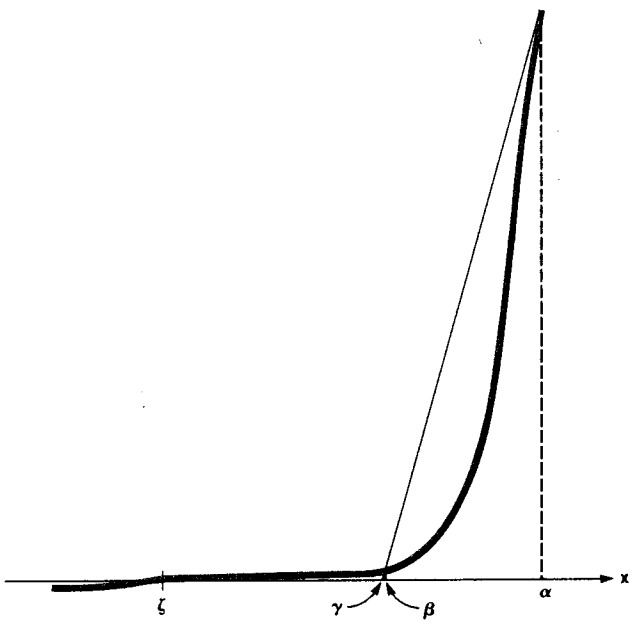


Fig. 7. *With a wild initial guess α the rounded value of γ may coincide with β. This convinces some equation solvers that γ is the root.* **SOLVE** *perseveres until it locates the root ζ correctly.*

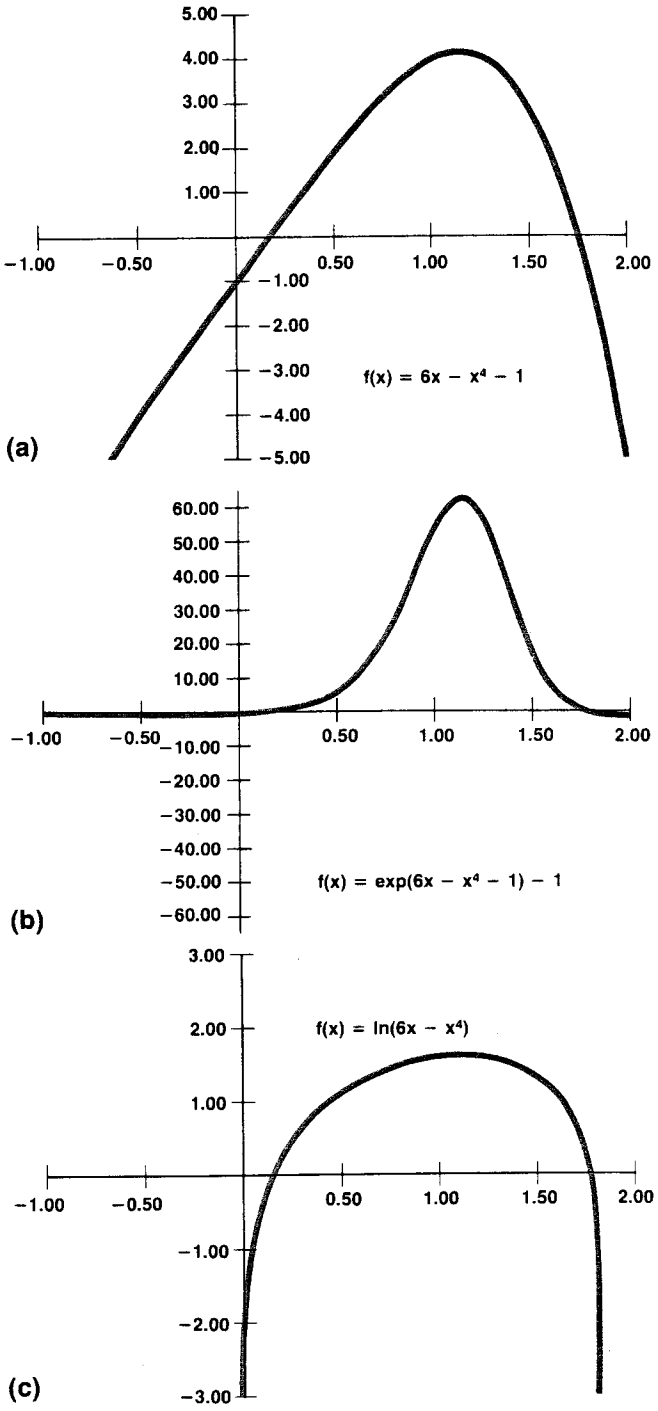


Fig. 8. *These three equations all have the same roots, but (a) is easy to solve, (b) with a bad initial guess gets worse, and equation (c) is defined only close to its roots.*

What does **SOLVE** do when all the values f(x) sampled so far have the same sign? As long as successive samples f(x) continue to decline in magnitude, **SOLVE** follows the secant formula (equation 1) with two slight amendments. One amendment prevents premature termination (see Fig. 7). The other deals with nearly horizontal secants, when $f(\alpha)=f(\beta)$ very nearly, by bending them to force $|\gamma-\beta|\leq$

$100|\beta-\alpha|$, thereby diminishing the secant iteration's tendency to run amok when roundoff becomes significant. Convergence now cannot be arbitrarily slow. As long as successive samples f(x) continue to decline in magnitude without changing sign they must decline to a limit at least as fast, ultimately, as a geometric progression with common ratio 1/2, and usually much faster. When samples f(x) decline to zero, **SOLVE** finds a root. When they decline to a nonzero limit, as must happen when $f(x) = 1 + e^x$ or otherwise declines asymptotically to a nonzero limit as $x \to \pm\infty$, **SOLVE** discovers that limit and stops with either ERROR 6, meaning no root was found, or $\pm 9.999999999 \times 10^{99}$, meaning overflow, in the display.

A different approach is needed when a new sample $f(\gamma)$ exhibits neither a different sign nor a diminished magnitude. To avoid the dithering exhibited in Fig. 5b, **SOLVE** sets the secant formula (equation 2) aside. Instead, it interpolates a quadratic through the three points $[\alpha,f(\alpha)]$, $[\beta,f(\beta)]$, $[\gamma,f(\gamma)]$ and sets δ to the place where that quadratic's derivative vanishes. In effect, δ marks the highest or lowest point on a parabola that passes through the three points. **SOLVE** then uses δ and β as two guesses from which to resume the secant iteration. At all times β and $f(\beta)$ serve as a record of the smallest $|f(x)|$ encountered so far.

But the parabola provides no panacea. Roughly, what it does provide is that if $|f(x)|$ has a relatively shallow minimum in the neighborhood of β and δ, the calculator will usually look elsewhere for the desired root. If $|f(x)|$ has a relatively deep minimum the calculator will usually remember it until either a root is found or **SOLVE** abandons the search.

The search will be abandoned only when $|f(\beta)|$ has not decreased despite three consecutive parabolic fits, or when accidentally $\delta = \beta$. Then the calculator will display ERROR 6 with β in the X register, $f(\beta)$ in the Z register, and γ or δ in the Y register. Thus, instead of the desired root, **SOLVE** supplies information that helps its user decide what to do next. This decision might be to resume the search where it left off, to redirect the search elsewhere, to declare that f(x) is negligible so x is a root, to transform $f(x)=0$ into another equation easier to solve, or to conclude that f(x) never vanishes.

When invoked from a running program **SOLVE** does something more useful than stop with ERROR 6 in the display: it skips the next instruction in the program. The calculator's user is presumed to have provided some program to cope with **SOLVE**'s possible failure to find a root, and then **SOLVE** skips into that program. This program might calculate new initial guesses and reinvoke **SOLVE**, or it might conclude that no real root exists and act accordingly. Therefore, **SOLVE** behaves in programs like a conditional branch: if **SOLVE** finds a root it executes the next instruction, which is most likely a **GTO** instruction that jumps over the program steps provided to cope with failure. Therefore the HP-34C, alone among handheld calculators, can embed equation-solving in programs that remain entirely automatic regardless of whether the equations in question have solutions.

Some Problem Areas

Equation solving is a task beset by stubborn pathologies; in its full generality the task is provably impossible (see page 23). Even though equations that matter in practice may not fall into the Chasm of the Impossible, yet they may teeter on the brink. Rather than leave the user teetering too, the HP-34C Owner's Handbook devotes two chapters to **SOLVE**, one introductory and one more advanced. The second chapter discusses equation solving in general rather than the **SOLVE** key alone, and supplies the kind of helpful advice rarely found in textbooks. Here follow examples of things that users might need to know but are unlikely to have learned except from bitter experiences, which the Handbook tries to forestall.

Hard versus Easy Equations. The two equations $f(x)=0$ and $\exp(f(x))-1=0$ have the same real roots, yet one is almost always much easier to solve numerically than the other. For instance, when $f(x)\equiv 6x-x^4-1$ the first equation is easier. When $f(x)\equiv\ln(6x-x^4)$ the second is easier. See Figs. 8a-8c.

In general, every equation is one of an infinite family of equivalent equations with the same real roots, and some of those equations must be easier to solve than others. If your numerical method fails to solve one of those equations, it may succeed with another.

Inaccurate Equations. Numerical equation solvers have been known to calculate an equation's root wrongly. That cannot happen to **SOLVE** unless the equation is wrongly calculated, which is what happens in the next example. This example resembles equations that have to be solved during financial calculations involving interest rates or yields on investments. For every $p \geq 0$ the equation

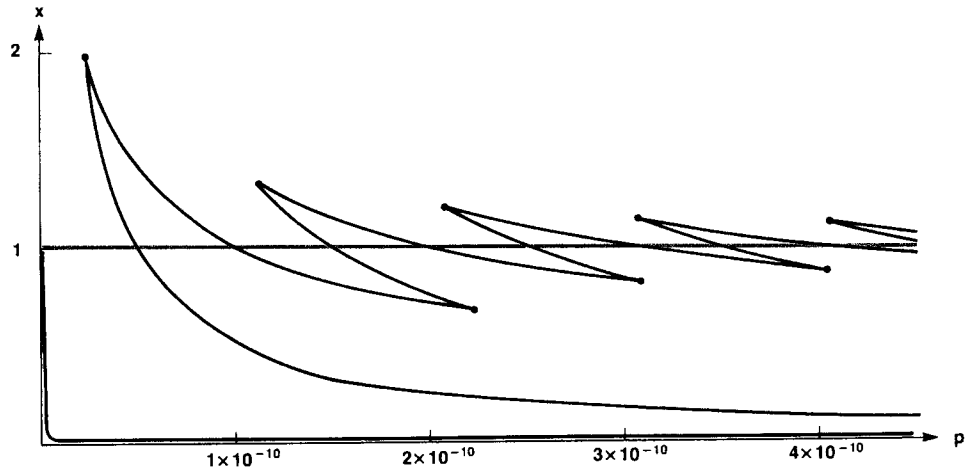


Fig. 9. *The jagged solid line is a graph of the ostensible roots of* $x-(1-\exp(-xp))/xp = 0$ *calculated carrying ten significant digits. The colored line is a plot of the correct root* $x = 1$ *(to nine significant digits) obtained by a rearranged calculation.*

$$x - h(px) = 0,$$

where $h(0) = 1$ and $h(z) \equiv (1 - \exp(-z))/z$ if $z \neq 0$, has just one root x, and $0 < x \leq 1$. The colored line in Fig. 9 plots this root x against p, and shows how smoothly $x \rightarrow 1$ as $p \rightarrow 0$. But when that root x is calculated numerically for tiny values of p using the most straightforward program possible, something awful happens, as shown by the black graph in Fig. 9. That serrated graph reflects the capricious way in which the calculated equation's left-hand side changes sign—once for $p = 10^{-11}$ at "root" $x = 10^{-88}$, seven times for $p = 2.15 \times 10^{-10}$ at "roots" $x = 4.65 \times 10^{-90}$, 0.233, 0.682, 0.698, 0.964, 1.163 and 1.181. All those "roots" are wrong; the correct root is $x = 0.999999999\ldots$. These aberrations are caused by one rounding error, the one committed when $\exp(-px)$ is rounded to 10 significant digits. Carrying more figures will not dispel the aberrations but merely move them elsewhere.

To solve $x - h(px) = 0$ correctly one must calculate $h(z)$ accurately when z is tiny. Here is the easiest way to do that: if $\exp(-z)$ rounds to 1 then set $h(z) = 1$, otherwise set $h(z) = (\exp(-z) - 1)/\ln \exp(-z)$. This reformulation succeeds on all recent HP handheld calculators because the **LN** key on these calculators retains its relative accuracy without degradation for arguments close to 1 (see reference 1). Consequently, $\ln \exp(-z)$ conserves the rounding error in the last digit of $\exp(-z)$ well enough for that error to cancel itself in the subsequent division, thereby producing an accurate $h(z)$ and a trustworthy root x.

Generally, wrong roots are attributable more often to wrong equations than to malfunctioning equation solvers. The foregoing example, in which roundoff so contaminated the first formula chosen for f(x) that the desired root was obliterated, is not an isolated example. Since the **SOLVE** key cannot infer intended values of f(x) from incorrectly calculated values, it deserves no blame for roots that are wrong because of roundoff. Getting roots right takes carefully designed programs on carefully designed calculators.

Equations with Several Roots. The more numerous the roots the greater is the risk that some will escape detection. Worse, any roots that cluster closely will usually defy attempts at accurate resolution. For instance, the double root in Fig. 3c ought to be $x = \ln 4 = 1.386294361$ instead of 1.386277368, but roundoff in the 10th decimal causes the calculated f(x) to vanish throughout $1.386272233 \leq x \leq 1.386316488$, thereby obscuring the last half of the double root's digits. Triple roots tend to lose 2/3 of the digits carried, quadruple roots 3/4, and so on. All these troubles can be attacked by finding where the first few derivatives $f'(x)$, $f''(x)$, etc. vanish, but nobody knows how to guarantee victory in all cases.

What Have We Learned?

The reader will recognize, first, how little the pathologies illustrated above have to do with the specifics of the **SOLVE** key, and second, how nearly certain is the user of so powerful a key to stumble into pathologies sooner or later, however rarely. While the **SOLVE** key enhances its user's powers it obliges its user to use it prudently or be misled.

And here is Hewlett-Packard's dilemma. The company cannot afford a massive effort to educate the public in numerical analysis. But without some such effort most potential purchasers will remain unaware of **SOLVE**'s value to them. And without more such effort many actual purchasers may blame their calculator for troubles that are intrinsic in the problems they are trying to **SOLVE** . To nearly minimize that required effort and its attendant risks, **SOLVE** has been designed to be more robust, more reliable and much easier to use than other equation solvers previously accepted widely by the computing industry. Whether that effort is enough remains to be seen. Meanwhile we enjoy the time **SOLVE** saves us when it works to our satisfaction, which is almost always.

Acknowledgments

I am grateful for help received from Dennis Harms, Stan Mintz, Tony Ridolfo and Hank Schroeder. Hank wrote the Handbook's chapters on **SOLVE**. Tony found ways to improve the **SOLVE** key's program while microcoding it. Dennis contributed some improvements too, both to the program and to this explanation of it, but I owe him most thanks for, along with Stan, supporting our efforts enthusiastically despite justifiable doubts.

Reference

1. D.W. Harms, "The New Accuracy: Making $2^3=8$," Hewlett-Packard Journal, November 1976.

William M. Kahan

William Kahan is professor of mathematics and computer science at the University of California at Berkeley. An HP consultant since 1974, he has helped develop increasingly accurate arithmetic and elementary functions for the HP-27, 67/97, 32E, and 34C Calculators, financial functions for the HP-92 and 38E/C, and other functions for the HP-32E and 34C, including $\int$ and SOLVE for the 34C. A native of Toronto, Canada, he received his BA and PhD degrees in mathematics and computer science from the University of Toronto in 1954 and 1958, then taught those subjects at Toronto for ten years before moving to Berkeley. A member of the American Mathematical Society, the Association for Computing Machinery, and the Society for Industrial and Applied Mathematics, he has authored several papers and served as a consultant to several companies. He is married, has two teenage sons, and lives in Berkeley.