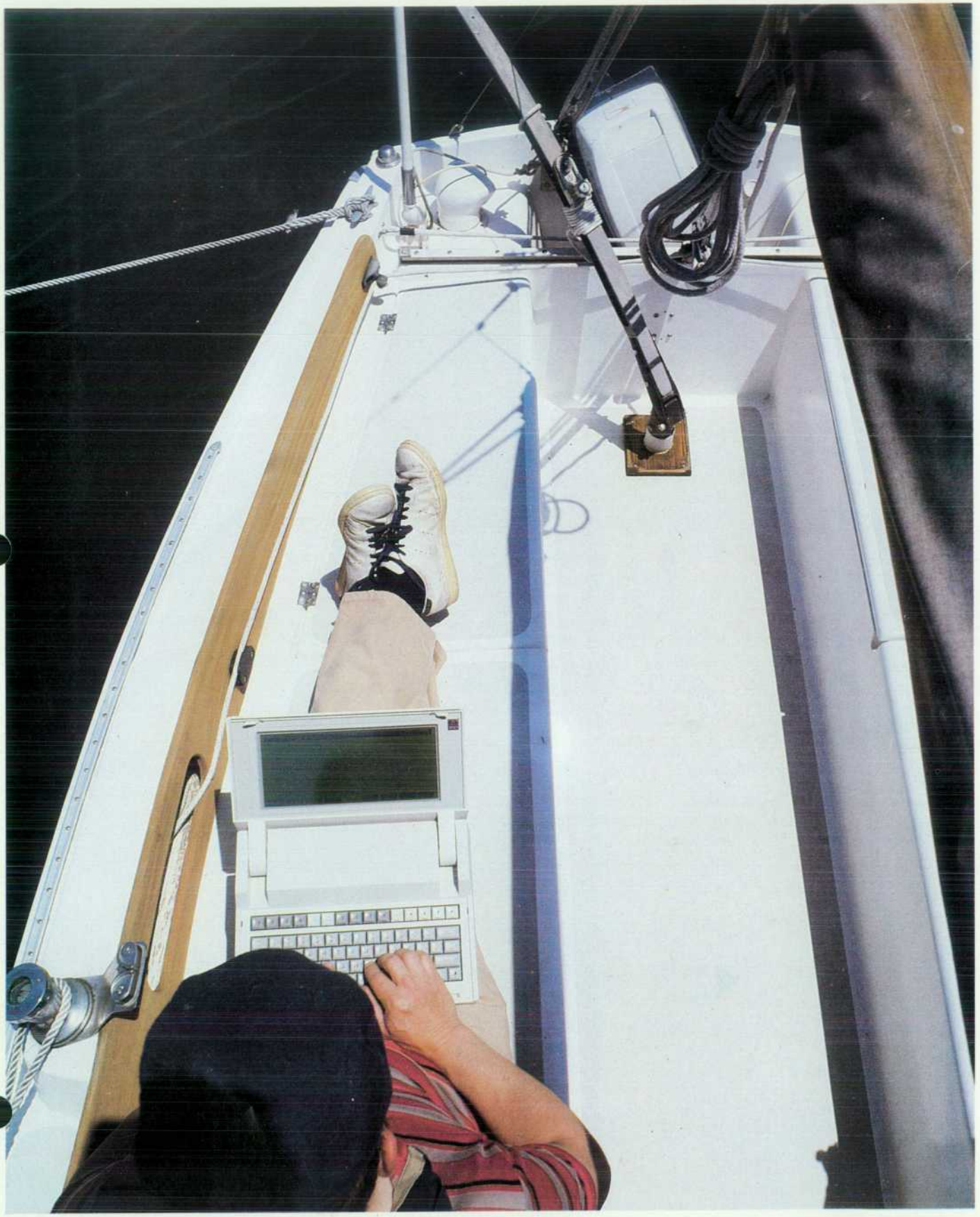
HEWLETT-PACKARD
JOURNAL

JULY 1986



HEWLETT-PACKARD
JOURNAL

July 1986 Volume 37 • Number 7

Articles

In this Issue

HP's briefcase-portable computers, The Portable and the Portable Plus, are designed for professional people who need the power of a personal computer when they are away from their offices. The Portable has as much memory and processing power as a typical desktop personal computer, and the Portable Plus has a larger memory capacity and other enhancements. Both computers use the same industry-standard MS™-DOS operating system as most desktop PCs. According to HP studies, over 80% of HP Portable owners use their portable computer as a go-anywhere complement to their desktop PC, running the same software on both machines and transferring data between them as needed. In this issue, the designers of The Portable and the Portable Plus give us the inside story of these machines. After a discussion of the differences between the portable and office environments and an overview of the design (page 4), we're given a tour of the I/O and data communications facilities (page 14), and a description of the Personal Applications Manager, which gives a novice user easy access to most of the features of the MS-DOS operating system (page 18). Memory management, including the concept of the "electronic disc" and memory expansion in the Portable Plus, is the subject of the article on page 21. The fast-turnaround design of a hybrid controller for the Portable Plus' 25-line liquid-crystal display is described on page 25, and the article on page 28 tells how the Portable Plus can be customized by the addition of special ROMs. Using a ROM Development Package supplied by HP, customers and software developers can produce custom ROMs for this purpose with no assistance from HP.

The HP-UX operating system is HP's version of AT&T Bell Laboratories' UNIX™ System V™ operating system, with enhancements. Described on page 34 is the latest HP-UX release for HP 9000 Series 300 Engineering Workstations. This release provides features such as virtual memory and local area networking, and several new features including support for device I/O, X.25 wide area networking, and a window-oriented human interface for the new Series 300 bit-mapped displays.

LANs—local area networks—are networks of computers located entirely within a single building or at a single site. Various proposed and accepted standards, or protocols, define the physical interconnections, data formats, and network control procedures that guarantee that the computers on a given LAN will communicate effectively, even if they are manufactured by different companies. A widely accepted protocol is defined by IEEE Standard 802.3, which is nearly identical to the Ethernet protocol developed by Xerox Corporation and first promulgated by Xerox, Digital Equipment Corporation, and Intel Corporation. On page 42, a group of designers from HP's Colorado Telecommunications Division describe the design of a new protocol analyzer for troubleshooting and maintaining IEEE 802.3 and Ethernet local area networks. Capable of monitoring, simulating, and analyzing traffic on a LAN, the HP 4971S provides a powerful filtering capability—16 user-definable filters for selective data capture—and the ability to display up to 500 node identifiers as easy-to-remember names.

-R. P. Dolan
Editor

What's Ahead

Next month's issue will present the processor and I/O system definitions of the new HP Precision Architecture. Other articles will describe the HP Precision simulator and performance analysis methods.

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers. Letters must be brief and are subject to editing. Letters should be addressed to: Editor, Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, CA 94304, U.S.A.

Design of HP's Portable Computer Family

The Portable and Portable Plus Computers are compact, lightweight, battery-powered personal computers with built-in software and 80-character-line liquid-crystal displays designed for use by professionals who need portable computing capability in their work.

by John T. Eaton, Carl B. Lantz, Clifford B. Cordy, Jr., James W. Pearson, Michael J. Barbour, Courtney Loomis, and Ella M. Duyck

IN MAY 1984, Hewlett-Packard introduced The Portable Personal Computer (Fig. 1), HP's concept for applying the power of personal computing to problems in the field. The Portable contains as much memory and processing power as a typical personal computer, but is battery-powered and the same size as a three-ring notebook. More recently, the Portable Plus (Fig. 1) was added to HP's portable computer family. The Portable Plus has a larger display, enhanced data communications, and the greatest memory capacity (1.28M bytes) currently available in a battery-powered computer.

Portability and Enhancements

Although HP's portable computers are as powerful as many desktop versions, they have a number of enhancements that are necessary to satisfy the requirements of the portable user:

- Electronic disc
- Operating system in ROM
- Video interface (Portable Plus only)
- Memory expansion (Portable Plus only)
- Permanently resident applications software
- 80-character-per-line liquid-crystal display
- Built-in modem (optional in Portable Plus)
- Personal Applications Manager
- Built-in RS-232-C/V.24 and HP-IL interfaces
- Battery power with power-saving features.

Portable computers are used differently on the road than they are at a desk. The typical desktop computer takes several minutes after it is turned on to run self-tests and load its operating system from disc. Hence, many desktop users turn their machines on at the start of the day and

Fig. 1. *The Portable (left) comes complete with MS™-DOS 2.11, HP's Personal Applications Manager, MemoMaker, and Terminal Emulation software, and 1-2-3™ from Lotus™. The Portable contains 272K bytes of memory, a 300-baud modem, an RS-232-C/V.24 serial interface, an HP-IL interface, a 16-line-by-80-column liquid-crystal display, lead-acid battery power supply, and a full-size keyboard. The Portable Plus (right) comes complete with MS-DOS 2.11, and HP's Personal Applications Manager, SECURE, HPLink, EDLIN, and Terminal Emulation software. The Portable Plus features a 25-line-by-80-column antiglare liquid-crystal display, 128K bytes of RAM, an RS-232-C/V.24 serial interface, an HP-IL interface, a full-size keyboard with embedded numeric keypad, lead-acid battery power supply, and two expansion drawers for additional RAM and/or application ROMs. A built-in 300/1200-baud modem is optional.*

leave them running all day long. Portable computers, on the other hand, are normally turned on and off quite frequently. Portable users turn on their machines when they need to use them and then turn them off when they are finished. It is not acceptable to force a user to wait during a lengthy start-up sequence each time a portable computer is turned on. Access to data must be quick and easy.

HP's portable computers can use a portion of their internal memory as an electronic disc. User files can be stored in battery-backed CMOS RAM and application programs are kept in ROM. This electronic disc is faster than a hard disc and safe from the problems of disc wear and head crashes.

You can turn on an HP portable computer, select an application, and load in a data file in about 15 seconds. To provide this capability, several features common to desktop systems were modified for the portable environment. Automatic power-on self-tests are replaced with a user-selected self-test. This allows for battery backup of the electronic disc so that it retains all of its files while the unit is powered off. A typical desktop system cannot do this because its power-on diagnostics usually erase all the files in an electronic disc each time the computer is powered on.

The operating system often must be reloaded during a typical computer session. The Portable and Portable Plus have MS™-DOS completely in ROM so it can be loaded faster than from a hard disc. Portions of the operating system drivers are executed directly from the ROM. This provides a double bonus in that this code does not have to be copied each time the computer is turned on, and the RAM space normally required to hold this code is available for the user's applications and data.

New features for the Portable Plus include video interface capability and memory expansion. Whereas The Portable is a closed system, the Portable Plus has an open architecture with two user-installable plug-in ports. This allows the product to be enhanced from the base machine to a higher-performance machine. (See "Plug-In Memory" on page 9.) The Portable Plus can be configured with a ROM drawer that can contain up to 1.5 megabytes of ROM or EPROM code. Instead of having to carry a disc for each application, a user can install desired applications permanently inside the machine and have them available at a moment's notice. The Portable comes with word processing (Memomaker), terminal emulation, and spreadsheet, graphics, and file management (1-2-3™ from Lotus™) software built-in.

The Portable Plus incorporates several new technologies. Its liquid-crystal display (LCD) is larger (25 lines) than The Portable's LCD (16 lines) and its optional built-in modem operates at 300 and 1200 baud compared to The Portable's standard 300-baud modem. Not so obvious is an increase in packaging density achieved with the use of 1M-bit CMOS ROMs. Other advances, such as low-power RS-232-C/V.24 serial interface drivers, result in increased battery life.

There are many differences between the portable and desktop environments that must be addressed in the design of any portable computer. One example is documentation and ease of use. Portable computer users are not likely to have all the computer and software documentation with them at all times. Programs should be easy to use and have as much on-line help as possible. The Personal Applications Manager (PAM, see article on page 18) and its built-in help facilities are resident in The Portable and Portable Plus to achieve these objectives.

Portable computers are used mostly in situations where desktop computers cannot be used because of their size and power requirements. But there are times when it is advantageous to be able to use a portable computer in a desktop environment. Battery-powered portables have the built-in advantage of an uninterruptable power source that can power the computer for hours. A desktop computer user who needs dependable computing might otherwise spend hundreds of dollars for a device that can power a desktop computer for only 15 minutes after a power failure.

The smaller size of a portable computer can be another advantage. Many users discover that there is not much room left on a desk after putting a normal desktop computer on it. A portable computer takes up very little desk space when not in use and can even be stored out of sight in a drawer. It is easy to set up and can be fully operational with the touch of a key. Portable computers are quieter since they do not require noisy fans and they do not anchor the user to a desk. When peripheral devices such as a printer are needed, they can be connected to HP's portable family by using the HP-IL (Hewlett-Packard Interface Loop),[1] which provides both a quick easy connection and a thin cable that is easy to handle.

The built-in software has several features that help prolong battery life (see box on page 10). A time-out function powers off the computer if it has been inactive for a period of time. This time-out is nondestructive, meaning that a touch of a key restores the computer to exactly where it was when it timed out. If you are called away from the computer and it times out while you are gone, you can pick up exactly where you left off when you return. This also works when the computer shuts down because of a nearly discharged battery. A user has up to one week after the computer shuts down to recharge its battery and be able to continue with no loss of data or programs.

Another method used to prolong battery life is to put the computer into a low-power mode when it is not actively performing a calculation. Implementing this function is somewhat tricky because a portable computer could be running an application that was not written with battery life in mind. The Portable and Portable Plus use a special algorithm that monitors I/O use and the frequency of keyboard status checks to decide if an application is doing something or simply waiting for the user to press a key. If the algorithm considers that the computer is waiting, then it places the computer in the low-power mode until the next key press.

Hardware Design

Putting all the components for a complete personal computer into a box the size of a notebook is not a trivial task, but necessary for the design of a portable system. It also requires that the circuitry consume the minimum amount of power possible to conserve battery life. CMOS circuitry was used throughout The Portable and Portable Plus Computers to achieve this goal. The hardware system (Fig. 2) is designed around a CMOS version of the 80C86 micropro-

Inside the LCDs for The Portable and Portable Plus

A truly portable computer requires a display technology that meets the requirements of portable design. In the case of The Portable and Portable Plus Computers, this means low power consumption, light weight, ruggedness, and a large display format. An 80-character screen width, with a minimum of sixteen alphanumeric lines and full graphics capability was determined to be the smallest acceptable size. Use in HP products targeted for a competitive market made a low price a particularly critical criterion in choosing the type of display.

In examining potential technologies, a liquid-crystal display (LCD) was the only alternative at the time that came close to meeting all these constraints. The use of an LCD provides a very-low-power (about an eighth of a watt), sturdy display. The low power requirements of the reflective LCDs chosen for this application stem from the fact that they are nonemissive (i.e., they have no internal light source). Since these thin displays use ambient light as their source, there are some trade-offs in using this technology. Reflective LCDs are most useful in well-lighted situations where the overhead light is diffuse and plentiful. Direct light can cause problems with glare. Lack of adequate light will result in poor readability. Even in adequate light, the technique used to drive these panels only gives acceptable contrast over a small range of viewing angles. To accommodate this limitation, the LCD panel is mounted in a display case with adjustable tilt.

The particular type of LCD used in HP's portable computers is a twisted-nematic LCD. Nematic refers to the phase of the liquid-crystal material. In the nematic phase, the long axes of coplanar liquid-crystal molecules align in a parallel orientation. A common analogy is that of pencils thrown into a box. When the box is shaken, the pencils representing the molecules tend to line up parallel to one another. Layers of this nematic LC are sandwiched between two pieces of grooved glass. The glass on the top has grooves on an axis perpendicular to that of the bottom glass. As a result, the surface tension at each glass-LC interface causes successive planes of the medium to form a spiral, twisting upward from the lower glass to the upper glass. Optically this twisted structure has a special property—it is capable of rotating linearly polarized light. By placing polarizing filters above and below this glass sandwich, each parallel to the texturing on the adjacent glass, a polarizer with a 90-degree twist is formed (see Fig. 1).

If a voltage is applied perpendicularly across this medium, the helical shape of the LC structure is perturbed, and the LC molecules align in the direction of the electric field lines, no longer twisting the light polarization. In this condition, the effects of the surface polarizers cancel and incident light is absorbed. Thus, in the relaxed state, the LCD passes polarized light and in the excited state the light is blocked. For the LCD, the potential is applied across the LC medium using transparent row and column conductors etched on the inner surfaces of the glass plates.

By placing a reflector on the rear surface of the panel, incident light can be used as the source for this light modulator. The fact that the liquid-crystal structure is affected by the application of voltage rather than the passing of current (it is a capacitive element), along with the reflector and some intelligent design of the drive circuitry, minimizes the power needed by this type of display.

The LCD panels purchased for HP's portable computing products are procured as a complete assembly having a four-quadrant multiplexed display and are interfaced with a printed circuit board containing surface-mounted drive circuitry. Panels with this type of configuration were chosen because they offer the best possible contrast for a large number of rows. Also for this reason, a 480-pixel-wide display, rather than the conventional 640 pixels, was chosen. All that the designer needs to supply for the display are serial bits for each quadrant, and the appropriate timing signals.

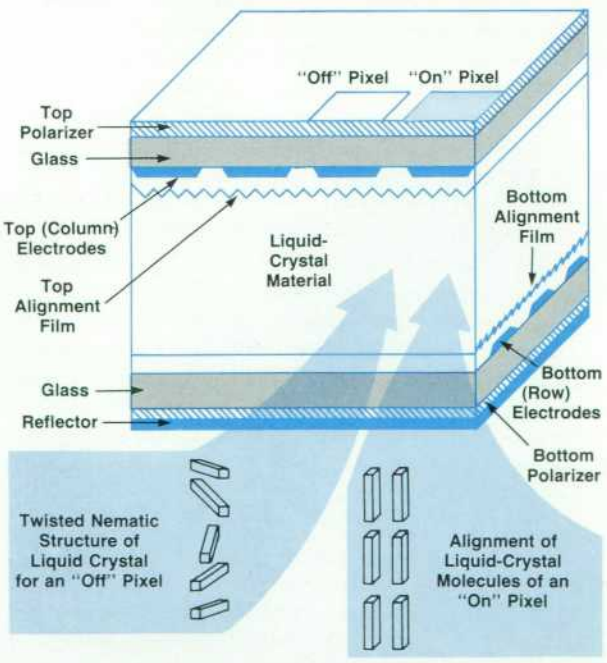


Fig. 1. *Cross section of liquid-crystal display.*

Contrast

The issue of contrast is a complex subject, depending mostly on the particular liquid-crystal material's properties, the operating temperature, and the number of rows being multiplexed. Multiplexing is used because direct drive would require an impractical number of interconnects. Each row line is selected while the representative voltages for on and off dots are set up on the column lines. Rows not being selected are biased to a background voltage. Overall, the more rows, the smaller the difference in rms voltage between selected and nonselected pixels for a given maximum bias. Since the alignment of the LC helix is perturbed relative to the amount of rms potential it experiences, the difference between on and off pixels, the contrast, is limited by the multiplexing and peak voltages of the drivers. Contrast is also closely linked to viewing angle, and a critical specification for readability relates these two factors.

The LCD's glass is separated into top and bottom sectors to reduce the number of rows being multiplexed by a factor of two. Each of these sections is subdivided to form four quadrants. Column lines enter from both the top and the bottom of the panel, and are driven by four 240-bit parallel-load CMOS latches whose outputs are capable of an 18V swing. The row lines are scanned in a similar manner to a keyboard scan, with the top and bottom

refresh occurring simultaneously. By keeping the refresh rate low, less power is used, EMI is reduced, and an acceptable level of flicker is achieved.

At the time of release of The Portable in May 1984 only 64-way multiplexed LCDs could be produced with acceptable contrast. By mid 1985, the 100-way multiplexed panel was used in the Portable Plus with improved contrast and viewing angle. Better contrast will be achieved with the development of better LC materials and CMOS drivers with greater voltage swings.

Glenn Adler
Development Engineer
Portable Computer Division
(now with Beckman Instruments)

cessor that was developed by Harris Semiconductor.

The Portable contains 272K bytes of CMOS static RAM and 384K bytes of ROM. The Portable Plus contains less memory in the base unit (128K bytes of RAM and 192K bytes of ROM), but is currently expandable via plug-in cards to 1.28M bytes of RAM or 1.5M bytes of ROM.

The Portable Plus is designed for software compatibility with The Portable and to accommodate plug-in expansion. System RAM starts in memory space at address 00000_{16} and grows upward. Display memory begins at address 80000_{16}, and the system ROMs reside in the upper addresses. A 256K-byte swap space is used to swap in a pair of one-megabit application ROMs completely when needed. The I/O space allocation of built-in devices is compatible with The Portable. The 32K bytes of I/O space not used in The Portable is used in the Portable Plus for its plug-in drawers. (For details about the memory system and its management, see the article on page 21.)

CPU

The 80C86 processor operates at a 5.333-MHz clock rate, which is about 12% faster than processors used in typical personal computers. This, combined with the increased efficiency of the 80C86 16-bit bus, lets The Portable and Portable Plus operate almost twice as fast as the standard 8088-based computer. The 5.33-MHz clock rate was chosen to provide increased processing power in addition to saving precious printed circuit board area. The 82C84A clock generator and a 74HC393 dual binary counter form the clock generator circuit. The 82C84A uses a 16-MHz crystal to create a 5.33-MHz (33% duty cycle) clock for the 80C86 and a 2.67-MHz (50% duty cycle) clock for peripherals. The 16-MHz output clocks a counter in the 74HC393, creating 1, 2, 4, and 8-MHz signals. To use all of the available space, the unused half of the 74HC393 is pressed into service as a counter for wait state insertion. In The Portable, the lower 512K bytes of memory space are allocated to system RAM, and run with no wait states. I/O devices requiring longer access times have one wait state to pull the open drain ready line low to insert additional 80C86 wait states.

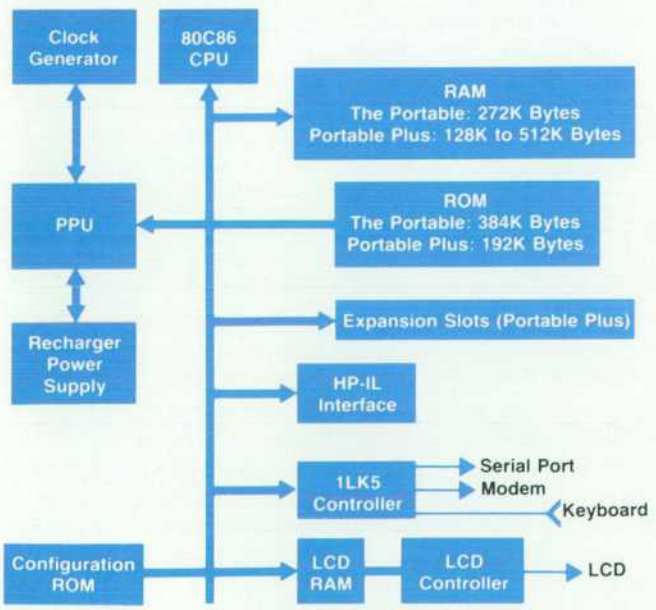


Fig. 2. *Block diagram of the hardware system for The Portable and Portable Plus Computers.*

The current drawn by the 80C86 is low compared to an NMOS part, but is still too high to leave running all the time. The Portable and Portable Plus contain a second microcomputer called the peripheral processor unit (PPU) that controls the system when the 80C86 is turned off. This processor (an MC146805) typically draws 1% of the current of the main processor and is left running all the time.

PPU and Battery Supply

The peripheral processor unit actually operates the computer because it controls the switches that power the rest of the system. The PPU contains a timer that is used to provide a real-time clock for the system. It also provides for time-zone conversions and alarms. Because it has a clock and controls the power, the PPU can wake the system up at a predetermined time to run an MS-DOS program and, when finished, it can then put the computer back to sleep. It monitors all the interrupt lines so that if an interrupt occurs while the main CPU is turned off, it can power up the CPU to service the interrupt. For example, the computer can be awakened by an incoming telephone call and can be programmed to answer it.

A fuel gauge value that represents the percentage of charge left in the battery is computed by the PPU using configuration data supplied by the main CPU and then displayed in the main menu screen of the computer's built-in Personal Applications Manager. The PPU knows what sections of the computer are active and drawing current. It also receives a signal from the charging circuit that indicates when the battery is being charged. It uses this data to calculate the amount of charge available in the battery at any time. Although the accuracy of this value will decrease as the battery ages, the only other way to determine the charge of the battery is to measure the terminal voltage, a method that would only give a result within 20% of the true value.

The PPU controls the charging circuit and can switch it

into a low-charge-rate, or float, mode when the batteries are fully charged. This prevents battery damage if the user leaves the computer plugged in constantly.

The battery charger is a current-limited, precision voltage regulated power supply. The output voltage is either 7V or 7.5V as selected by the PPU. As the battery nears full charge, it accepts less current. To speed up the final stage of charging, the charging voltage is switched from 7V to 7.5V for a period of time. This also ensures that each cell is fully charged. Thereafter, the supply remains at 7V for as long as the charger is plugged in.

One of the characteristics of the sealed lead acid batteries used in HP's portable computers is that optimum battery life is achieved by charging them whenever possible, using a charging voltage of 2.32V to 2.35 volts per cell. The optimum voltage is temperature dependent. Hence, the charger circuit is temperature-compensated to produce the optimum voltage over the range of −20°C to +60°C.

After discharge, each battery cell is slightly overcharged as mentioned above to ensure cell equalization. The PPU tailors the overcharge cycle to match the depth of cell discharge. The charger control circuit also determines if there is a shorted cell (which can happen if the battery is excessively discharged). If so, the main charger is turned off to protect the regulator.

One weakness of a lead acid battery is that it can be damaged if it is deeply discharged. This occurs if a battery is discharged to significantly beyond its normal capacity. To prevent this, the power switch is operated by the PPU. The PPU monitors the battery voltage to see if the battery charge is too low to power the rest of the system. If so, it refuses to turn on the 80C86 CPU. This prevents the user from running the battery down to a level where it may sustain damage. In such a case, the batteries still have enough charge to preserve memory for more than a week so the user can continue with the same application and data immediately after connecting the charger supply.

A single wall-mounted transformer serves as both an ac adapter and a power supply for the charging circuitry. The switch from ac to battery power is all electronic, which gives the HP portable computer family several advantages. If a user unplugs the transformer from the wall outlet (or if the power should fail), the computer instantly starts using its battery for power. Some portable computers use a mechanical switch in the charger plug and they lose power when this happens.

Battery Choice

Choosing a battery was a significant design activity. Nonrechargeable batteries were rejected as a source because their life would be intolerably short (7 to 13 weeks for alkaline D cells, depending on the number of cells used). The package is large enough to use batteries the size of D cells, however, so the choice was between three sealed lead acid cells or five nickel-cadmium (NiCd) cells. Size and weight per watt hour in the two chemistries are essentially the same. NiCd cells have several well-known problems: short lifetime of typically two years (the average data sheet is exceedingly optimistic), near impossibility of building an optimum charger, reduced lifetime caused by use of simple chargers, electrolyte leakage from dead cells, larger internal current leakage, and memory (inability to produce large energy cycles after being repeatedly used in small energy cycles). Lead acid batteries suffer from none of these problems. On the other hand, lead acid batteries are often damaged, sometimes ruined if they are run into an overdischarged state. This is easily avoided with a low-battery cutout as discussed earlier. Finally, lead acid batteries are less expensive.

The low internal leakage of the lead acid batteries provides a margin of safety for the user's data which simply is not available with other simple portable power systems. At room temperature, a fully charged lead acid battery will maintain memory (without other use) for nearly a year. Even if the computer is run until the low-battery-charge cutoff algorithm turns it off (which leaves about 10% of the battery charge), the memory will typically be maintained for a month at room temperature. Even at elevated temperatures, the lead acid battery has a relatively low self-discharge rate. At a constant temperature of 50°C, a fully charged battery will maintain memory for over four months. By comparison, a fully charged NiCd battery with no load at all will self-discharge at room temperature in about three months.

Power Supply

Since the battery supply selected always exceeds 5V and most of the electronics uses a regulated 5V, a simple linear regulator is used for the power supply. Regulators that were commercially available when The Portable was designed used a few milliamperes internally even under no-load conditions. It makes little sense to design a computer that only requires 0.25 mA in sleep mode if the regulator requires 2 mA at the same time. Newer regulator ICs have since been introduced that draw only about 50 μA, which is more acceptable. The no-load current of the custom regulator in The Portable is 12 μA for the reference, 6 μA for the feedback resistors (which you also need for the new IC regulators), and 3 μA internally in the regulator.

The display needs a −7V to −12V regulated supply that can be varied under control of the PPU. This was done with a feedforward, inverting regulating converter, an approach not commonly used for regulation. Unless a lot of second-order effects are considered in the design, the load regulation is poor. In this case, since the display looks like a constant resistance, the load regulation is not so important and supply stability is easily obtained since there is no feedback. The contrast ratio of the LCD screen is adjusted by varying the negative voltage. The supply design provides an output voltage that is constant to well within 1% over the range of the input voltage (5.5V to 7.5V) and to within 2% over a load current variation of at least 5%.

LCD Controller

A custom IC to control the liquid-crystal display was designed for both The Portable and the Portable Plus. Two factors drove the decision to design a custom chip in The Portable. First, no existing controllers were available that were compatible with the 80C86 bus. The additional circuitry needed to integrate them into the system would have been too expensive in terms of printed circuit board space. Second, the only controllers available were in surface-

mount packages while all other chips in The Portable are in dual in-line packages (DIPs). Adding a major manufacturing step for one component was undesirable.

An LCD presents design constraints not seen with CRT displays. Primarily, the LCD is a truly digital display—it has neither horizontal nor vertical blanking periods that allow for CPU access of display RAM. The data flows to the display in an uninterrupted stream. Access to display RAM has to be controlled so that the LCD controller always has priority while the CPU sees the display RAM as a normal segment of its address space.

In The Portable, an additional packaging constraint is in force. Namely, as mentioned above, manufacturing considerations required that the controller be packaged in a standard plastic DIP configuration. This restricts the pin count to a maximum of 48. In addition, the display RAM had to be made up of the same 8K×8-bit CMOS static RAM parts used elsewhere in The Portable to meet low-power requirements. This restriction prevented multiplexing of address and data to the RAM since these parts have separate address and data lines.

These constraints resulted in the LCD controller and the display RAM being placed on a private bus that can be used by either the CPU or the LCD controller. This private bus is isolated from the system bus by buffers under the control of the LCD controller. Data fetches by the LCD controller are arranged to leave windows during which the CPU can access the display RAM or the controller registers without interrupting data flow to the display. Any CPU access to either the display RAM or the controller registers causes the controller to halt the CPU by immediately pulling the ready line low. The controller then uses CPU access windows as available to move data between the system bus and either the display RAM or the control registers as appropriate. When the data transfer is complete, the ready line is released and the CPU cycle proceeds normally. In this way, display accesses are no different from conventional access methods as viewed by the software, despite the need for the CPU to wait for the next window of opportunity. The LCD controller also converts any 16-bit memory accesses from the CPU into two 8-bit accesses for the RAM.

Experience with the hybrid RAM in The Portable gave enough confidence in the hybrid technology to use it for the LCD controller in the Portable Plus (see article on page 25). With the limitation of 48 pads on the controller chip eliminated, the private bus used in The Portable was moved onto the controller chip. The result is a 77-pad IC with two complete 16-bit, nonmultiplexed bus interfaces—one interface for the 80C86 CPU and one interface between the controller and the display RAM. With the Portable Plus' two display RAM chips on the same hybrid substrate as the controller, the entire display control function is placed in a 48-pin DIP configuration package with enough otherwise unused pins to allow some of the Portable Plus' system glue logic to be implemented on the LCD controller chip.

The system of holding off the CPU with the ready line is the same in the Portable Plus as it is in The Portable. The LCD still must have priority in accessing display data.

Video Interface

When it was decided to provide a video interface for the Portable Plus, the machine design (both hardware and software) was firmly set. The principal design constraints were minimal hardware impact and no software impact. Thus, the information for the external video display is tapped from the data and clock lines driving the LCD panel. In this way, the existence of the video interface is invisible to the software. The drawback is that the software has no control of the video display independent of the LCD display.

The external video display normally shows a copy of what is on the LCD. It also has a freeze-frame feature that is activated manually with a pushbutton on the video interface box. In freeze-frame mode, the interface controller simply stops refreshing its buffer RAM from the Portable Plus display signals. Thus, the monitor can hold one frame of information (frozen) while the Portable Plus proceeds with its normal active display. This can be useful, particularly in word processing or spreadsheet analysis, for comparing results or to double the amount of displayed information.

Keyboard Controller

The keyboard controller was especially designed for the needs of this portable computer family. It has a built-in sleep mode that lets it watch the keyboard for any key strike while consuming far less than a milliwatt of power. The keyboard controller can wake up the system with an interrupt and then read the keyboard to see what key is pressed. This allows the system to be started with the touch of a key. The keyboard controller does not scan the matrix by strobing every line individually, but rather strobes all the rows at once and then all the columns. The rows are strobed low and the columns read to see if a key is pressed and then the columns are strobed high and the rows are read. This method will detect any key closure on the matrix but consumes very little power until a key is struck.

Plug-In Memory

The Portable is configured with over one-quarter megabyte of RAM and 384K bytes of ROM. One of the chief design goals of the Portable Plus was to increase this capacity while giving the user the ability to customize the machine's configuration. To do this, the Portable Plus is designed with two plug-in memory slots.

These drawer slots provide a means for customizing the hardware configuration according to the specific needs of various sectors of the computer marketplace. The documented open architecture of the Portable Plus gives independent hardware vendors easy access to the machine, opening opportunities for market expansion beyond those Hewlett-Packard is formally pursuing. The second general enhancement made possible by the drawer slots is the ability to expand the basic functionality of the machine. Both RAM and ROM can be added to the Portable Plus through these slots.

The ability to expand the basic functionality of a machine is especially important when designing a product that is at the leading edge of packaging technology. The physical constraints imposed by the packaging density of integrated circuitry (especially memory) result in a product whose basic functionality may be marginal in some applications. Expansion drawers provide an opportunity for improving

Low-Power Modes for Portable Computers

One of the first discoveries that a new user makes about Hewlett-Packard's portable computers, The Portable and Portable Plus, is that there is no **ON/OFF** switch. Each computer draws its power from a battery pack that will maintain the system for up to 20 hours under normal operating conditions. Two low-power modes are implemented to conserve battery power: powersave mode and sleep mode. The powersave mode is achieved through a CPU halt instruction. Sleep mode is similar to an off state since the display is turned off and many of the internal circuits are powered down. These low-power modes are controlled by a single-chip microcomputer known as the peripheral processor unit (PPU).

Powersave Mode

If an application frequently checks for keyboard activity, battery life can be improved by about 44% by allowing the system to enter the low-power halt known as powersave mode. In this mode there are no memory accesses since the CPU is halted, and therefore, the power drain is decreased. The PPU is made aware of the halt so that it can adjust its power consumption calculations to account for the reduced system activity. The result of these calculations is displayed in the main PAM (Personal Applications Manager) screen as a percentage indicating the level of remaining battery charge. The CPU remains halted until a hardware interrupt occurs.

The powersave mode is implemented by the keyboard driver and accessed through either the console driver or the BIOS interrupt hexadecimal 16. There are two situations (Fig. 1) that can put the system into powersave mode. First, if the keyboard driver is asked to return a key input and the keyboard buffer is empty, the system halts and waits until a key input becomes available. The halt state is terminated when a hardware interrupt occurs. The maximum amount of time that the CPU will be halted is dependent on the hardware timer interrupt interval (The Portable timer interrupts once per second and the Portable Plus timer interrupts 18 times per second). The interrupt is processed and the keyboard buffer is checked again to see if the CPU should do another halt.

The second situation occurs when an application is expecting a keystroke, but rather than letting the keyboard driver wait for it, the application does the waiting by repetitively reading the keyboard input status. The keyboard driver keeps count of the number of input status requests made in a one-second interval. If that count reaches a set limit, the driver halts in powersave mode. When a subsequent hardware interrupt occurs, the interrupt is processed, the status request counter is reset, and control is returned to the application.

The powersave mode can be disabled manually by a user in the PAM system configuration menu or, on the Portable Plus, it can be disabled programmatically by an application through the system service functions. In either case, the powersave mode is only disabled for the second situation described above. The keyboard driver will always enter the powersave mode when a keyboard read request is attempted while the keyboard buffer is empty.

Sleep Mode

During sleep mode, most of the system is electrically shut down and the PPU is in a low-power state monitoring system events. The CPU, modem, LCD display, and serial and HP-IL interfaces are turned off. Only display RAM, built-in RAM, plug-in RAM, keyboard hardware, and the PPU remain on. Coming out of sleep mode, the BIOS code completely restores the system—the LCD displays the same information as before the sleep and the suspended application resumes where it left off.

There are three events that prompt the BIOS code to put the computer to sleep. First, the sleep interrupt is available to all applications to put the computer into an off state. PAM calls this interrupt when the **OFF** softkey is pressed. Second, when the computer is on but not being used, it may eventually time out and go to sleep. The time-out feature is designed to help prevent running down the batteries when the computer has been left idle but not turned off. Finally, if the batteries are run down to a level providing only questionable power to the system, the BIOS code forces the computer to sleep to prevent the loss of data on the internal RAM disc.

The computer may go to sleep in the middle of an application. Once awakened, it must continue from exactly where it left off. The sleep routine must gather enough information from the current system so that once it is reset, the presleep state can be restored. The optional modem in the Portable Plus contains user-definable features that are lost when power is cut to the modem while sleeping. A special command is sent to the modem before sleeping that requests the current state of these features. The information is then kept in the system RAM, which remains powered during sleep. In both The Portable and the Portable Plus, the HP-IL chip registers and the address of the current stack are read and saved to be restored upon waking.

To initiate sleep mode, a command is sent to the PPU. The PPU remains powered during sleep mode to maintain the real-time clock, calculate the current power level, and watch to see if the system should be brought out of sleep. It will wake up the system when any key is pressed, an alarm goes off, when the modem or serial port receives a ring interrupt, or if an interrupt is received from a plug-in card.

Immediately after being reset, the CPU determines that it has

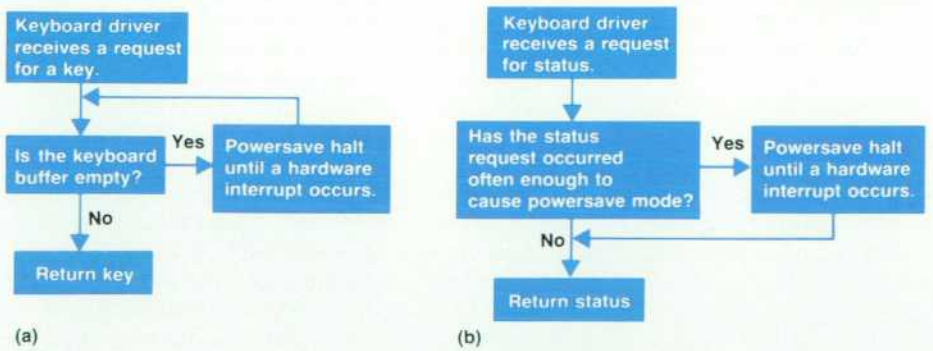


Fig. 1. *Powersave mode operation can be triggered by situation (a) or (b).*

been sleeping and that the system must be restored rather than rebooted. The stack is set up from the information saved in system RAM before the reset. The modem is reconfigured and the HP-IL chip status is restored. The system interval timer is restarted and the time displayed in the softkeys is updated. Finally an interrupt return instruction is executed to return to the active application.

Time-Out

Several factors determine whether or not the unit will time out. The BIOS code assumes that it is safe to stop in the middle of an application if it has asked the keyboard driver to return a key input and is waiting. If the keyboard buffer is empty, the driver waits a set amount of time before deciding to time out. The application may also be waiting for a key input if it is repetitively reading the keyboard input status. This is determined in the same way as the second powersave situation described above—the keyboard driver counts the number of input status requests in one second and waits for that count to reach a certain limit to time out. There is a difference, however, in that any other driver call or I/O activity will reset the count to zero. For example, a program in a simple loop checking keyboard status will certainly time out and go to sleep, but a program that alternates between checking the serial port status and checking the keyboard status will not. The computer cannot go to sleep in this case since, although a keystroke would wake the system up, any characters coming in from the serial port would be lost.

Once it has been determined that the application is only waiting for a key input, the computer makes a time-out decision (Fig. 2). On the Portable Plus the system checks for the presence of a serial or modem carrier. The intention is to not drop a carrier and lose the connection. The current power level and the presence of a battery recharger also affects whether or not to time out. The Portable will not time out if the recharger is connected; the Portable Plus will not time out if the recharger is connected and the current battery level is above 80%.

Using the PAM system configuration menu, the user can manually disable the time-out feature or change the length of time that the system must be idle before the time-out occurs. In the Portable Plus, a system service is provided to modify the rate at which the keyboard status calls must occur to cause a time-out. This rate can be customized by an application to either prevent or force a time-out.

Battery Shutdown

When the battery charge has fallen too low to maintain a reliable system, the computer is put to sleep. At this point, the most important use of the remaining power is to preserve the data on the RAM disc by conserving all power for the system RAM. The PPU determines that a shutdown should occur and alerts the CPU. The computer goes to sleep and will not wake up until the recharger is plugged in and a key is pressed.

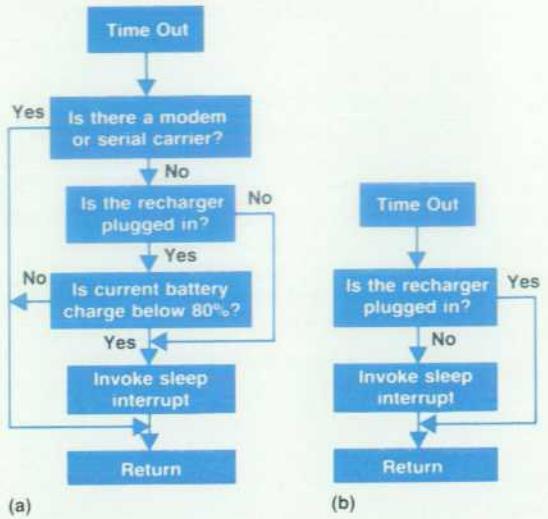


Fig. 2. *Time-out decision process for (a) Portable Plus and (b) The Portable.*

Acknowledgments

The concept and design for the low-power modes was first implemented by The Portable design team. Cliff Cordy designed the hardware capability of dropping to low power and T.W. Cook and John Eaton designed the firmware for powersave and sleep modes. The design was extended on the Portable Plus by Bill Frolik in the keyboard driver, Bob May in the modem sleep routine and the PAM interface, and Carl Lantz in the PPU firmware implementation and system hardware design.

Alesia Duncombe
Development Engineer
Portable Computer Division

this functionality and can help prolong the life of a product.

The Portable Plus was introduced with a RAM drawer and a ROM drawer. The RAM drawer is delivered from the factory with 128K bytes of CMOS static RAM, but is expandable to 384K bytes in 128K-byte increments. From the user's point of view, this additional RAM is the same as the RAM built into the Portable Plus.

The ROM drawer provides a means for the user to add ROM-based software to the computer. This drawer has a storage capacity of 1.5M bytes and supports up to twelve ROMs. Applications software can be designed to execute directly out of ROM or it can be downloaded into RAM before execution, as in typical disc-based packages.

Coupled to the plug-in board space issue is the method of identifying and configuring the plug-in cards. The goal is to make the plug-ins autoconfigurable, so that all the user has to do is install them. The board area limitations shifted this responsibility from hardware to memory management software (see article on page 21). The mainframe provides a sixteen-word region in I/O space for each plug-in. In this area, each plug-in has a card identification register and a configuration register. This allows software to determine system resources and configure them accordingly. To ease hardware and software designs, plug-in RAM increases are constrained to 128K-byte increments.

Self-Test

Manufacturing and serviceability are important issues in the design of any product. In the Portable Plus, much effort was put into the development of a built-in self-test. This is useful in isolating failing components in the factory and the field. It is available to customers and can be used by them to determine whether their unit needs service. It is also useful to dealers and service centers that install op-

tional accessories such as modems, software drawers, and memory drawers. The self-test can be used to verify that the optional hardware is properly installed before giving the unit to the customer.

The tests are accessed by turning the computer off and then pressing the **Shift**, **Extend char**, and **f8** keys simultaneously. This causes the computer to boot into the self-test code instead of the Personal Applications Manager. The test code is part of the firmware stored in ROM in the machine. It consists of a number of different tests and each test checks a different block of circuitry in the unit. These include an LCD test, a timer test, an RS-232-C/V.24 test, a modem test, a ROM test, a RAM test, and a software/memory drawer test.

A displayed menu indicates the tests available and warns the user of the length of time required to run the RAM test and the RAM portion of the software/memory drawer test. The user presses the appropriate softkeys and the **Shift** key to select any single test or all tests in sequence.

When a failure is found, a message is displayed indicating the assembly and reference designator of the failing part. If no failure is found, the message -ok is displayed when the test is completed. Pass/fail messages are also sent to a ThinkJet Printer if one is connected via the HP-IL port. This feature proved useful during the QA evaluation phase of the product's development. Units were placed in an environmental chamber and subjected to extremes of temperature and humidity while running the built-in self-test. Each computer was connected to a printer outside the chamber and the pass/fail information was recorded.

Since the tests are available to customers, they must run without any external test equipment. This requirement diminishes the coverage of the RS-232-C/V.24 and modem tests, since the interfaces to the RS-232-C line and telephone line cannot be checked without some form of plug-in test equipment. For this reason, the built-in tests are not used by HP factory and service facilities to test the RS-232-C port and the optional built-in modem. Instead, another set of tests is used that requires plug-in test equipment and provides a more comprehensive check of these two sections.

Self-test code should require a minimum number of working components in the system to run. Although there are 28 LSI circuits in the Portable Plus, only five are needed to run the built-in self-test.

Mechanical Design

The HP portable computer family is a modular computer family. The main computer is a completely self-contained package with disc drives and printers available in their own separate packages. This allows the portable computer user to reduce the system weight by taking only the components that are needed on a particular trip. Since most of the fragile mechanical devices are excluded from the main system, it can be designed to withstand a much more hostile environment.

Portability and durability were the major design considerations throughout the mechanical development of The Portable and Portable Plus. The printed circuit boards were laid out in parallel with the plastic parts being designed to ensure maximum space efficiency. The system board in The Portable fits into the bottom case, component side down, and the stiffening ribs on the bottom case are placed around the ICs. This cooperation between printed circuit and mechanical design helped keep the product height low. Keeping the product height to a minimum was also a factor in choosing the keyboard design. The final selection uses a low-profile switch that has 75% of the travel of a standard key switch.

The clamshell design not only provides portability, but also protects both the keyboard and the display from the hazards of the transport environment. A carrying case is included instead of a built-in handle. The carrying case has a shoulder strap which is often more convenient than a handle, and the handle on the carrying case is padded for comfort.

Manufacturability, along with mechanical integrity, was a strong consideration in the mechanical design of both products. In The Portable, the printed circuit boards are made to mount into the bottom case, stacked onto hollow studs (see box on page 13) with spacers to keep the leads from shorting. The assembly consists of the bottom case, the system printed circuit board (component side down), a set of spacers, a copper shield for EMI (electromagnetic interference) and ESD (electrostatic discharge) protection, another set of spacers, the I/O board (component side up), and finally a set of nuts to clamp the assembly together.

The display assembly consists of two plastic parts that snap together to form the housing around the display, the arms that come down to the pivot point, two brackets to stiffen the assembly and mount the display, miscellaneous spacers, latches, and springs, and the bezel and bezel cap. This assembly, like the bottom case assembly, is designed with manufacturing in mind. Most components of the assembly are stacked onto studs and then the assembly is held together with nuts. The bezel is held to the assembly by two screws hidden beneath the bezel cap, which is attached to the bezel with a double-sided adhesive foam tape. The cabling that brings power to the LCD is routed through one of the arms before the two plastic parts are snapped together. This eliminates the exposed cable common in many competitive products. The clutches, or friction-restrained hinges, are mounted on the arms at the pivot point and are attached later to the top case assembly.

The top case acts as the "hub" of the product since all parts connect to it. The keyboard is mounted to the top case to avoid alignment problems and eliminate the possibility of keys interfering with the top case surround, causing them to stick. The mounting of the keyboard is somewhat unusual, because it is threaded through the mounting tabs and then rotated into position before being tightened down with eight nuts. This mounting method allows the keyboard to be mounted in the top case without requiring special slides or pulls in the top case mold tooling to allow for the angle of the keyboard.

The display assembly is mounted to the top case by the clutches, which slip over hollow studs similar to those used in the bottom case. Being hollow, these studs allow screws to come from the bottom of the top case to connect to the hinge cover, making that connection invisible on the final product. The top case and bottom case assemblies are then connected together by a rotating motion which engages a latch detail at the front of the unit, electrical con-

Hollow Studs for Package Assembly

Loss of torque has been the subject of many discussions related to the use of ultrasonically installed inserts in plastic parts. The problem occurs because of the tendency of plastic materials to deform slowly, or creep, under force. When a screw is tightened into an insert, the torque causes tension in the screw. This tension is transferred from the insert to the plastic boss into which it is inserted. Most inserts have a holding design of opposing knurls or steps in the body of the insert. These designs do help to hold the insert in, but as the plastic creeps, the insert moves slightly, releasing the tension and thus the clamping force and torque on the screw.

One method used to solve this problem is to add a shoulder or flange to the insert and ensure that the hole in the mating part is smaller than the size of the flange. This way, the insert is supported by the mating part. In most cases, though, the mating part is also plastic, and even if the insert cannot pull out (termed "jacking out"), the wall of the plastic mating part will creep under compressive forces and the result is the same. Another cure for jacking out (and good standard practice) is to install the insert either slightly above or flush with the boss, but never below its surface. This will prevent jacking out of the insert, but as with the added flange, if the mating part is plastic there will be creep and loss of torque.

A different concept is employed in the fastening design of The Portable. Hollow studs were designed to attain metal-to-metal contact wherever torque and clamping loads are critical (see Fig. 1). Both the hollow stud and the associated female insert have flanges at their mating surfaces. The installed height of the hollow stud and female insert is toleranced so that there is a zero clearance or an interference fit. With this design, the tensile stresses are never transferred to the plastic. Hence, creep does not come into play and there can be no jacking out of the insert or compressive relaxation.

The hollow stud is designed to allow internal clearance of an M2.5 screw and the stud's external threads are M6. The outer diameter of the stud seems quite large for so small a product,

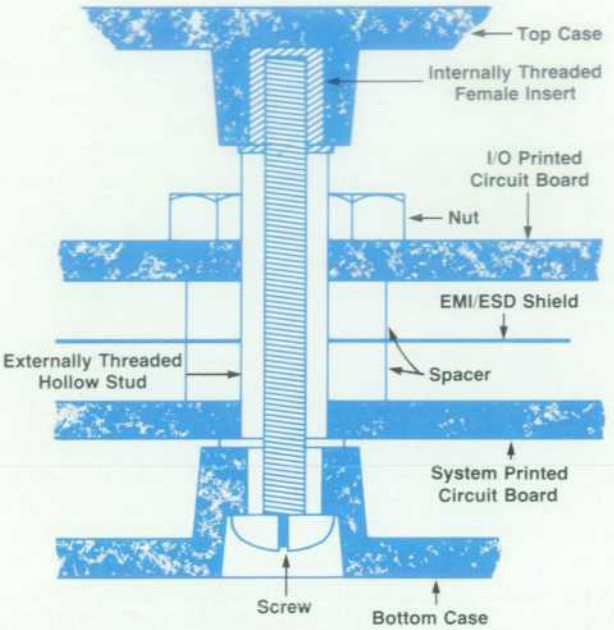


Fig. 1. *Hollow stud fastener design used in HP's portable computers to avoid transferring tensile stresses in fasteners to the plastic surrounding the fasteners.*

but considering the space taken on the printed circuit board, one larger hole requires less room than the two smaller holes that would have been required, one to hold the boards and the other to assemble the top case to the bottom case.

This stud design accomplishes more than one function. It makes a good mechanical connection between the top and bottom cases and also provides support and mounting for the printed circuit boards in a space-efficient manner.

nections are made, and the unit is completely closed by inserting and tightening screws from the bottom case to the top case via the hollow studs.

Durability

Both The Portable and the Portable Plus are tested to HP's Class A2 specifications. Several design iterations and close cooperation with the clutch manufacturer resulted in clutches that remain within the specified torque range for 30,000 cycles. The key switches were cycled 10 million times and the latch and spring assemblies were cycled 50,000 times. The units withstood step drops, on all six faces, at six-inch increments up to 24 inches. Some units were still functional after drops up to 42 inches, but the abuse they had seen was cosmetically quite evident. The main problem in the Portable Plus was breakage of its LCD. The bigger display size in the Portable Plus posed problems not seen in The Portable. The support of the display was improved by padding and cushioning added by the vendor within the display itself and additional cushions and bumpers added to the assembly on our production line. These changes made surviving the required drop height of 24 inches attainable.

Acknowledgments

Eric Gruenberg designed the bottom case, display back and mounting, bezel and its cap, latches, battery door, and keyboard on The Portable.

For the Portable Plus, Charlie Gibson designed the ROM drawer and modified the bottom case before his promotion to manager. Eric took over his responsibilities after rotating back from production.

Reference

1. R.D. Quick and S.L. Harper, "HP-IL: A Low-Cost Digital Interface for Portable Applications," *Hewlett-Packard Journal*, Vol. 34, no. 1, January 1983.

I/O and Data Communications in Portable Computers

by Andrew W. Davidson and Harold B. Noyes

ONE OF THE MAJOR FEATURES of The Portable and the Portable Plus Computers is the extensive input/output capabilities that are built into each machine: specifically, the RS-232-C/V.24, HP-IL (Hewlett-Packard Interface Loop[1]), and modem interfaces.

Why I/O?

In the world of portable computing, it is assumed that people would want to carry the capabilities of a desktop personal computer with them to most places they go—be it home, office, car, bus, airplane, business trip, or pleasure cruise. They would create documents, access data bases, comunicate with the home office, and exchange information (files) with other computer users. It is reasonable to assume that the users would occasionally be away from ac power sources for a significant length of time (up to eight hours for a transatlantic flight or up to 18 hours for a transpacific flight) and that they would want the machine to run on batteries during those times. If a portable computer does not allow its user to operate in this manner, the machine becomes less attractive, especially when a portable computer is usually more expensive, feature for feature, than a comparable desktop unit.

The question thus changes from "Why I/O?" to "What I/O can be included considering battery life, small size, and the technologies currently available?"

I/O Selection

As a result of these types of issues, the selection of the types of I/O included in HP's portable computer family was based on four major factors: power, industry standards, physical size, and ease of use.

Power. The power consumption of the system was a major design concern and some of the major consumers of the system power are the I/O interfaces. For systems based on CMOS ICs, the power consumed by the system is directly proportional to the speed of the system. When applied to I/O interfaces, this means that the faster the interface operates, the more power it requires. Because of this relationship between speed and power consumption, several attractive interfaces had to be eliminated as candidates. These included the HP-IB (IEEE 488/IEC 625), Ethernet, and other local area networks.

Industry standards. By definition, a portable computer is going to be moving around, and consequently is likely to encounter a variety of different computers and peripherals that it needs to talk to. The ability to communicate via industry standards is a must.

Physical size. The word portable has become a computer industry buzzword, the definition of which has become very cloudy. The assumption made at HP's Portable Computer Division is that a portable computer user would want to carry the machine and at least one other bag while running through Chicago's O'Hare International Airport to catch the next flight. Even more important, the user shouldn't have to go into endurance training to do it (at least not because of the portable computer). Hence, the physical size and weight of a portable computer become very important design parameters. The space required for the I/O interfaces (printed circuit board space as well as the space needed to bring the I/O connector to the outside world) is a significant factor.

Ease of use. People who travel tend to be in a hurry. If they need special tools and/or lots of time to connect or disconnect the I/O interface, it becomes a significant inconvenience. All of the interfaces chosen for The Portable and The Portable Plus can be connected and running or disconnected and put away with very little time and effort.

The Portable Modem

There are two main types of applications for The Portable's modem. One is subscription to the various dial-up personal or financial computer services that are becoming more available. The other is remote communication with host computers or other personal computers. In both of these applications The Portable communicates with a modem at the other end that is generally not made by Hewlett-Packard. The Portable's modem must therefore be able to communicate with industry standard modems. In the U.S.A., these standards are Bell 103 (300 baud) and Bell 212 (300/1200 baud). The Bell 212 standard is preferred over the Bell 103 standard because of its higher data rate. Other required features are that the modem be usable over the switched telephone network, and that tone/pulse autodial and autoanswer features be provided.

Space constraints required that the modem design be

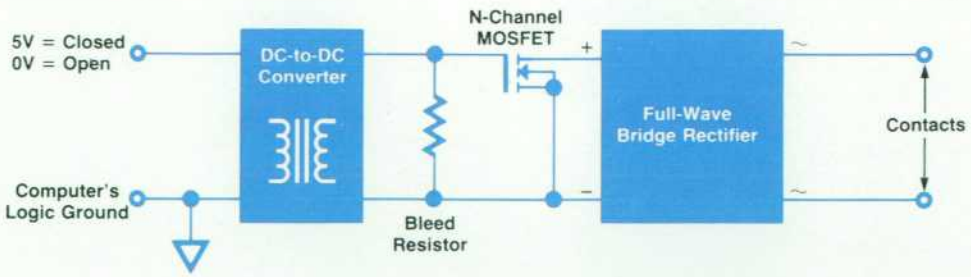


Fig. 1. *Electronic relay used in The Portable's modem.*

centered around a modem/filter chip set purchased from an outside vendor. At the time The Portable's modem was designed, a Bell 212 chip set was not available, so Bell 103 was selected. To avoid adding the space required by a dc-to-dc converter, a design that requires only a single 5Vdc supply is used. This allows the modem circuit to be powered from the existing 5Vdc regulator that supplies the computer's digital circuits.

Although the modem/filter chip set can operate from a single 5Vdc supply, the analog signals within the modem must be referenced to an analog ground voltage of 2.5Vdc. This voltage level is generated by the modem/filter chip set, and provided on an output pin. This eliminates the need for adding discrete circuitry to generate 2.5Vdc.

The single 5Vdc supply influences the design of the transmit section of the modem. A modem that provides DTMF (tone) dialing capability must be able to generate two tone signals of sufficient amplitude simultaneously at the telephone network interface. The assumed load impedance of the network is 600 ohms, and the distortion permitted in the transmitted tones is limited. Typical transmit levels for the two signals are −5 dBm and −7 dBm into the 600Ω network. These levels correspond to voltages of 1.23V p-p and 0.98V p-p. The two tones are of different frequency, so the combined waveform swings up to 1.23 + 0.98 = 2.21V p-p across the 600Ω network.

A typical modem is designed with a 600Ω output impedance. Hence, such a modem would have to generate 2×2.21V p-p or 4.42V p-p at the transmit operational amplifier. This is difficult to achieve with an op amp powered only by 5Vdc such as the transmit op amp used in The Portable's modem design. Without sufficient voltage swing, limiting occurs and distortion is added to the DTMF signals. To avoid distortion, The Portable's modem uses a lower output impedance, roughly 475 ohms, and transmits the DTMF signals at lower levels (still within specifications).

The chip set selected to implement the 300-baud modem and all digital circuitry within the computer is CMOS. This helps keep power consumption low. Some of the analog circuitry within the modem is of the bipolar technology, however. Where possible, low-power bipolar circuits are used and impedances are kept as high as possible. The total current used by the modem circuit is typically 10 to 15 mA.

Since some users seldom use the modem, a firmware-actuated power switch is included in the modem design. This allows the modem to be turned off even when the rest of the computer is on. The switch is implemented by a pnp transistor connected in series with the modem's 5Vdc supply rail. When the switch is off, the 5Vdc supply is disconnected from the modem, all node voltages within the modem decay to zero (ground), and power consumption of the modem is reduced to zero. The inclusion of this switch adds some complexity to the modem design, however. Some of the digital inputs to the modem would normally be in the logic one state (5V) while the modem is not being used. These inputs are forced to logic zero (0V) by additional logic gating and control lines when the power switch is off. If not logically forced to 0V, these input lines would actually supply power to the modem circuit through the input protection diodes of some of the modem chips, and power consumption would not drop off to zero.

All modems that make direct connection to the telephone network must include the functionality of a hook switch. As in a standard telephone, the hook switch is closed to indicate that the modem is using the line to which it is connected, and opened to indicate nonuse. Many modems use a mechanical relay to perform this function. A mechanical relay, however, can require 10 to 100 mA at 5Vdc to keep its contacts in the closed position. This would be up to 10 times the typical power required by the rest of the modem.

To avoid the power dissipation of a mechanical relay, an electronic relay (Fig. 1) was designed for The Portable's modem. This circuit consists of a full-wave bridge rectifier, an n-channel MOSFET, a bleed resistor, and an isolating dc-to-dc converter. The MOSFET and bridge rectifier act together as the contacts of the relay. When a suitable voltage is applied to its gate, the MOSFET turns on (contacts closed). When 0V is applied to the gate, the MOSFET turns off (contacts opened). The dc-to-dc converter, under the logic control of the computer, generates the gate voltage required to turn on the MOSFET and maintains the voltage isolation between the computer and the relay contacts as required by the Federal Communications Commission

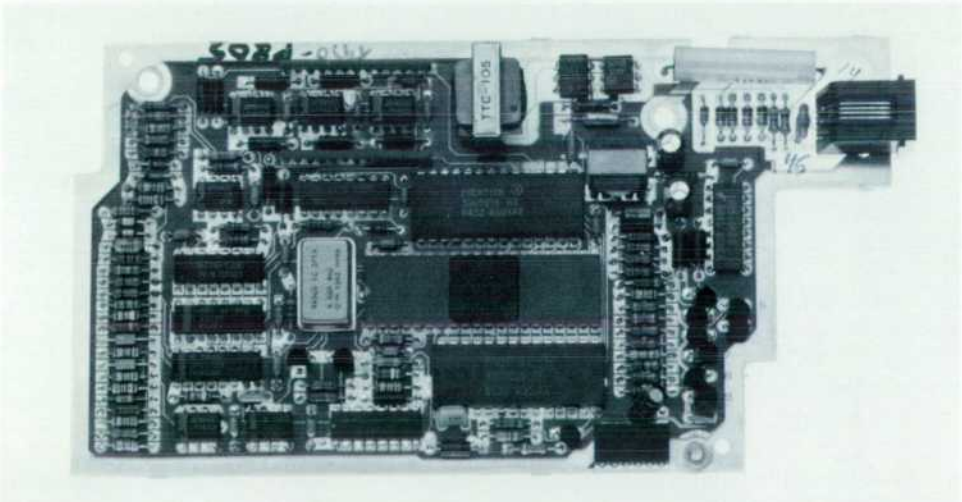


Fig. 2. *Optional HP 82983A Modem for the Portable Plus Computer.*

(U.S.A.). The computer disables operation of the dc-to-dc converter to turn off the MOSFET. The bleed resistor is included to turn off the MOSFET in a sufficiently short time. This electronic relay design requires much less power (100 μA at 5Vdc) than a mechanical relay, because of the extremely low leakage current of the gate of the MOSFET and the high impedance of the bleed resistor.

Another design choice is direct or acoustic connection to the telephone network. Direct connection through an RJ11 modular jack is best for ease of connection and disconnection. Some sacrifice is associated with this choice, because direct connection is prohibited at pay telephones by the Federal Communications Commission (U.S.A.) and some hotel telephones do not use modular telephone jacks. Acoustic connection, through cups which fit over the mouthpiece and earpiece of a telephone receiver, is generally possible with all standard telephones. However, acoustic connection is sensitive to receiver seating within the cups and to ambient noise and jarring. Direct connection does not have these problems. Also, an RJ11 jack is much smaller than two acoustic cups. The Portable's modem uses direct connection.

Portable Plus Modem

While the Bell 103 modem offered with The Portable was the best that could be done at the time, many users suggested that it would really benefit them if the modem could be an option and if it could be enhanced to meet the Bell 212A modem specification. The optional modem (Fig. 2) for the Portable Plus accomplishes both goals.

The Bell 212A specification has achieved wide support as the *de facto* standard for 1200-baud communication over the telephone network in the U.S.A. and Canada. The major data base services support the specification as well as all of the major American modem manufacturers. Included as part of the Bell 212A specification is the ability to fall back to 300-baud transmission using the Bell 103 standard.

The technology available to implement the Bell 212A specification progressed significantly during the time The Portable was being developed. Shortly after The Portable was introduced, the first true CMOS Bell 212A chip set became available. Consisting of two custom CMOS ICs, a CMOS microprocessor, and a handful of discrete components, the modem developed by a third-party company became the first Bell 212A modem that had the potential of filling the requirements of a portable computer. Working with this company to modify their design so that it would better fit in the Portable Plus, we implemented a modem that reduced the printed circuit board space needed for a Bell 212A modem from approximately 80 square inches to approximately 20 square inches. The custom CMOS chip set also reduces the power requirements from approximately two watts to approximately 0.25W.

Circuits were also implemented that allow the entire modem circuitry to be turned off when the modem is not in use, helping to maintain the overall system battery life at acceptable levels. This was accomplished by using a power supply switch similar to that used for the Portable's modem.

Direct connection through RJ11 modular phone plugs (with acoustic couplers as an option) was maintained from The Portable. In addition, an intelligent modem was implemented. This allows the user to communicate with and control the modem by using a set of commands. These commands are compatible with the Hayes Smartmodem™ 1200 command set. This allows the many people that have had previous experience with the Hayes command set to feel at home when using the Portable Plus modem.

Serial Interface

The RS-232-C/V.24 interface is included in both machines because of the great variety of computers and peripherals that can communicate over this type of interface. Terminals, printers, plotters, and modems are among the many devices that can be addressed. In addition, when The Portable or the Portable Plus is emulating a terminal, this interface can be used to communicate with a number of host computers.

RS-232-C has long been a recognized standard for the U.S., Canada, and Europe. It is also a *de facto* standard in many other parts of the world. As such, it is an extremely versatile interface to add to a portable computer.

Because of limitations in fitting everything in the package for The Portable and Portable Plus, the connector chosen for the RS-232-C interface is not the 25-pin D-subminiature connector commonly used for RS-232-C devices. Instead, a 9-pin D-subminiature connector is used (Fig. 3). This prevented the implementation of some of the most esoteric signals described in the RS-232-C documents. However, the most common signals (including DCD, RTS, CTS, DTR, RING, and DSR) are implemented. To minimize the effect of having a 9-pin connector instead of the more common 25-pin connector, custom cables that implement the most common RS-232-C configurations are available. These cables also have thumbscrews to facilitate the attachment of the cable to the computer.

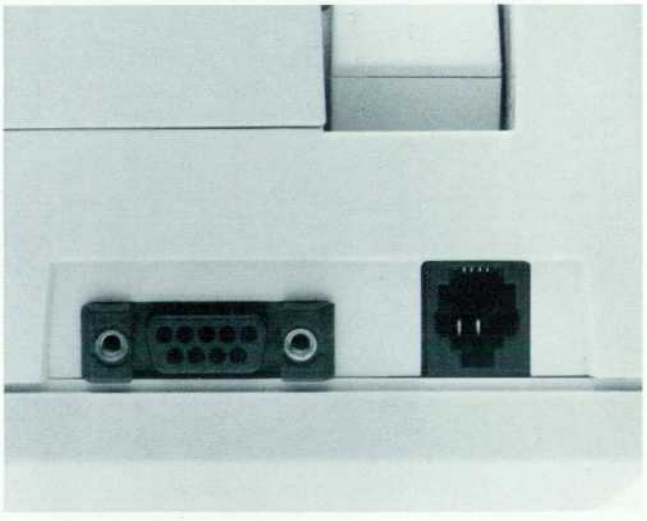


Fig. 3. *The serial interface port (left) for The Portable and Portable Plus Computers uses a 9-pin D-subminiature connector instead of the standard 25-pin RS-232-C/V.24 connector, because of space limitations. Most common RS-232-C signals are present and custom cables are available for common configurations. Direct connection to the built-in modem is made using an RJ11 connector (right).*

To minimize the power required, the entire interface can be turned off when not in use and only drives the output signals to approximately ±7V (still meeting the specification) instead of the more common ±12V.

HP-IL

HP-IL stands for Hewlett-Packard Interface Loop, an interface designed by HP specifically for data communication between small, low-powered devices.[1] This interface is included in both the Portable and Portable Plus designs for easy connection to other devices. Although HP-IL cannot be considered an industry standard, it is a company standard, and it is designed and documented so that electronic manufacturers can easily develop products that use the interface. HP-IL is modeled after the Hewlett-Packard Interface Bus (HP-IB/IEEE 488/IEC 625), an industry standard.

As an HP company standard, HP-IL fits well within the requirements of data communication in portable computers. Through a single pair of connectors on its back panel, portable computers such as The Portable and Portable Plus can have access to flexible disc drives, printers, other computers, and many other devices. HP-IL allows groups of these devices to be connected simultaneously in a single loop, so that access to a device does not require a change of interconnect cables. Some interfaces, such as RS-232-C, allow connection of a computer to only a single device at a time.

The availability of a custom integrated circuit[2] designed by Hewlett-Packard makes the HP-IL a compact interface. This CMOS IC, part no. 1LB3-0003, performs many HP-IL functions automatically. It easily interfaces through an 8-bit bus to a microprocessor system. In the case of The Portable and the Portable Plus, the 8086 microprocessor system around which both computers are designed provides the necessary microprocessor support for the HP-IL. The only added circuitry required for the HP-IL is the 28-pin CMOS IC and some discrete components.

The CMOS HP-IL integrated circuit maximum power consumption is 5 mA at 5.5Vdc. Some of the IC's automatic HP-IL functions make it possible to reduce power consumption even further by reducing microprocessor system activity.

A group of devices connected via the HP-IL forms a loop structure. Each device has two connectors on its back panel, a female **IN** port and a male **OUT** port (Fig. 4). The **OUT** port of one device is connected to the **IN** port of the next device on the loop. What makes this system so easy to connect is that the user merely takes a device and an HP-IL cable, plugs either end of the cable into the port on the device that will accept it, takes the other end of the cable, and plugs it into the next device in whichever port will accept it. The cables are thin and pliable, so they are easily routed on a desk and take up little space. Cables typical of other computer interfaces are often thick and stiff, sometimes so much that moving the cable causes the computer to move. HP-IL cables require no retention screws and they can be chained together to give longer separation distances between devices.

Data always passes in the same direction on the loop, from the source device to the destination device. Depending on the positioning of devices, data must often pass through a device that is neither source nor destination. Such devices know that they are not destinations, so they merely pass any data they receive on to the next device. When the destination device receives data it is expecting, it retransmits the data back on the loop, continuing in the same direction. The data eventually makes it back to the source device. In this way, the source device can do error checking on data passed around the entire loop and knows whether the destination device receives correct data. All data transmitted on the loop, then, makes a complete trip around, no matter where the source and destination devices are.

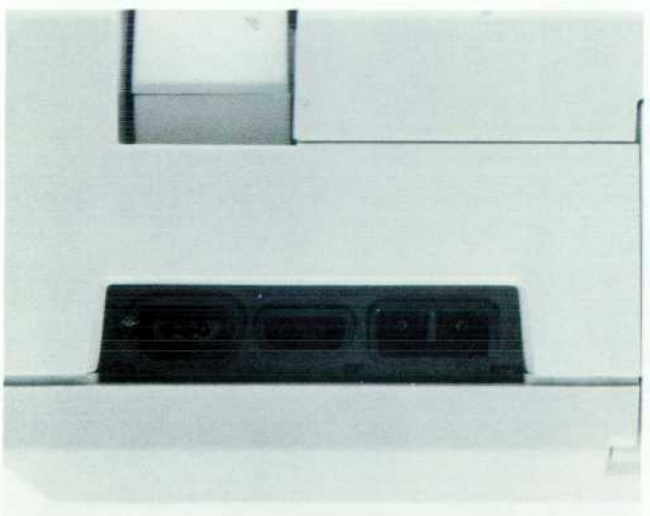


Fig. 4. *HP-IL ports (left and center) on The Portable and Portable Plus Computers. Also shown is the receptacle (right) for the ac adapter.*

Acknowledgments

Cliff Cordy designed the electronic relay used in The Portable's modem. Firmware to support I/O devices and data communication in The Portable was written by T. W. Cook, Bill Hunt, and Charlie Amacher. Firmware for the same purpose in the Portable Plus was developed by Bob May, Bill Hunt, and Bill Frolik. Don Chillrud designed the power supply for the optional modem in the Portable Plus. Joe Fazzio obtained regulatory agency approvals for both computers.

References

1. R.D. Quick and S.L. Harper, "HP-IL: A Low-Cost Digital Interface for Portable Applications," *Hewlett-Packard Journal*, Vol. 34, no. 1, January 1983.
2. S.L. Harper, "A CMOS Integrated Circuit for the HP-IL Interface," *ibid.*

Personal Applications Manager for HP Portable Computers

by Robert B. May and Alesia Duncombe

THE PERSONAL APPLICATIONS MANAGER (PAM) on The Portable and Portable Plus Computers evolved from the original PAM for the HP 150 Touchscreen Computer. PAM is designed to provide a novice computer user access to most of the features of MS™-DOS without forcing the user to learn all of its various commands. PAM helps a user become productive immediately by presenting a simple display that shows which applications are currently available in the system, and providing a simple, one-key method for invoking any of those programs. PAM also provides a file manager, which can perform several file maintenance functions, such as formatting discs, creating directories, and deleting, copying, and renaming files.

One of the most important functions of PAM is the control of the system environment that it provides to the user. Using PAM, variables such as the size of system RAM, the current font, and the printer and data communications interfaces can be set and changed according to the needs of the user, in a manner that is transparent to an application program. This increases the flexibility of the system, and enhances application software by releasing it from many of the system and device-specific details that can tie it to only one system configuration.

User Interface

In keeping with the interface developed for the HP 150, PAM in The Portable and Portable Plus displays the available application programs as a series of inverse video blocks on the screen (Fig. 1). The blocks consist of two lines, each 14 characters long. One line displays the letter identifier of the disc drive on which that application can be found. The top line of each block contains a label identifying the application to the user. The text of this label does not have to be the actual name of the program file that is invoked by MS-DOS, which can be rather complicated. Instead, it can be a word or phrase that is relevant to the user. For example, one of our ROM products is the PC2622 Terminal Emulator, which can emulate an HP 2622 terminal or a VT100 terminal made by Digital Equipment Corporation. To run this program on an MS-DOS system without PAM, you must type PC2622 /H to start it in HP mode. On the PAM screen, however, the user sees a block labeled HP Terminal, and can select it simply by moving the pointer to the block and pressing the Start Application softkey. In addition, a company that has its employees use a standard spreadsheet program and a common template can hide the command used to start that program under a PAM block simply labeled Spreadsheet.

Unlike the Touchscreen version of PAM, applications do not have to be explicitly installed by a utility program to appear on the PAM screen and be easily available to the user. On The Portable and Portable Plus, all disc drives connected to the computers (including the internal RAM and ROM discs) are searched for a text file called PAM.MNU. This file defines the text of blocks that appear on the screen as well as the commands that are issued to MS-DOS when a block is selected. This mechanism allows a piece of software on a disc to become immediately accessible from PAM by simply placing the disc into a drive and having PAM read the disc.

The Portable Plus accepts plug-in ROM modules that contain application programs. Each software module appears as part of the internal ROM disc to the system, and can use the PAM.MNU feature to make its programs accessible through PAM. Since the ROM disc is always present in the system, any application that is installed in this manner is permanently accessible from the main command screen. This provides a powerful customization capability—a company can place its application programs into ROMs and distribute them to its personnel.

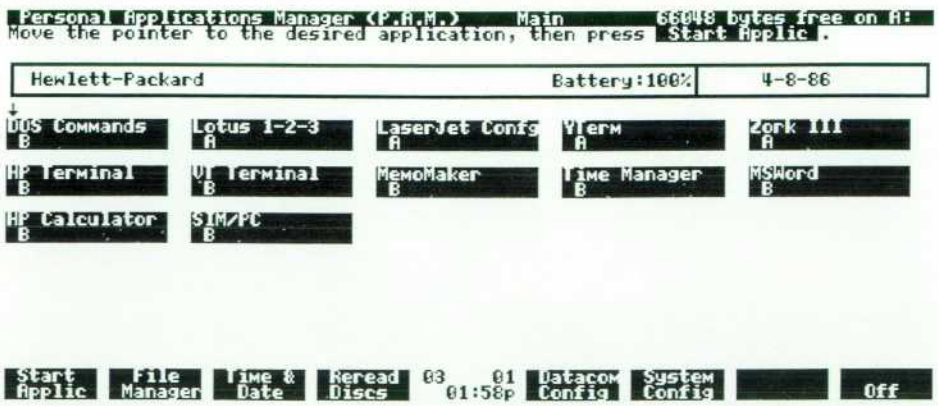


Fig. 1. *The PAM display screen for The Portable and Portable Plus Computers shows the available applications as a series of inverse video blocks.*

System Configuration

Another important feature of PAM is the ability to configure various parameters of the system. There are three main areas that can be modified through PAM: the clock, the data communications parameters, and the system itself, which encompasses such details as the partition between system RAM and RAM disc, control of the power saving features, and which physical peripheral devices are associated with several MS-DOS logical device names.

Through the PAM system configuration screen (see Fig. 2), the user can control many aspects of The Portable and Portable Plus. One of the most important fields is the Memory/Edisc partition adjustment, which allows the user to vary the size of the internal RAM disc. Because of the ability of the Portable Plus to accept additional RAM in its expansion slots, this adjustment allows maximum use of the RAM disc to make it as useful as any external disc drive. For example, a user who only needs one or two applications, but who uses several data files, might want to use the largest possible RAM disc to carry the data files in the computer without needing an external disc. Another user who uses a very large spreadsheet might want the maximum system memory, with only enough RAM disc to hold the stored form of the spreadsheet model.

Since battery life is very important on a portable machine, HP's portable computers have several built-in power saving features (see box on page 10). PAM allows the user to modify two of these features: the display time-out and powersave mode. Powersave mode is a state where the system processor halts whenever the keyboard is idle for a period of time. Most of the time, this is a desirable feature, since it reduces system power requirements by about 44%. For some applications, however, this mode can cause program execution to be sporadic. Through PAM, this feature can be turned on or off as needed. The display time-out is a timed function; if no key has been hit in a given number of minutes, the entire machine enters a sleep mode in which the display, keyboard, and data communications ports are turned off, and the machine suspends execution of the current process until it is awakened by an external interrupt. Thus, whenever the machine is left in an idle state for a period of time, it will automatically turn off, but no data will be lost, and no application program will be affected. PAM allows a user to select the length of time that will elapse before sleep mode is entered. This interval can be anywhere from 30 seconds to 20 minutes, and it can also be disabled.

The keyboard and display can be placed in normal HP mode or a special alternate mode. In HP mode, the display uses the standard HP Roman8 font, and the function keys generate standard HP terminal escape sequences. When the alternate mode is selected, the display uses the IBM PC font and the keys generate IBM-style function codes. Through PAM, the user can select which mode the computer will operate in. This selection is transparent to the application, and allows many existing IBM-PC-based applications to be run on HP's portable computers without modification.

Another system configuration feature is the ability to assign different output devices to some of the standard MS-DOS logical device names. In MS-DOS, there are several reserved device names. The PRN device is defined to be the system printer, the PLT device is associated with a plotter, and the AUX device is defined as the primary data communications interface. PAM allows the user to assign these device names to different pieces of hardware, and to change the definition in a manner that is invisible to a program. For instance, if a word processor is written so that it always writes to the PRN device, a user can print drafts on an HP-IL ThinkJet Printer until the text is satisfactory. The user can then exit to PAM, change the printer interface definition, and run the program again, this time printing a final copy of the document on an HP LaserJet Printer connected to the built-in RS-232-C/V.24 port. The same redirection can also apply to a communications program. If it is designed to use the AUX device for data exchange, it can run through either the built-in serial port, the internal modem, or the HP 82164A HP-IL/RS-232-C Converter. This allows a user to access several different peripherals without the need for a specialized program customized for each particular device.

Datacom Configuration

Data communications over RS-232-C/V.24 or a modem usually involves setting several parameters in the interface, such as baud rate, number of stop bits, word length, etc. Often there is no simple way to change these parameters, which is often required when several different systems or devices are being accessed. PAM contains a data communications configuration menu (Fig. 3) that allows the user to

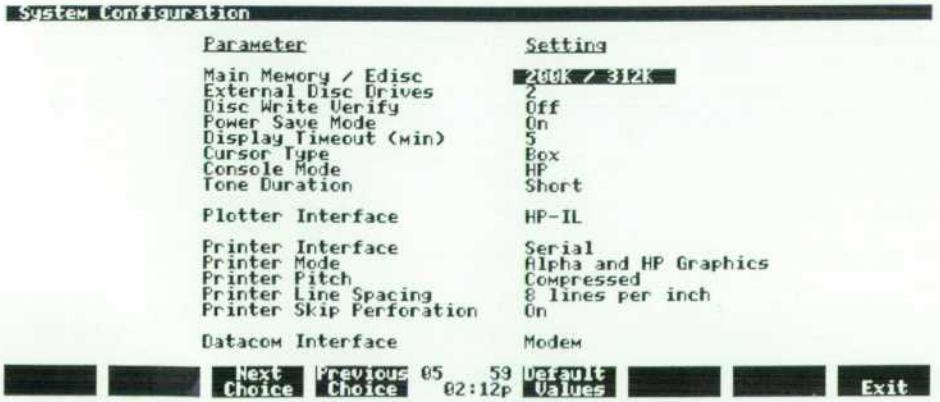


Fig. 2. *PAM system configuration screen on the Portable Plus.*

vary the settings of the built-in serial port, the internal modem, and an external HP 82164A HP-IL/RS-232-C Converter quickly and simply. In addition, in keeping with the need to conserve power on portable computers, the datacom menu shows the current power state of the internal RS-232-C interface and the modem. When these interfaces are turned on, battery consumption is increased. From this menu, the user can see if an application has left the power on to one of these devices, and can turn it off by simply selecting the Off menu choice in the power field for the appropriate interface.

Clock Configuration

The system clock can be set through the clock configuration menu (Fig. 4). In addition to the standard hours, minutes, seconds, days, months, and years, the clock in HP's portable computers allows the time zone to be set. This provides a simple method of keeping the system clock current when traveling across the U.S.A. or abroad.

PAM and Alarms

The PPU (peripheral processor unit) chip in The Portable and Portable Plus allows the scheduling of interrupts at a given date and time in the future. PAM gives the user a means of accessing this feature, along with the ability to display a message or perform a command when an alarm occurs. When PAM is started, reentered, or used to change the clock, the internal RAM disc is searched for a file named PAM.ALM in the top-level directory. If this file exists, PAM reads it. Each line is of the form: MM/DD/YY HH:SS Text. If the text begins with the > character, the rest of the line is interpreted as an MS-DOS command to be executed when the alarm occurs. Otherwise, the text is displayed.

When an alarm occurs, the system makes a warbling sound that continues for about ten seconds or until a key is pressed. If the alarm occurs while the machine is in PAM, the PAM alarm screen immediately appears. If the alarm has a text message associated with it in the PAM.ALM file, the message is displayed and PAM waits for a key to be pressed before resuming execution. Then any command or program associated with the alarm is executed and PAM is restarted when the command or program terminates.

If the alarm occurs in an application other than PAM, the warbling sound is generated, but no other action is taken until PAM is reentered. When PAM restarts, it notes that an alarm has occurred, and behaves as described above.

If an alarm occurs while the machine is in the sleep state, it resumes normal execution and then proceeds as described above. This feature is very useful for data communications. An alarm can be scheduled late at night when telephone rates are discounted to invoke a program that uses the modem to connect to a data base on a mainframe computer. In this manner, a company can distribute information to its field personnel electronically at a lower cost and a faster rate than more traditional methods.

PAM and Ringing Phones

The internal modem or an external modem connected to the serial port can generate a "ring" interrupt when they are called by another telephone or computer. When this happens, The Portable and Portable Plus make a ringing sound to alert the user that someone is calling the computer. If a ring occurs while PAM is running, PAM attempts to execute a command file named AUTOANSR.BAT. If this file is present on the RAM disc, it is executed. It can do anything a standard MS-DOS batch file is allowed to do. When it terminates, PAM is restarted.

If the ring interrupt occurs when PAM is not running, such as during the execution of an application, the ringing sound occurs, but no other action is taken. If a ring interrupt occurs while the machine is in the sleep state, it wakes up. If it is in PAM, the AUTOANSR.BAT file is executed if present.

Localization

In the initial design of the Portable Plus, it was decided that native language support would be a high priority for the project. Hence, we invested significant effort to ensure that its PAM, which is the "outward face" of the computer, would be as easy as possible to convert from one language to another, with a minimal impact on the internal structure of the program.

PAM on The Portable has all of its messages compiled within the code of the program, at several different points. This made conversion from English a time-consuming task, and meant several different versions of PAM—an English PAM, a German PAM, a French PAM, etc. Our objective for the Portable Plus was to have one version of the program that uses a generic message system that makes the current language of the machine invisible to PAM. To this end, it was decided to place all of the messages, softkey labels,

Fig. 3. *PAM datacom configuration menu on the Portable Plus.*

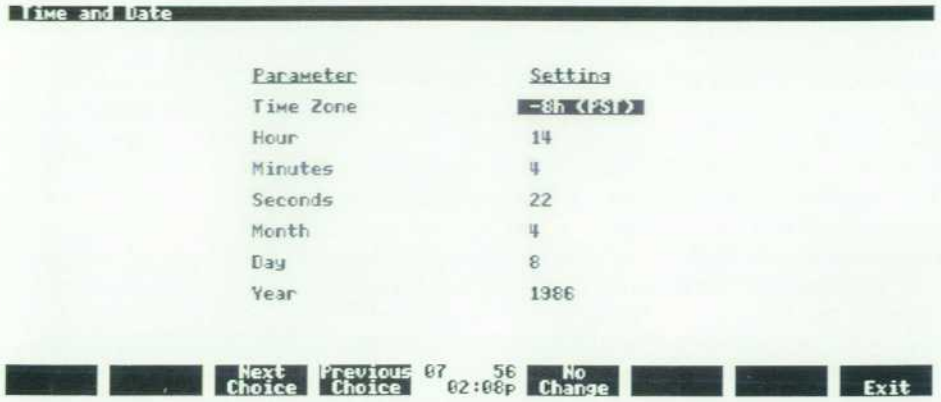


Fig. 4. *PAM clock configuration screen.*

and other text strings into the system configuration ROM.

The configuration ROM is an EPROM (erasable programmable ROM) that is programmed on the assembly line to contain system data and the serial number for each machine. For localization, the EPROM is filled with the PAM messages and the keyboard layout for a specific country. The system software uses the values contained in this ROM when possible. The keyboard driver, for example, defines the keyboard according to the mapping contained in the configuration ROM. When PAM prints a message to the user, it calls a system routine to display a numbered message from the ROM. The system then scans the EPROM for the text that corresponds to the desired number, and shows it on the screen. In this way, PAM deals with abstract message numbers, with the configuration ROM providing the actual text of the message in the proper language. The rest of the built-in software applications on the Portable Plus also use this mechanism to display their messages, making the basic machine fully localizable by simply inserting the appropriate configuration ROM on the manufacturing line.

Acknowledgments

Steve Sakoman did much of the original work on PAM for The Portable. Bill Frolik and Mark Rowe contributed many ideas on design and usability. Rick Bell worked on the configuration ROM, Jim Axtell provided guidance and suggestions, and Karen Oda, Gordon Staley, and George Sachs were the quality assurance engineers on the project.

Memory Management for Portable Computers

by Mark S. Rowe

THE PORTABLE AND PORTABLE PLUS Computers run under MS™-DOS 2.11. This operating system requires a certain amount of contiguous read/write memory (RAM) beginning at physical address 0. This memory is called system memory and is managed by the operating system. The minimum amount of system memory required depends upon what drivers are installed and what applications must run. On The Portable, the system memory size can be set to as little as 96K bytes and on the Portable Plus, to as little as 80K bytes. Both machines have considerably more RAM than this minimum. The Portable has 272K bytes of RAM. The Portable Plus (with plug-in expansion) can have from 128K bytes to 1280K bytes with the current expansion options, and has an upper limit of over two megabytes of RAM.

The RAM that is not assigned to system memory is organized into a RAM disc that appears to the operating system as unit A: and functions identically to a mechanical disc from the system's point of view. Though lacking the capability of off-line archival storage and the ability to read actual discs, the RAM disc has the advantages of being much more durable, consuming much less power, and being very much faster than its mechanical counterpart.

Part of the operating system is a block of code called the BIOS, which is executed when the system is initialized (boot up). This code determines (in addition to many other tasks) the amount of system memory in the machine and executes the program that provides the user interface. As with other MS-DOS machines, the BIOS code resides in read-only memory (ROM) and executes automatically from

a reset condition. However, on both The Portable and the Portable Plus, the user interface program (along with several other system files) is also stored in ROM. These system files are organized in a ROM disc that appears to the system as unit B: and is similar to the RAM disc, except that its contents are predetermined and cannot be overwritten. The Portable Plus has the additional capability, through its expansion ports, of allowing ROM-based applications to be added. These ROM-based applications appear as files on the ROM disc from where they can be downloaded to system memory and executed. These applications can also directly execute code out of the plug-in ROM.

Memory Management Code

The operations described above are handled by the memory management code on the Portable Plus. Memory management on The Portable is similar to that of the Portable Plus, but is less complex since it deals with a fixed amount of RAM and has no provision for handling plug-in application ROMs. Within the Portable Plus the memory management code must determine the total amount of RAM in the system, allocate a portion of the total RAM to system memory, maintain the RAM disc including read, write, and formatting functions, identify any plug-in ROMs and maintain the ROM disc, and provide utility functions to allow applications to execute out of ROM and directly access a ROM's contents.

When the system comes up, the memory management code must determine how much RAM is in the machine. The Portable Plus has 128K, 256K, or 512K bytes of contiguous built-in RAM beginning at address 0 in the memory address space. In addition, there are four logical plug-in ports (two in each of the two plug-in drawers). Each drawer can support up to eight banks of RAM (each bank contains 128K bytes). Thus, there can be up to two megabytes of plug-in RAM in addition to the built-in RAM. (However, current physical constraints on the drawer size and memory density limit this to 768K bytes in the plug-in drawers.)

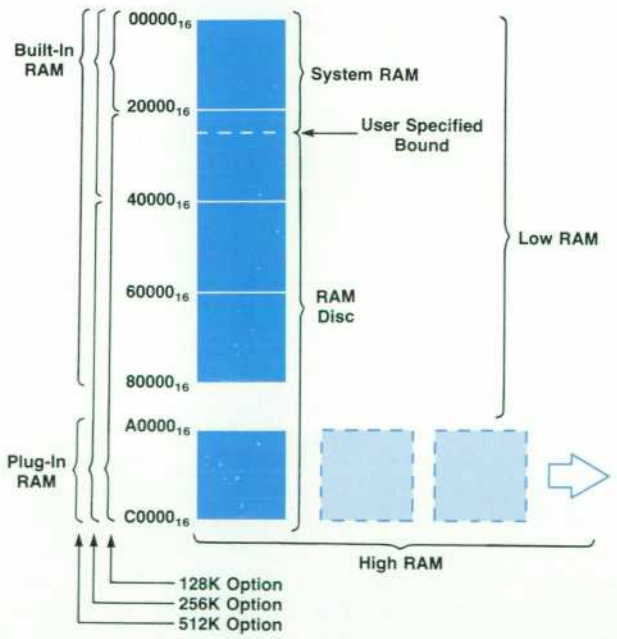


Fig. 1. *Physical RAM organization. The three built-in RAM options for the Portable Plus are 128K, 256K, and 512K bytes.*

Each of the plug-in RAM banks can be independently enabled and disabled. In addition, each bank can be independently mapped into any of the eight 128K-byte partitions of the one-megabyte physical address space (unlike the built-in memory which is always enabled and fixed in the address space beginning at address 0). Enabling a RAM bank so that it overlays built-in RAM, the display RAM, another already enabled plug-in RAM bank, built-in ROM, or an enabled plug-in ROM bank would cause it to function improperly. As banks of plug-in RAM are identified, they are assigned to consecutive 128K partitions of the address space, beginning at the end of the built-in RAM until display RAM is reached at location 80000_{16} in the address space. This RAM in the memory space below 80000_{16} is referred to as low RAM and includes all banks of built-in RAM and the first banks of plug-in RAM to a total of four banks (512K bytes). Any additional plug-in RAM is referred to as high RAM and is assigned to the 128K-byte address partition beginning at address $A0000_{16}$. Only one bank of high RAM can be enabled at any time. Furthermore, since plug-in ROMs occupy a 256K-byte partition beginning at address 90000_{16}, a bank of high RAM cannot be enabled concurrently with plug-in ROM. Refer to Fig. 1 for a diagram of the physical RAM organization.

Since system memory must reside in contiguous address space, only the low RAM is available for use as system memory; all high RAM must be part of the RAM disc. Access to the RAM disc is always performed by code that is resident in the system ROM and data transfers are always to or from a buffer in system memory. Therefore, it will never be necessary to have two banks of high RAM enabled simultaneously, nor will a plug-in ROM ever need to be enabled during a RAM disc access.

Once all of the plug-in RAM has been identified and assigned to a memory partition in either low RAM or high RAM, the Portable Plus memory management code determines the amount of system memory, which is subsequently reported to the operating system during boot up. The boundary between the system memory and RAM disc can be set by the user within the constraints that the boundary must be on a 4K-byte boundary, the minimum system memory size is 80K bytes, the minimum RAM disc size is 4K bytes, and the maximum system memory size is 512K bytes. The boundary can be changed without altering the current contents of the RAM disc. The memory management code determines where the last used sector of RAM disc is and will not allow the memory partition to be moved past that point. A utility is provided to allow the used sectors to be packed into lower-numbered unused sectors to free up more space that can be allocated to system memory.

RAM Disc

The sectors of the RAM disc (see Fig. 2) are assigned to physical memory beginning with the first bank of high RAM. Each sector contains 512 bytes of RAM. The end of the RAM disc (highest-numbered sectors) is in low RAM. Within the 512K-byte address space of low RAM, the sectors begin at the high end and proceed toward low memory.

Thus, the highest-numbered sector in the RAM disc is adjacent to the high end of system memory. This allows the boundary between RAM disc and system memory to be moved easily without disturbing the current contents of the RAM disc. The RAM disc is increased by allocating sectors from the high end of system memory, decreasing the size of system memory accordingly. System memory size is increased by allocating memory from the highest numbered sectors of the RAM disc, assuming that these RAM sectors are not in use.

Although the built-in RAM is always enabled, the plug-in RAM is enabled and disabled under software control. RAM banks can contain either system memory, RAM disc memory, or a mix of system memory in the low end of the bank and RAM disc in the high end of the bank. Plug-in RAM banks that contain only RAM disc memory remain disabled except when being accessed by the memory management software. This is a safety precaution that minimizes the risk of a program inadvertently overwriting the RAM disc memory. Since system memory must always be enabled when a program is executing, a bank of memory that contains both the last sector of RAM disc and the high end of system memory is always enabled, leaving the high sectors of the RAM disc vulnerable to program errors. This vulnerability can be avoided by setting the system memory boundary on a boundary of a plug-in RAM bank. In this case there is no RAM bank that contains both system memory and RAM disc sectors. Thus all RAM disc memory is left disabled except when being accessed by the memory management code and cannot be inadvertently overwritten by a faulty program.

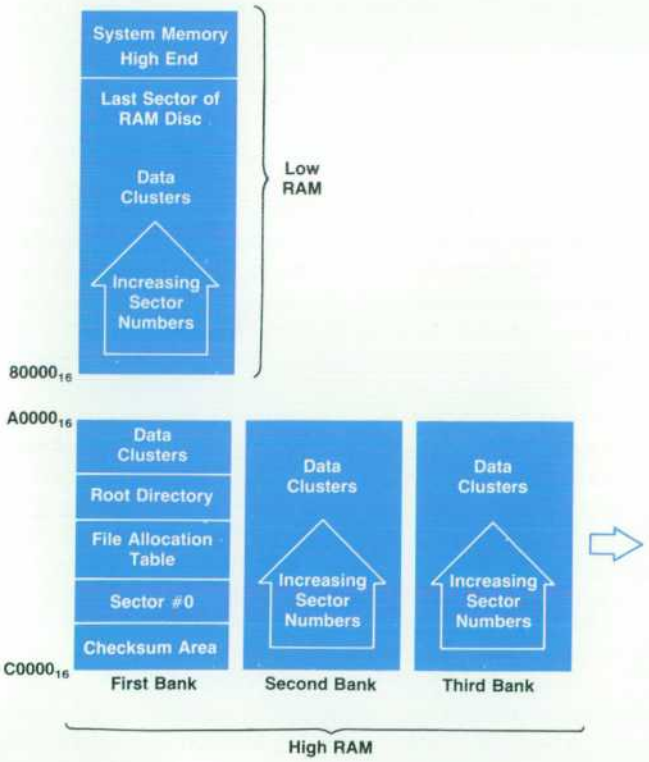


Fig. 2. *RAM disc sector organization.*

The first sector of the RAM disc is a reserved sector (sector 0) that contains a BIOS parameter block in the first 24 bytes, some system information, and checksum data for the first 384 sectors of the RAM disc. Each sector has a one-byte checksum. If additional space is required for checksums, it is allocated in a multiple of 512 bytes above sector 0 in the physical memory space. Enough checksum space is allocated to accommodate the maximum number of sectors that the RAM disc can have, given the total amount of memory in the machine.

The BIOS parameter block defines the structure of the RAM disc as seen by the operating system. Except for the number of sectors on the disc, the structure is fixed, given the amount of RAM in the machine. The sector size is 512 bytes per sector and there is one sector per data cluster. The RAM disc has one reserved sector as discussed earlier, and one file allocation table. There are 64 entries in the root directory. The number of sectors on the disc is set during system configuration. The number of sectors in the file allocation table depends on the memory size. As for the checksum area, a sufficient number of sectors is allocated for the file allocation table to handle the largest RAM disc possible. Therefore, the size of the file allocation table will not change when increasing or decreasing the RAM disc size. Fig. 2 depicts the organization of the sectors within the RAM disc.

When the RAM disc driver is first entered during an I/O operation, it enables any low RAM that may have been disabled, it disables any plug-in ROM to free up the swap space at address $A0000_{16}$, and it enables the first bank of high RAM. Since the checksum space, sector 0, the file allocation table, and the root directory are all at the beginning of the RAM disc and use less than 128K bytes, it is guaranteed that all of these frequently accessed components will be enabled by default. If the operation requires access to a sector in a bank of high RAM other than the first bank, the RAM disc driver will disable the first bank of high RAM, enable the required bank, perform the specified operation, disable the bank, and reenable the first bank of high RAM. After the operation is complete (including updating a checksum value if required), the high RAM bank and any low RAM banks that contain only RAM disc data are disabled. If a plug-in ROM was previously enabled, it is reenabled and the RAM disc driver exits.

The structure and operation of the RAM disc as described above provides a fast, flexible, and reliable scheme for implementing a disc-based operating system on a portable computer without a mechanical disc drive.

ROM Disc

In addition to the ROM-based BIOS software, the Portable Plus has about 100K bytes of system software that is built into a ROM disc. This software appears to the operating system to reside in a subdirectory of the B: disc unit. Furthermore, the expansion ports on the Portable Plus allow the user to plug in application ROMs that will also appear to the operating system as files on subdirectories of the B: disc unit (a distinct subdirectory for each plug-in ROM). Unlike the RAM disc, where each 512-byte sector is represented by 512 bytes of RAM somewhere in the machine, many of the ROM disc sectors do not correspond to any

existing memory and are generated on the fly when requested from the various internal tables and parameters that are set up when the system is initialized.

When a sector of the ROM disc is requested, the ROM disc driver first determines what type of sector is being requested. From the point of view of the ROM disc driver, there are six different types of sectors in the ROM disc (see Fig. 3). The type of sector can be determined from the sector number and each type invokes a unique software routine to extract the appropriate data and construct the sector in the specified buffer. If sector 0 is requested, the ROM disc driver fills the first 24 bytes of the destination buffer with fixed data from a table in ROM (this data includes the BIOS parameter block for the ROM disc). The remaining 488 bytes of the sector are set to zero in the buffer.

If the requested sector number is 1 through $0C_{16}$, then the sector is part of the file allocation table. Sector 1 is special in that it contains the special media byte entry at the start of the file allocation table and includes the table's entries for the sixteen root subdirectory files and the system files in addition to some of the entries in the first plug-in ROM. All of the other file allocation table sectors contain entries for plug-in ROM files. The root subdirectories are each one cluster long so the file allocation table entries are end-of-file entries ($0FFF_{16}$). The entries for the system files are computed from a table in ROM. The file allocation table entries for files in plug-in ROMs are computed from the values in the file allocation table of the plug-in ROM. Since each plug-in ROM is mapped into an explicit 512-sector range of the ROM disc, the file allocation table entries returned by the ROM disc driver are at a known offset from the values present in the plug-in ROMs themselves. From the particular sector requested, the ROM disc driver can determine what range of table entries are required and return the correct values on the fly.

If the requested sector is number $0D_{16}$, then the root directory is being requested. The 512 bytes in the root directory sector contains sixteen 32-byte file directory entries. The first entry is for the BIN subdirectory of the ROM disc, which is always present and provides the path for accessing the system files. Its fixed data is extracted from the system ROM. The remaining fifteen directory entries are for plug-in application ROMs. Each of the fifteen possible plug-in ROM banks defines a subdirectory entry in the root directory of the ROM disc. The data for the plug-in ROM entries is computed from special data within the first sector of the plug-in together with the ordering of the plug-in ROMs in the expansion ports. The root directory sector is created on the fly when accessed.

Sector Number (hexadecimal)	Sector Type
0	Boot Sector
1 ⋮ C	File Allocation Table
D	Root Directory
E ⋮ 2D	Root Subdirectories
2E ⋮ 1E9	Built-In System Files
1EA ⋮ 1FE9	Plug-In ROM Sectors

Fig. 3. *ROM disc sector organization.*

If sector number $0E_{16}$ through $2D_{16}$ is requested, then a sector of one of the root subdirectories is being accessed. Each of these subdirectories is two sectors long. Sectors $0E_{16}$ and $0F_{16}$ are the sectors of the BIN subdirectory. The data for these sectors is computed from a table in the system ROM. The other thirty sectors (10_{16} through $2D_{16}$) are subdirectory sectors for the plug-in ROMs. The data for these sectors is contained in the root directory space of the corresponding plug-in ROM. Since the file entries in the plug-in root directory describe the starting cluster for each file relative to the start of the plug-in ROM, the ROM disc driver must modify each entry as it is copied out to map it relative to the start of the ROM disc. This mapping is dependent upon the ordering of the plug-in ROMs. The only other modification necessary is that subdirectory files contain dummy entries to link to the parent directory and the current directory. Since these entries are not present in the plug-in root directory, they are created on the fly when the directory sector is accessed. The data required for these entries is contained in special fields in sector 0 of the plug-in ROM.

A request for a sector number in the range from $2E_{16}$ to $1E9_{16}$ will access a sector of the system files. The data for these files is present in the system ROM. The address of the requested data is computed from a table also in system ROM.

Requesting a sector number from $1EA_{16}$ to $1FE9_{16}$ accesses a sector from a plug-in ROM. Each of the fifteen possible plug-in ROM banks is allocated 512 sectors of the ROM disc. Thus the first plug-in ROM occupies sectors $1EA_{16}$ through $3E9_{16}$, the second plug-in ROM occupies sectors $3EA_{16}$ through $5E9_{16}$, and so forth. The sector allocation is independent of the actual size of the ROM. Sectors beyond the defined bounds of the plug-in ROM return undefined data. All data in the plug-in ROM memory is mapped into some sector of the ROM disc. This includes the boot sector, the plug-in file allocation table, the root directory, and any ROM-executable code. However, these sectors are in general marked as available (unused) sectors in the file allocation table of the ROM disc and will therefore not be accessed by the operating system since the ROM disc is read-only.

The sectors within a plug-in ROM are part of the file structure of the ROM disc. These sectors belong either to a normal file or to a subdirectory in the plug-in ROM. One constraint on the contents of plug-in ROMs is that all sectors of subdirectory files must occur in the ROM before any of the normal file sectors. The boundary between the subdirectories and the normal files is specified by a special field in sector 0 of the plug-in ROM. This allows the ROM disc driver to determine whether the requested sector is a subdirectory sector or not. This distinction is important

since subdirectory files are composed of file entries that contain a value specifying the starting cluster number. Within the plug-in ROM all starting cluster numbers are specified relative to the start of the ROM itself. When these values are read from the ROM disc they must be mapped into the appropriate cluster number within the plug-in's sector space. Sectors from normal files in a plug-in are transferred without any modification.

This implementation of the ROM disc driver allows efficient use of the system ROM space while allowing the disc-oriented operating system to run unaltered. The implementation also accommodates plug-in ROMs to allow user application files to be incorporated into a ROM or EPROM in a fairly straightforward manner (see article on page 28). The plug-in ROM contents are incorporated into the ROM disc and made accessible to the operating system automatically by the ROM disc driver.

Plug-In ROM

In addition to application files, a plug-in ROM may contain software that is intended to be executed directly out of the ROM rather than being downloaded into system memory first as is required of program files. This feature slightly reduces the time necessary to run a program, but more important, it can greatly reduce the amount of system memory required to execute the application, thus allowing more memory to be used for program data or a larger RAM disc.

Two system routines are provided to allow an application to access code or data in a plug-in ROM directly. The first routine allows a program to enable a plug-in ROM. Once enabled, a program can directly access data in that ROM. The memory management code automatically handles memory swapping when accessing RAM disc or invoking ROM executable code in another plug-in ROM. The second routine allows a program to execute code in plug-in ROMs directly. The routine disables any currently enabled memory bank (plug-in ROM or high RAM), enables the specified plug-in ROM, and transfers control to the specified target location (relative to the start of the ROM). A return instruction will then exit back through the memory management code which then restores the originally enabled memory.

Acknowledgments

Much of the memory management design on the Portable Plus is based on work done for the Portable by T.W. Cook. T.W. was also a valuable technical resource during the early design phase of the memory management software for the Portable Plus. Also contributing to the design were the other members of the Portable Plus software design team: Alesia Duncombe, Bill Frolik, and Bob May.

A Hybrid Solution for a 25-Line LCD Controller

by Glenn J. Adler

UPGRADING TO A 25-LINE liquid-crystal display for the Portable Plus required a redesign of the 16-line controller used in the earlier HP 110 Computer, The Portable. Because of the larger display size, even if the old controller could have been reprogrammed, the refresh rate required by the larger LCD for flicker-free operation could not be met. Hence, the designers decided to do a fast turnaround design which leveraged the architecture of the earlier 16-line custom controller.

The objective of the LCD controller is to regulate screen refresh while allowing the CPU to access screen memory for character placement. To support a full screen of graphics, it is necessary to have more memory than the single 64K-bit static RAM used in The Portable. Therefore, two such RAMs are used in the Portable Plus.

Features

While allowing the CPU access to screen memory without any additional circuitry, the Portable Plus LCD controller regulates screen update independent of the CPU without requiring any additional circuitry in this function. When put in sleep mode, the controller draws a maximum of 10 μA at 3V, while actively retaining the contents of its memory.

With 128K bits of memory the controller supports 1.25 pages of 200×480-pixel graphics displays and 2.5 pages of 25 lines of alphanumeric data, along with three independent programmable fonts.

The most significant feature of this controller is its ability to lock lines of alphanumeric display for softkey definition. This is handled solely in hardware through programming of an internal PLA (programmable logic array). The user need only write the desired number of rows of memory lock into the status register, and the controller will place these rows, starting at the top of RAM, at the bottom of the screen. It is the programmer's responsibility to point the top-of-page around these rows so as not to show redundant information on the screen. This feature makes scrolling a simple algorithm, even when softkeys are being used. All that need be done is to increment the top-of-page register. The display RAM is treated as a continuous cylinder. Point-

ing the top-of-page to the bottom of RAM will wrap the display around back to the top of RAM.

In graphics mode, the graphics bits are acquired in a single fetch. A single PLA cycle acquires screen data in groups of four bytes, one for each quadrant of the display. In the case of alphanumeric characters, the byte is modified by either inversion or blanking, dependent upon the attributes and whether there is a cursor present. In both alpha and graphics modes there are dead periods when the screen RAM is not being accessed. These periods are when the CPU is granted access to display memory. The CPU can asynchronously request access by strobing the CSM line, but the controller will always hold the ready signal low until there is a dead cycle. With a 5-MHz input clock, the time allowed for one RAM read is 400 ns, requiring a RAM with a 150-ns access time.

The display interface consists of four dot outputs and four timing signals. The dot outputs are shifted out serially with their values determined by the data fetched from RAM. The clock signals are generated directly from outputs of the PLA. The dot clock's falling edge latches a bit of data into each of the LCD panel's four shifters. The row clock shifts the value of the panel's shifters down into its column drivers. The frame clock strobes at the start of a new frame, and resets the row scan back to row zero. The fourth clock is a bias signal which changes the polarity of the voltage sent out on the row and column lines. This bias signal is necessary to prevent damaging effects on the liquid-crystal material that might occur if a net dc voltage were allowed to build up. Thus, each time a pixel is refreshed, the bias voltage remains constant, but its polarity is reversed.

The PPU interface is activated by accessing the I/O space belonging to the PPU shifter. The system will generate a CSPPU signal and in the case of a write, will parallel-load a byte of data to be transmitted to the peripheral processor. This write also causes the serial interface to the PPU to be driven high. By polling this pin, the PPU knows data is available and begins to drive the serial shifters' clock pin. When the PPU wants the CPU to read the communication register, an interrupt is generated and a byte is read after being shifted serially out of the PPU.

Packaging and Testing

To minimize the pin count, the controller is placed directly on the CPU's demultiplexed address/data bus. Even with this reduction of pins in the CPU interface, the RAM circuit requires too many additional pins to package the chip in a conventional 48-lead dual-in-line package. Therefore, it was decided in the interest of conserving board space (see Fig. 2) and minimizing small parts count to mount both the controller and the RAM chips inside a ceramic hybrid package. Once this decision was made, pin constraints still dictated that the hybrid be packaged in a 48-lead package, but this left several pins unused. Since the package was already on the CPU bus, a logical use for the additional pins was the interface to the peripheral processing unit (PPU). This serial-load-parallel-out, parallel-load-serial-out shifter was included on the LCD controller chip at the expense of four extra pads.

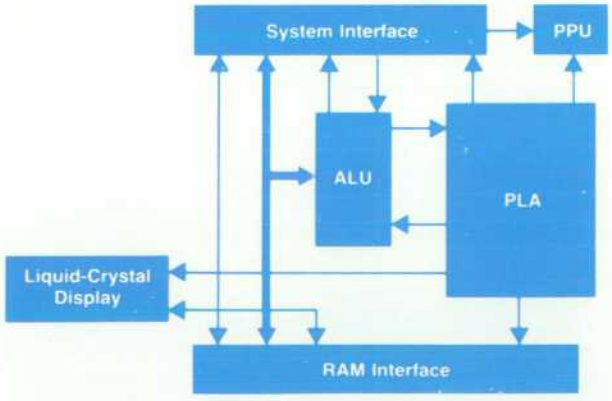


Fig. 1. *Block diagram of Portable Plus LCD controller.*

The LCD controller was laid out on a CAD workstation according to design rules specified for the 3.5-μm CMOS technology fabricated in our in-house integrated circuit facility. Special care was used to ensure that the design was created free of dynamic nodes (except for the PLAs) to minimize the power used by the chip. Logic verification was done using an HP 9000 Computer system and a custom toolset. The test program was constructed directly from test vectors on the logic simulator using a conversion program provided by the tools group. It was necessary to generate two different tests, one for the chip and one for the hybrid, that tested both the controller and the memory without access to many of the controller's pads.

The layout of the ceramic hybrid was closely supervised to reduce the possibility of crosstalk. Chip capacitors are mounted inside the package to decouple the supply of each chip separately and reduce any noise. The package is sealed with a ceramic lid for environmental protection.

The package is dynamically burned-in to screen infant mortality RAM failures, since the chips are purchased only after being functionally tested. The first chips showed only one bug, which was easily remedied through the removal of five rectangles in the PLA. These rectangles were etched off the mask, and parts then appeared to function correctly. Only upon margin rating fast and slow lots on the production tester did a problem arise. Fast parts exhibited a fight on the internal bus. Both these problems were solved with only a two-mask design turnaround.

EMI was a problem with the 16-line controller for The Portable. By minimizing the output to the drive transistors for the LCD clocks and dots, we were able to obtain regulatory agency qualification. Adding a video interface loads these signals more than initially intended, but all parts still pass their margin rating.

Circuit Architecture

A simplified block diagram of the the controller architecture is provided in Fig. 1. The system interface consists of address latches to capture a 16-bit CPU address on the fall of the ALE (address latch enable) signal, read, write, and three chip select strobes, a ready signal, and an input clock running at 5 MHz.

The positioning of the LCD controller directly on the CPU bus eliminates the need for additional buffers in the system, but increases the capacitive loading on an already heavily taxed CMOS 8086 microprocessor. The three chip

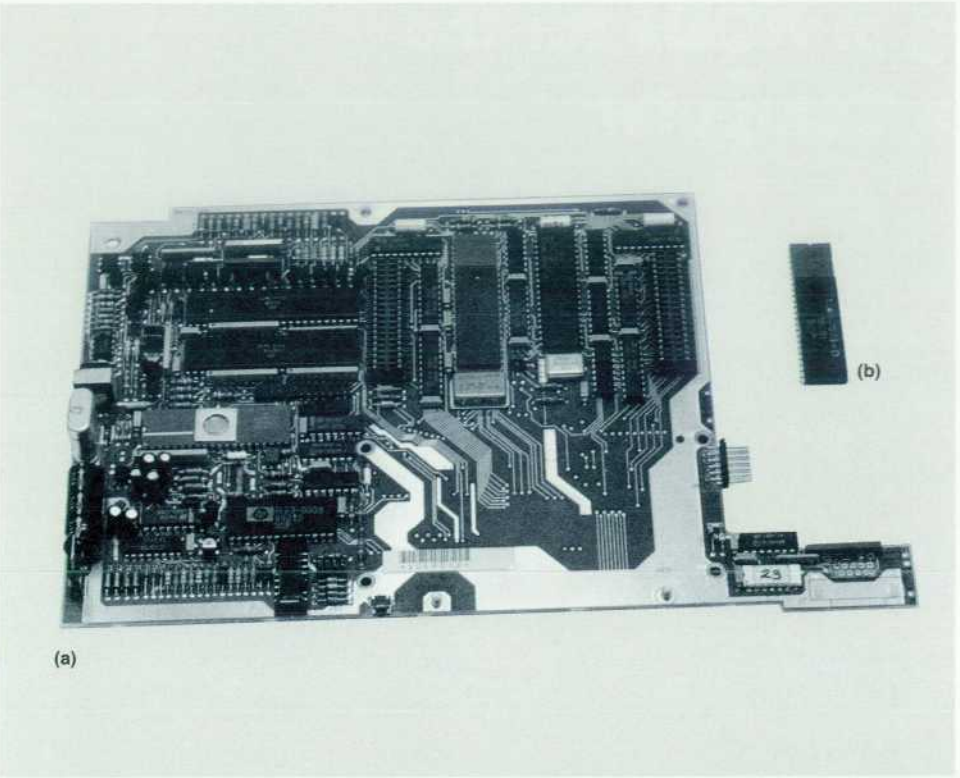


Fig. 2. *Photograph comparing LCD controller board space in The Portable (l) to the size of the 25-line LCD controller hybrid used in the Portable Plus (r).*

selects regulate three different functions of the controller. The selection of memory chip select (CSM) generates an access to display RAM and the activation of CSPPU (PPU chip select) generates an access to the PPU shift register. The PPU (peripheral processor unit) is a separate processor in The Portable and Portable Plus that controls the switches that power the system (see article on page 4).

By decoding the three lowest address bits and sending an I/O chip select (CSIO) to the chip, each of the read/write registers can be accessed. These registers are the A status register, the top-of-page register, and the cursor register. The status register contains bits that dictate the mode in which the controller operates. By setting the appropriate bits in this register, the programmer can control whether the display is blanked, alphanumeric, or graphics, and select the type of cursor (box or underline) and the number of lines locked at the bottom of the screen (0, 1, 2, or 3 softkey rows).

Two internal PLAs control the contents of two ALU registers: the top character address (TCA) and bottom character address (BCA) registers. Upon reset the ALU fetches the top character address from the TCA register and, using a hardware subtractor, generates subsequent addresses for screen fetches. The BCA register is used in alphanumeric mode to keep track of the RAM address being pointed to in the lower two quadrants of the display. Constants used to generate the proper RAM addresses are kept in a ROM located within the ALU. The ALU also uses its subtractor to check if the address being pointed to by either the TCA or the BCA register matches the cursor address. If a match is found, the character font is automatically modified to display the selected cursor.

The display RAM interface is also controlled by the PLAs. The mapping of display RAM onto the screen depends on whether the controller is in alpha or graphics mode. In alpha mode, characters are stored as ASCII values with attribute bits. Selectable attributes include underline, inverse video, blinking, and a choice of three fonts which are software defined within the display memory. Upon fetching a character, the ASCII code and the font attributes are converted into a font address. The lower three bits of this font address are dictated by the current dot row. It is this font address whose data is the actual bit pattern sent to the screen.

Creating Plug-In ROMs for the Portable Plus Computer

by William R. Frolik

HP'S PORTABLE PLUS COMPUTER was designed with multiple plug-in ROM applications in mind. By enhancing the internal ROM disc concept used in its predecessor, The Portable, it has been made possible to custom bundle just about any application into the machine. This provides an expandable computer system that the user can easily customize by building in permanent software of the user's choice or design.

Until now, there has been no easy way for either the user or software vendors to produce programs in ROM form. What we needed was a way that this could be done by the customer, without requiring any assistance from Hewlett-Packard.

The Portable Plus ROM IMAGE Development Package was written for just this purpose. Used together with an additional RAM module and a peripheral EPROM programmer, it enables any Portable Plus to become a plug-in ROM development system. With the addition of a special ROM simulator card, the user can test the appearance and behavior of the final product before creating a ROM.

The ROM IMAGE Package consists of three parts:

- An installable electronic disc driver that assigns and maintains the RAM space in which the user creates an image of the final ROM
- A ROM image maintenance utility, with which the user can adjust the ROM's appearance into its desired final form and then save it in an MS™-DOS file
- A data transfer utility for copying the saved ROM image file to an external EPROM programmer or other device for conversion into one or more ROMs.

Electronic Disc Driver

The key component in the ROM IMAGE Package is an installable electronic disc driver called EDISK.SYS. This driver is used to set up and control one or more virtual disc drives in the Portable Plus in which the user constructs an image of the final ROM product. These virtual drives exist in addition to the two built-in electronic disc drives, A: and B:, and any external drives that may be in the system. When used in conjunction with one or more optional ROM simulator plug-in cards, EDISK.SYS sets aside the available RAM on the cards (generally some multiple of 256K bytes) into one or more drives of usable disc space. If used without a simulator card, EDISK.SYS attempts to acquire enough system RAM (in multiples of 64K bytes, up to a maximum of 256K bytes) to form one or two drives of usable disc space. Once EDISK.SYS has been installed and the virtual disc space has been allocated, the user is free to use the space in the same way as any physical disc—it must be formatted before it can be used, it has its own drive identifier and root directory, the user can build subdirectories and copy files, and so forth.

After the user has determined what the final ROM design should contain, the first step in building a plug-in application ROM is to set up an electronic disc drive of appropriate size and structure using EDISK.SYS with an appropriate DEVICE= command in a CONFIG.SYS file. Rebooting the system causes the driver to be installed and the virtual disc drive(s) to be allocated. The user then treats each drive as though it were any normal physical disc; application programs and files are added using the usual MS-DOS commands. Once all of the desired files, programs, and subdirectories have been stored on the virtual disc drive, the user runs the IMAGE.COM utility.

A user does not have to be creating a plug-in ROM to use EDISK.SYS. In a simple and effective way, EDISK.SYS provides an extra bit of fast-access file storage by allowing the addition of another electronic disc drive in the machine. However, drives set up by EDISK.SYS are volatile—their contents vanish if the computer is rebooted.

ROM Image Maintenance

IMAGE.COM is a ROM image maintenance utility. Used in conjunction with EDISK.SYS, IMAGE.COM is used to mark appropriate information into the boot sector of the virtual disc so that it conforms to the format of all plug-in application ROMs. Think of IMAGE.COM as sort of a ROM image formatter and debugger.

When IMAGE.COM starts running, the first thing it does is look for EDISK.SYS among the currently installed device drivers. If it is not found, an error message is issued and IMAGE.COM terminates. If it is found, IMAGE.COM retrieves information from EDISK.SYS about the currently installed virtual drives and then prompts the user for a command. From this point on, the program input is interactive. The user has available a variety of commands that can be used to make the electronic disc drive appear in the desired final ROM form.

In all but the most elaborate situations, the user generally goes through nine steps to finalize the ROM design and prepare it for transfer to an actual ROM or EPROM:

1. The MARK command is issued first. This formats the drive's boot sector so that, in the final ROM, it will be recognized by the BIOS ROM driver as a plug-in application ROM. A "ROM existence" byte is added, along with a time-and-date stamp.
2. The user specifies an 8-character name for the plug-in ROM using the ROMNAME command. This name becomes the name of the drive B: subdirectory in which the ROM's contents will appear. For example, if the ROM name is HPTOOLS, the entire contents of the virtual drive will, in ROM form, be mapped into the subdirectory B:HPTOOLS.
3. The OEMNAME command is optionally issued. This is

Structure of a Plug-In ROM

The Portable Plus supports two variations of plug-in ROM: the half bank and the full bank. While each has its own advantages and disadvantages, both forms share the common disc-like structure shown in Fig. 1. Each plug-in ROM is structured to look like a complete disc. The system's memory management code combines the file allocation tables and root directories of the individual ROMs "on the fly" into a single file allocation table and root directory for drive B: (see article on page 21).

The boot sector is not really a boot sector since the system does not attempt to boot from it. However, it does contain BIOS parameter block information like a normal disc's boot sector. In addition, it contains a ROM name, an optional OEM name, a time-and-date stamp, and a ROM type identifier and checksum information.

The ROM chips that compose a full-bank ROM come in pairs—one set of chips (or more, if EPROMs are used) covers the odd addresses, and an equal number of chips covers the even addresses. The EDISK.SYS driver partitions a single contiguous block of memory into a full-bank structure. Even and odd RAM addresses correspond to even and odd bytes within sectors. The entire drive maps one-for-one with the allocated RAM. When IMAGE.COM saves the drive image, it makes two passes through the data. The first pass copies all of the even-addressed bytes into a file ending with .EVN, and the second pass copies the skipped odd-addressed bytes into a file ending with .ODD. When the pair of ROMs are installed in the system, the BIOS verifies that they are a correctly installed full-bank pair by examining the first word of the boot sector. The byte at boot sector address 0 (the first byte of the .EVN half) must contain $B2_{16}$ and the byte at boot sector address 1 (the first byte of the .ODD half) must contain $B1_{16}$. If these ROM "existence" bytes are incorrect, the BIOS rejects the ROMs and ignores them.

A half-bank ROM can be a single chip—the entire drive structure is linearly stored within one or more chips with no differentiation between even and odd-addressed bytes. Address 0 within the ROM chip contains boot sector address 0, address 1 contains boot sector address 1, and so on. When plugged into a ROM slot, the contents of the half-bank ROM will appear at every other address, be it even or odd, since the hardware maps each whole ROM into only even or odd addresses. The BIOS detects the presence of a half-bank ROM by reading the first byte of its boot sector. A $B3_{16}$ byte identifies the ROM as a half bank. In accessing the data in a half-bank ROM, the ROM disc driver performs the address mapping necessary to read only the even or odd-addressed bytes containing the ROM's data. Note that this scheme permits two half-bank ROMs to reside adjacent to each other in odd and even ROM slots. If looked at on an absolute, sequential address basis, their contents will be interleaved. Functionally, however, the half-bank ROM is 99% the equivalent of a full-bank ROM. The only constraints are that, because of its every-other-address organization in the system plug-in memory space, the half-bank ROM cannot contain code executable directly from the ROM, and its maximum size is half that of a full-bank ROM.

Cluster	Sector	Boot Sector
0	0	00, 01: ROM existence bytes: B3B3 for a halfbank, B2B1 for a fullbank
		02 Status (bit 1 = ROM contains init code)
		03.. 0A: OEM name
		0B, 0C: Bytes per sector (512)
		0D Sectors per cluster (2)
		0E, 0F: Reserved sectors (1)
		10 Number of file allocation tables (1)
		11, 12: Maximum root directory entries (32)
		13, 14: Total number of sectors (256 or 512)
		15 Media descriptor (0FA)
		16, 17: Sectors per file allocation table
		18.. 1D: Extended device parameters (unused)
		1E, 1F: Last directory sector
		20.. 27: ROM name (drive B: subdirectory)
		28.. 2B: Time and date stamp
		2C.. 2F: Checksum (2C and 2E for even addresses, 2D and 2F for odd addresses)
		30.. 1FF: Optional initialization code (executed if bit 1 of the status byte is set)
	1	File Allocation Table
1	2, 3	Root Directory (32-byte entries define current directory (.), parent (..), and 30 other files
2	4	ROM Files, which includes: • Subdirectory files (if any, must come first) • Programs, data, batch files • ROM-executable code
...	...	
7F	FF	(Last sector of a half-bank ROM)
FF	1FF	(Last sector of a full-bank ROM)

Fig. 1. *Disc-like structure of Portable Plus plug-in ROMs.*

another 8-character name that is simply written into the boot sector. It is not required by the operating system.

4. The DIR command is optionally issued. This is similar to the MS-DOS DIR command, except that it shows only the contents of the drive's root directory. An important feature is that, unlike the MS-DOS version, the DIR command also shows deleted files, the initial sector (and paragraph) address of each file, and whether or not each file occupies contiguous sectors. (A ROM-executable file is created by storing a program file in the root directory of the virtual drive, ensuring that it occupies contiguous sectors, and noting its initial paragraph address. A small separate front-end program is then added to enable the ROM and jump to that initial paragraph.)

5. The FENCE command is issued. One of the requisites of a plug-in ROM is that all subdirectory files reside ahead of program and data files; the last subdirectory (or root directory) sector is the "fence" between the directory structure and all other data. The FENCE command scans the directory structure of the drive in an attempt to locate the fence. If found, its position is written into the boot sector. If any nondirectory files appear ahead of subdirectory files, the corrupt structure is reported to the user.
6. The CHECKSUM command is issued. This checksums the entire contents of the drive, writing the result into the boot sector. It is important that this be done last, since most of the preceding commands alter data in the drive's boot sector, which in turn affects the checksum.
7. The STATUS command is optionally issued. This displays a synopsis of the drive's boot sector.
8. If all is well, the SAVE command is issued. This command stores the entire image of the virtual drive into one or two MS-DOS files on some other (probably external) drive. If a half ROM bank is being constructed, one file with the extension .HAF is written. If a full ROM bank is being constructed, two files, with the extensions .EVN and .ODD, are written.
9. The QUIT command is issued, causing IMAGE.COM to terminate.

IMAGE.COM provides several additional commands which, while generally not required, can be used to examine or alter the structure of the ROM. An ENTER command, for example, permits individual bytes to be entered into any address in the disc structure. The HIDE command can be used to hide a root directory file so that it will not show up in a normal MS-DOS directory listing. INITCODE can be used to copy a small executable file into the normally unused leftover space in the boot sector. Each time the system is reset, this code can be downloaded into system RAM and executed. Typing HELP, or just ?, causes ROM IMAGE to display a list of all supported commands.

At this point, the final ROM image is totally contained in one or two normal MS-DOS files, ready to be transferred to ROM or EPROM. Included in the package is a data transfer program, BURN.COM, which can be used to copy each image file to an EPROM programmer (or any other device) via one of the supported transfer formats (Intel MDS hexadecimal, Motorola Exorcisor hexadecimal, RCA COSMAC hexadecimal, unformatted ASCII hexadecimal, or raw binary).

Acknowledgments

The HP Portable Plus ROM disc (drive B:) driver was designed and implemented by Mark Rowe, who also defined the full-bank and half-bank ROM formats and wrote the code necessary to map them into the ROM disc subdirectory structure. The entire electronic disc structure and its management are based loosely on the original HP 110 Portable ROM disc and RAM disc designs by T.W. Cook. The initial version of the ROM IMAGE program, which made first use of John Eaton's ROM simulator card, was written by Randy Salo, with enhancements by Jennifer Wood.

Authors

July 1986

4 Portable Computer Design

Carl B. Lantz

An R&D engineer with HP since 1981, Carl Lantz was a member of the team that designed the multicontroller IC for The Portable and was a system engineer for the Portable Plus. His specialty is CMOS IC design. Carl was born in Massena, New York and completed work for a BSEE degree from Clarkson College of Technology in 1981. He lives in Corvallis, Oregon and likes to spend his spare time customizing vans, traveling, skiing, and ballooning.

Michael J. Barbour

Born in McMinnville, Oregon, Mike Barbour received a BS degree in physics from Washington State University in 1971 and a Masters degree in education from the University of Hartford in 1972. He has been with HP since 1977 and has worked on the tape read/write chip and plug-in modules for the HP-85 Personal Computer. He has also contributed in the design of the LCD and video controllers for The Portable and Portable Plus. Mike lives in Corvallis, Oregon.

James W. Pearson

An R&D engineer at HP's Personal Computer Division, Jim Pearson has been with HP since 1980. He was a manufacturing engineer for the HP-85 and HP-86 Personal Computers and wrote the diagnostics for The Portable and Portable Plus. He was born in Muncie, Indiana and studied zoology at the University of California at Davis, completing work for his BS degree in 1970. He did research work on yellow fever mosquitoes at Davis before continuing his education at San Diego State University (MSEE 1980). He also served in the California National Guard. Jim and his wife and daughter live near Corvallis, Oregon. He's active in his church and enjoys spending time with his family.

Courtney Loomis

Born in Honolulu, Hawaii, Courtney Loomis received his BS and MS degrees in electrical and computer engineering from Oregon State University in 1979 and 1982. He has been with HP since 1982 and was codesigner of an IC that was used in The Portable and Portable Plus. He also designed the circuits for the RAM and ROM modules for the Portable Plus. His professional interests include user interface software and software tools for scientists. Courtney lives in Corvallis, Oregon and enjoys mountaineering, cross-country skiing, and backpacking. He's also interested in natural history and field biology.

Ella M. Duyck

With HP since 1980, Ella Duyck is a mechanical engineer and has contributed to the design of the HP-87 Personal Computer, The Portable, and the Portable Plus. She was also a production engineer for the HP-87. She earned her BSME degree from Oregon State University in 1977 and designed dies at a tube mill before coming to HP. Her technical specialties include plastic part design and product packaging. Ella is an Oregon native, born in Hillsboro and now living in Philomath. She likes outdoor pastimes, especially skiing, backpacking, and gardening and reports that she and her husband are nearly finished with a three-year home-building project.

John T. Eaton

Born in Vincennes, Indiana, John Eaton served for six years in the U.S. Navy and is a graduate of Purdue University (BSEE 1980). After starting at HP the same year he was a production engineer for the HP 82901 Disc Drive. He later became an R&D engineer and contributed to the design of the HP-86 Personal Computer. He also did the system memory and microcontroller design for The Portable. John lives in Corvallis, Oregon.

Clifford B. Cordy, Jr.

With HP since 1980, Cliff Cordy has worked on the HP-86 Personal Computer and The Portable. His contributions have been in the areas of analog circuit design, power supplies, motor drivers, and the testing and characterization of batteries. Before coming to HP he worked on optical instrumentation, electro-optical systems, laser systems, and power supplies. His work on electronic circuits and laser systems has resulted in 12 patents. Cliff was born in Medford, Oregon and studied mathematics, physics, and electrical engineering at Oregon State University. His degrees include a BS in mathematics (1959), a BS in physics (1960), a BSEE (1960), and an MSEE (1963). In addition, he recently completed course work for a PhD in electrical engineering. He's a runner and set several track records while at Oregon State University and has set American records and a Senior Olympics record. He also participated in white water kayaking trials for the 1972 Olympic Games. He supports track events at Oregon State and is a member of the President's Club at both Oregon State and Southern Oregon State College. He lives in Corvallis, is married, and has two children and one grandchild.

14 I/O and Data Communications

Harold B. Noyes

With HP since 1981, Harold Noyes worked on the design of the HP 82983A Modem for the Portable Plus and on the design of the CMOS RAM for The Portable. Earlier, he wrote a technical reference manual for HP Series 80 Personal Computers. His work

on the HP 82983A Modem is the subject of a pending patent. He was born in Downey, Idaho and attended Utah State University, completing the requirements for a BSEE degree in 1981. He and his wife and daughter live in Corvallis, Oregon. He's active in his church and is a Boy Scout advisor. His leisure activities include all kinds of skiing, flying model airplanes, and dancing.

Andrew W. Davidson

Andy Davidson has been an HP R&D engineer since 1980. He has contributed to the development of the HP-86 and HP-87 Personal Computers, the modem for The Portable, and the power supply for the Portable Plus. He was also a production engineer for the HP-87 and was coauthor of an *HP Journal* article on Series 80 HP Computers. Born in Abington, Pennsylvania, he earned a BS degree in electrical and computer engineering from Clarkson College of Technology in 1980. He's a resident of Corvallis, Oregon, is married, and has one daughter. He enjoys listening to music and playing his guitar.

18 Personal Applications Manager

Alesia Duncombe

A software engineer with HP's Portable Computer Division, Alesia Duncombe has been with the company since 1979. Her first job at HP was in technical customer support, and after completing her studies for a BSEE degree from Oregon State University in 1981, she moved to an R&D position. She has worked on the design and implementation of operating system firmware for the HP ThinkJet Printer and for the Portable Plus. Alesia was born in Seattle, Washington and is currently living in El Cerrito, California while her husband, also an HP engineer, attends graduate school. They are the parents of a baby boy. Her hobbies include skiing, sewing, backpacking, and travel.

Robert B. May

Bob May was born in Belleville, Illinois and grew up in Albuquerque, New Mexico. He began working for HP as a part-time student in 1980 and earned a BS degree in computer science in 1981 from the New Mexico Institute of Mining and Technology. His contributions as an R&D engineer include working on several ROMs for the HP-86 and HP-87 Personal Computers and developing system software for the internal modem and serial port for the Portable Plus. He also developed an arcade game for the HP-86 and HP-87, which HP has sold through the employee software program. Bob and his wife live in Corvallis, Oregon. He likes touring Oregon wineries, riding his mountain bike, and cooking Mexican food.

21 Memory Management

Mark S. Rowe

An alumnus of the University of California at Davis, Mark Rowe holds a 1973 BS degree in computer science and a 1977 MSEE degree. With HP since 1975, he has contributed to the development of the HP 7910 Disc Drive and the HP-75 Handheld Computer. He worked on application software and testing for The Portable and wrote the memory management software for the Portable Plus. Mark was born in Sacramento, California and now lives in Philomath, Oregon. He's married and has a son and a daughter. He likes to spend his free time sailing his family's 26-foot sailboat and also has a private pilot's license with glider and instrument ratings.

25 Hybrid LCD Controller

Glenn J. Adler

Born in Englewood, New Jersey, Glenn Adler studied electrical and biomedical engineering at Brown University and completed work for his BS degree in 1981. He came to HP the same year and was an IC designer and a production engineer for The Portable and HP Series 80 Personal Computers before leaving HP. He's the author of an article on liquid-crystal displays and is interested in neurosciences and electrical modeling. He likes ultimate frisbee, skiing, and jazz.

28 Creating Plug-In ROMs

William R. Frolik

An Oregon native, Bill Frolik has been with HP's Portable Computer Division since 1980. He designed the extended memory controller for the HP-86 and HP-87 Personal Computers and was an IC production engineer for both products. He was also a designer of the CMOS ROM for The Portable and was a member of the system firmware team for the Portable Plus. Bill was born in Corvallis and earned a BS degree in physics from Willamette University in 1979. He completed work for his MSEE degree from Stanford University in 1980. He currently lives in Corvallis and has many outside interests, including photography, piano, woodworking, skiing, tennis, bicycling, golf, landscaping, and travel.

34 Series 300 HP-UX Features

Robert J. Schneider

Born in Syracuse, New York, Bob Schneider has been an HP software engineer since 1983. He worked on the HP-UX device I/O library definition and on its implementation for the HP-UX system for the HP 9000 Series 500 Computer. He studied computer science at the University of Vermont (BSCS 1981) and at Oklahoma State University (MSCS 1983). His professional interests include operating systems and non von Neumann computer architectures. Bob and his wife live in Fort Collins, Colorado. During his leisure time he enjoys sailing, skiing, guitar, and piano.

David L. Frydendall

David Frydendall studied computer science at Colorado State University (BS 1976) and at the University of Colorado (MS 1982). After working on several computer graphics projects for Sperry Univac, he joined HP in 1978. His first assignment was on the HP 1350S Computer Graphics System. Later he developed the 16-bit parallel and serial drivers for the HP 9000 Series 500 Computers. He also contributed to the HP NS/9000 LAN for the Series 500 and initiated the development of windowing software for HP 9000 Computers. A Colorado native, David was born in Greeley. He's married, has a son, and lives in Fort Collins. When not working on the construction of his mountain home, he enjoys music, camping, fishing, and cross-country skiing.

Robert M. Lenk

Bob Lenk was born in New York City and is an alumnus of the University of Connecticut. He completed work for his BA degree in mathematics in 1975 and for his MS degree in computer science in 1981. With HP since 1981, he's a software development engineer and has worked on the HP-UX system kernel for HP 9000 Series 500, 200, and 300 Computers. His other major contribution has been on the HP NS/9000 LAN for the Series 500 Computer. He is a coauthor of several technical papers and is interested in operating systems. Bob and his wife are residents of Fort Collins, Colorado. His outside interests include cross-country skiing and gardening.

Bonnie Dykes Stahlin

With HP since 1969, Bonnie Dykes Stahlin has held a variety of marketing and R&D positions in the U.S.A. and Germany and is now a project manager working on the HP-UX system at Fort Collins, Colorado. She was born in Wray, Colorado and studied computer science at Colorado State University. She received her BSCS degree in 1978 and her MSCS degree in 1983. She and her husband are residents of Loveland, Colorado and her outside interests include sailing, windsurfing, camping, and sewing. She's also a fitness instructor for the health and fitness program at her division.

Andrew G. Anderson

Drew Anderson was born in Rochester, Minnesota and educated at the University of Minnesota at Minneapolis-St. Paul. He received a BS degree in mathematics in 1982 and a BS degree in computer science in 1984. He has been with HP since 1984 and has worked on the HP-UX system kernel for the HP 9000 Series 200 and 300 Computers. He was a musician for ten years before coming to HP. Drew and his wife and two children live in Fort Collins, Colorado. During his leisure time he coaches his son's soccer team.

Robert D. Gardner

With HP since 1983, Rob Gardner is a specialist in data communications and UNIX systems. He has worked on asynchronous data communications and on terminal emulators for the HP-UX system and has written data communications software for HP 150 Personal Computers and for the X.25 networking protocol. He's the author of a paper on data communications and coauthor of a paper on graph theory. Born in New York City, Rob has a BS degree in computer science and an MS degree in mathematics from Clarkson University. Both degrees were awarded in 1983. A resident of Fort Collins, Colorado, he enjoys chess, music, tennis, mathematical puzzles, and motor sports. He also reads science fiction and brews beer.

Ronald G. Tolley

Born in Salt Lake City, Utah, Ron Tolley graduated from the University of Utah in 1976 with a BSEE degree and from Purdue University in 1981 with an MSEE degree. He came to HP in 1978 and has developed software for the HP 9845B Computer and the HP 9000 Series 500 Computer. His work on the HP-UX system has been in the areas of system performance and native language support. His professional interests include data flow computation, internationalization, computer graphics, and image processing. Ron lives in Loveland, Colorado, is married, and has five children. He's active in his church and in local political activities and likes tennis, vocal music, and creating watercolor and pastel pictures.

James M. Umphrey

A project manager at HP's Colorado Telecommunications Division, Jim Umphrey has contributed to the development of a number of oscilloscopes, pulse generators, and protocol analyzers since joining HP in 1961. He was project manager for the HP 4971S Protocol Analyzer. Born in Eugene, Oregon, he attended Stanford University, receiving a BSEE degree in 1961 and an MSEE degree in 1964. His work on sampling is the subject of a patent and he is the author of several technical papers. Jim is married and has five daughters, including a set of twins. He lives in Colorado Springs, Colorado, is active in Big Brothers, and likes woodworking, running, and reading.

Jeffrey Tomberlin

Jeff Tomberlin was responsible for Pascal and assembly language programming for the HP 4971S Protocol Analyzer and also contributed to the analog and digital design and assembly language software for the HP 4945A Transmission Impairment Measuring Set. Born in Evanston, Illinois, he completed work for his BSEE from Georgia Institute of Technology in 1980 and came to HP the same year. His professional specialties are real-time software and hardware interfaces. He lives in Colorado Springs, Colorado and enjoys swimming, bicycling, hiking, and skiing during his leisure hours.

Jeffrey H. Smith

Jeff Smith is a Stanford University alumnus with a 1962 BSEE degree and a 1963 MSEE degree. He has been with HP since 1963 and has contributed to the design of a number of sampling oscilloscopes, pulse generators, and logic analyzers. He's named inventor on a patent related to sampling oscilloscopes and is the author or coauthor of several HP Journal articles and conference papers. Born in Portland, Oregon, he is a resident of Colorado Springs, Colorado, is married, and has three children. He is restoring a vintage Corvette and enjoys running, sailing, woodworking, and working with stained glass.

Gordon A. Jensen

Gordon Jensen is a graduate of the University of California at Davis. He completed work for his BSEE degree in 1981 and came to HP the same year. He's a hardware designer and has worked on the HP 4945A Transmission Impairment Measuring Set and the HP 4971S. He is named coinventor for two patent applications on LAN protocol analysis and is coauthor of a 1984 HP Journal article on the HP 4945A. Born in Santa Ana, California, Gordon lives in Colorado Springs, Colorado. His outside interests include skiing, mountain climbing, bicycling, and exploring the American Southwest.

Stephen P. Reames

With HP since 1979, Steve Reames is an R&D engineer at the Colorado Telecommunications Division. He has worked on the HP 3326A Synthesizer, the HP 4937A Transmission Impairment Measuring Set, and the HP 4971S. His work on the HP 4971S involved the hardware design of the interface to the LAN. He was born in Boston, Massachusetts and graduated from the University of California at Berkeley with a BSEE degree in 1979. He is named coinventor for two patent applications on LAN protocol analysis and he's the author of a tutorial article on telephone signaling. Steve lives in Colorado Springs and spends his spare time exploring Colorado, spelunking, and flying radio-controlled sailplanes.

Jerry D. Morris

Jerry Morris has been with HP since 1975, the same year he received a BSEE degree from Iowa State University. After an assignment in production engineering he moved to an R&D position and has done analog, digital, firmware, and software design for the HP 4960A and HP 4961A Pair Identifiers, the HP 4945A Transmission Impairment Measuring Set, and the HP 4971S Protocol Analyzer. A patent application has resulted from his work on a runt filter for the HP 4971S. He holds an MSCS degree from the University of Santa Clara (1983) and was a computer repair instructor while he served in the U.S. Army. Born in Des Moines, Iowa, he's married and lives in Colorado Springs, Colorado. He enjoys music and photography when not working on home remodeling projects.

New HP-UX Features for HP 9000 Series 300 Workstations

The capabilities of the HP-UX operating system have been extended in the Series 300 implementation to handle real-time applications, communication with X.25 networks, and operation in native languages.

by Andrew G. Anderson, David L. Frydendall, Robert D. Gardner, Robert M. Lenk, Robert J. Schneider, Bonnie Dykes Stahlin, and Ronald G. Tolley

SINCE ITS INTRODUCTION on HP 9000 Series 500 Computers in 1983,[1,2,3] the HP-UX operating system has been implemented on HP 9000 Series 200 Computers and the Integral Personal Computer.[4] In that time it has grown from a superset of AT&T Bell Laboratories' UNIX™ System III™ to a superset of AT&T's UNIX System V™, release 2. In addition, the HP-UX operating system has been extended in several directions that are important to Hewlett-Packard's traditional technical computer customers. All such extensions have been defined very carefully to fit in with the industry standard UNIX System V definition while providing added needed functionality.

The HP-UX 5.0 release is the first version for the Series 300, and the 5.1 release is a common version for both the Series 200 and Series 300. These releases provide many of the features previously available only on the Series 500, such as virtual memory and local area networking.[5] In addition, they introduce a number of important new features not found on the earlier HP-UX system. These features include support for device I/O, extensions for real-time programming, support for communications over an X.25 network, support for users' native languages, and a window-oriented human interface for the new bit-mapped displays developed for the Series 300. Most of these new features are also provided for the Series 500 in its HP-UX 5.0 release, and on the new HP 9000 Series 800. Many are available on the Integral PC as well.

Device I/O

Standard implementations of the UNIX operating system, while addressing the needs of software developers, do not provide the device I/O capabilities required for instrument controller systems. However, to be useful in solving scientific and technical problems, such instrument controller capabilities must be present.

To ensure a successful design and implementation that would solve instrument controller problems, we had to:

- Provide the functionality for full control of the HP-IB (IEEE 488/IEC 625) and the 16-bit parallel GPIO (general-purpose input/output) interface cards.
- Define an HP-UX standard for the device I/O functionality to ensure portability of solutions across the entire HP-UX-based product family.
- Maintain a unified I/O approach by using the existing I/O functionality provided in the HP-UX system.

Achieving the second objective required that differences in hardware and operating systems be hidden. This was accomplished by implementing the user interface at the library level. The device I/O library combines with the HP-UX system to provide a total I/O solution (Fig. 1). The library provides functions for controlling devices while the system provides functions for reading and writing data.

The device I/O library defines three classes of functions. The first class includes generic functions that apply to both interface cards. The second class of functions applies to the HP-IB interface card only. The third class of functions applies to the GPIO interface card only. The GPIO functions allow for setting control lines and reading status lines. The HP-IB functions allow for obtaining bus status information, conducting serial or parallel polls, addressing the bus, and performing several controller-related actions. The generic functions include such capabilities as setting time-outs and data path widths. In addition, the HP 9000 Series 300 HP-UX implementation provides an additional function within its I/O library for enabling a high-speed HP-IB instrument data transfer mode.

The ability to transfer small data packets at high speed is critical in many instrument controller applications. To provide this capability, a new function called ioburst was added to the I/O library. This function allows the calling process to enable or disable burst mode. Normally, a data transfer request results in a system call to the kernel to perform the actual I/O. When burst mode is enabled, the memory mapped I/O address space of the interface card is mapped directly into the user's address space. This allows the user to talk directly to the interface card without the overhead of a kernel call. Burst mode significantly improves the performance for reading and writing and sending HP-IB bus configuration commands. All other operations are unaffected and take their normal path through the HP-UX kernel.

After the interface card is mapped into the user's address space, subsequent data transfer requests are intercepted by the I/O library and directed to special data transfer routines in the library. These routines directly access the interface card through the user's address space. The typical path has been tuned to less than 50 processor states and results in

Fig. 1. *Sophisticated instrumentation solutions are possible with a Series 300 Computer interfaced to the instruments using the HP-UX 5.0 device I/O library.*

reducing the operating system overhead for a given data transfer by approximately two orders of magnitude for the Model 320 Computer.

Burst mode provides the fastest path for the transfer of small data packets typical of instrumentation systems. Very large data packets can be transferred faster by the HP-UX kernel using DMA. In systems where DMA is not available, burst mode always provides the fastest path. The break-even point is approximately 4K-byte packets on a Model 320 system. Larger packets can be transferred faster by DMA and smaller packets are transferred faster using burst mode. For example, reading an 8-byte packet from a voltmeter using a Model 320 will take approximately 114 μs in burst mode and approximately 7422 μs using DMA.

The Nelson benchmark[6] is commonly used to measure the performance of instrument controller systems. Running this benchmark on a Model 320 produced a result of 385 readings per second, which approaches the theoretical limits of the HP 3495A Scanner used. This is almost two times faster than the BASIC operating system running on a Model 320 Computer, which attained a rate of 230 readings per second.

Real-Time Extensions

Having been developed for timesharing applications, most UNIX systems do not provide the features or performance necessary for real-time applications. Some of the features included in the HP-UX system as part of its UNIX System V implementation are designed to meet some of the needs for real-time applications. In other cases, features have been taken from one of the other popular UNIX versions in the marketplace, 4.2BSD, which was developed at the University of California at Berkeley. When no appropriate feature existed in the marketplace, a new one was defined to fill the void. A more detailed description of these extensions appears in reference 7.

The definition used here of a real-time application is one that must reliably interact with or respond to entities outside the computer on a time schedule that is driven by those outside entities. This definition is extremely general, since it can be applied to virtually any computer application. The key is that real-time does not define a black-or-white distinction that can be applied to applications or to operating systems; it defines a continuum of requirements. Near one end of the continuum are applications that need to respond to discrete inputs such as keystrokes. The generally accepted response time for such applications is about one tenth of a second. Standard UNIX systems are generally designed to perform well in this area, but not much beyond it. Those applications at the other end of the continuum will most likely need custom operating systems to get the most out of any given hardware and meet their goals. However, there are many applications that fit somewhere in the middle, including those that run on HP's BASIC and Pascal Workstations and HP 1000 Computers. The purpose of the real-time extensions is to allow the HP-UX system to cover a larger segment of this continuum. This gives application writers and users the advantages of an industry standard operating system, and still allows them to solve their problems.

One of the key limitations in the standard UNIX feature set is in the resolution of primitives that deal with time. The standard primitives for scheduling events at a given time have a resolution of one second, which is insufficient even when dealing with humans. A feature known as interval timers was taken from the 4.2BSD system to address this problem. Each process can schedule either a single interrupt or regularly repeating interrupts with whatever precision the underlying hardware and operating system support. The interval is expressed in units of seconds and microseconds to keep the interface portable despite the system-dependent resolution. The supported timer resolution on the Series 300 is 20 milliseconds.

Another limitation is the minimal set of interprocess communication mechanisms available. There are only two mechanisms common to all UNIX systems, pipes and signals. These facilities have various weaknesses in functionality, performance, and reliability. The latest release of the HP-UX system includes three new interprocess communication facilities from UNIX System V, plus a reliable signal interface based on one introduced in the 4.2BSD system.

The three facilities from UNIX System V are semaphores, messages, and shared memory. The semaphore mechanism is very elaborate, allowing solutions to both simple and complex synchronization problems. The message passing mechanism allows transfer of data without any disc access, and provides features such as tagging and prioritization of messages that are unavailable with pipes. The shared memory facility is probably the most important for real-time needs, allowing by far the highest communication bandwidth, since data does not need to be copied to be communicated. In the Series 300 Computers, shared memory is paged by default, but can be locked in main memory to provide optimal performance. All three of these mechanisms share an interface that allows communication and synchronization among arbitrary unrelated processes, yet provides protection from unauthorized access.

Signals are essentially a software interrupt mechanism, but the standard UNIX definition includes several race conditions which make the mechanism unreliable for inter-

process communication. The 4.2BSD system introduced a new signal mechanism to solve these reliability problems. Modeled after hardware interrupts, its main contribution is the ability to mask out signals to eliminate race conditions. The major shortcoming of this new definition is that it does not allow full emulation of the standard signals used in virtually every other UNIX system, including early HP-UX systems. This is because of an orthogonal change that transparently restarts system calls that have been interrupted by signals. A few minor modifications to this portion of the 4.2BSD definition yielded one for the HP-UX system that can completely emulate both the standard UNIX mechanism and the new 4.2BSD mechanism. The HP-UX definition allows the user to choose whether an interrupted call is restarted as in 4.2BSD or aborted as in the standard UNIX mechanism.

Another limitation in standard UNIX systems is the degree of control available over process priorities. The UNIX process priority structure was designed for multiuser timesharing systems, with major goals of fairness to all users and acceptable response to users at terminals. To achieve these goals, the system dynamically adjusts process priorities, favoring interactive processes with light CPU use at the expense of those using the CPU heavily. Users are given some control of priorities with the nice system call, but the values specified by it are actually only one factor in a formula. As a result, it is difficult or impossible to guarantee that one process has an effective priority greater than another. In addition, processes executing within the kernel often have their priorities increased to favor them over any process executing user code. The priorities used in these situations are based only on the kernel code being executed, not the original priority of the process. Thus low-priority processes inside the kernel are favored over high-priority ones outside the kernel or in different portions of the kernel.

Since these problems have not been addressed in any of the commonly available UNIX systems, a new solution is introduced with the real-time extensions of the HP-UX system. This solution extends the priority model with a new range of priorities, named real-time priorities, and a new system call rtprio, which allows processes to set their priorities in this range. Priorities in the real-time range are not dynamically adjusted by the operating system, but maintain absolute values as set by the user. Any process with a priority in this range is favored over any user process with a priority in the normal range, regardless of whether either process is executing kernel or user code. Furthermore, the rtprio call allows processes to read and modify not only their own priorities, but also those of other processes owned by the same user. Processes with priorities in the normal range continue to behave with the standard UNIX semantics.

One major factor in the real-time response of some applications is the speed at which they can log data to disc files or read the data logged by other processes. The mechanism used by the standard UNIX file system to allocate file space does not lend itself to optimal performance in this area. File space is allocated only at the time a write is performed, adding new blocks to the file as needed from a list of free blocks. In the case of a single application writing to a freshly formatted file system, this may produce an optimal file layout. However, as files are created and destroyed, the free block list tends to become random, as does the layout of any given file.

The 4.2BSD file system implementation is better suited for optimal file layout than the standard UNIX implementation. It partitions disc space into cylinder groups with small seek times within any one group, and attempts to allocate space for a given file from a single cylinder group. Unrelated files are placed in different cylinder groups to minimize contention for the same space. A bit map of free blocks in each cylinder group is maintained and used in place of the linear free block list. The Series 300 HP-UX system uses essentially the 4.2BSD implementation, with minor modifications to eliminate incompatibilities with other UNIX and HP-UX systems that are visible to the user. Even though this implementation almost always results in more optimal file layouts than the free block list algorithm, the layouts can be unnecessarily fragmented when file space is only allocated as data is written. Thus a new system call prealloc is included in the real-time HP-UX extensions. As the name suggests, this call is used to preallocate space for a file before the actual data is written.

The Series 300 virtual memory system supports user processes whose individual or combined sizes exceed physical memory. The time required to bring a page or an entire process from disc into memory can range from several milliseconds to several seconds, and thus can violate almost any real-time requirement. The effect is often a wide variation in the performance of a given task, where the normal time is acceptable, but an occasional swap causes problems. The HP-UX system has adopted a solution to this problem from UNIX System V. The plock system call allows the caller to lock the caller's executable code and/or data in memory to avoid unexpected swapping and paging. In addition, as alluded to above, shared memory segments can be locked in memory with the shmctl call.

Some of these real-time features, notably real-time priorities and locking of memory, allow users to degrade the performance seen by others on a multiuser system significantly. The standard approach to capabilities of this nature in UNIX systems is to restrict the capabilities to the superuser or system administrator, who by definition has the ability to impact all other users. Unfortunately, the superuser has many very dangerous powers, such as the ability to write or remove any file in the system, and care must be taken by anyone running as the superuser to avoid their accidental misuse. Therefore, it is not practical to require users using these real-time features to run as a superuser.

The HP-UX system includes a new concept called privileged groups as a solution to this problem. Several potentially dangerous capabilities or privileges are ordinarily reserved for the superuser. However, the superuser can assign any of these privileges to a group of users, designating that group as a privileged group. All processes in a privileged group are empowered with the restricted functionality. The Series 300 HP-UX implementation permits assignment of the real-time priority and memory locking privileges, as well as a third privilege not related to the real-time extensions.

One experiment has been designed and run to give an indication of the effect of the new features on interrupt response time. The measurements involved the response to an SRQ interrupt on an HP-IB network. The experimental steps performed were:

- A program on the host is written with the device I/O library to wait for SRQ and then drop the REN line on the bus.
- A separate machine asserts SRQ.
- An HP 1630G Logic Analyzer clocks the interval from SRQ being raised to REN being dropped.

Three conditions are varied on the host:

1. Whether or not the system has a background load running. The background load used consists of a C compile and a copy of the B1D Whetstone benchmark. When the load is not present, the system is dedicated to the device I/O library program.
2. Whether or not the device I/O library program runs with a real-time priority (obtained with the rtprio call). Otherwise it uses the default nice value, as does the background load.
3. Whether or not the device I/O library program locks itself in memory with the plock call.

The results are shown in Fig. 2. The same experiment was run on an HP 9000 Model 550 system, and those results are included too. These results apply only to this experiment, and should not be extrapolated to other applications. In particular, the maximum times quoted are only the maxima observed in these samples, and should not by any means be interpreted as guaranteed worst-case times for this or any other application. No values are given for the Series 300 machines on a dedicated system using rtprio and/or plock. These features did not show a significant impact on a dedicated system, so full sets of measurements were not done.

These real-time extensions have been used in developing the interactive Starbase Graphics Library and the HP Windows/9000 system. The results of these efforts subjectively show a marked improvement over previous HP-UX systems in smoothly tracking input devices under human control. These particular applications will prove very important in the ability of the HP-UX system to support computer-aided engineering.

X.25 Extensions

HP-UX systems have always included the standard uucp (UNIX-to-UNIX copy) facility for asynchronous communication over RS-232-C/V.24 and modem lines as described in reference 8. This facility has been extended to support connection of HP-UX systems to an X.25 network. The use of the X.25 Public Data Network (PDN) has become widespread in Europe and is now becoming quite popular in the U.S.A. as well. Although it is quite difficult (from a hardware perspective) and expensive to attach a computer to an X.25 network directly, there is a much simpler and cheaper solution that is effective for many applications. The Packet-Assembler-Disassembler (PAD) is a microcomputer that has been programmed to make it easy to connect an arbitrary computer or terminal to an X.25 network. The PAD provides a standard serial RS-232-C/V.24 interface to a terminal or computer, does the work of electrical connection to the network, and also handles the necessary protocols to move data through the network.

The HP 2334A PAD is used in the HP-UX implementation, and is treated very much like a modem would be. The usual dialit mechanism is used as an interface between user processes and the HP 2334A. dialit takes an X.25 address (analogous to a phone number for a modem), initiates the communication with the HP 2334A, and returns with an open communication line for the calling process to use.

Several modifications were needed to accommodate some of the eccentricities of the PAD and the X.25 network. For security reasons, the PAD provides the address of the caller when a switched virtual circuit (SVC) is established. This address must be logged on the receiving system. The HP-UX utility getty, which usually receives incoming calls, does not have the capability to log the caller's address, nor does it have the ability to perform some simple configuration on the associated device. For these reasons, getty is replaced by getx25, which is almost the same, except for the additional features required by the X.25 network. To make it easy for a user to use a different PAD from the HP 2334A, we provide facilities to make this step straightforward. The central tool is a PAD configuration language which has such simple statements as send xxx and expect yyy. This saves the user the trouble of writing and debugging a C module that would ordinarily be linked into dialit.

The HP-UX X.25 connection has been running at many HP sites since mid-1984, and at several sites outside HP since mid-1985 when this software received official sanction and support under HP-UX 5.0. The most common use is for exchange of electronic mail among distant machines. In terms of traffic amounts, notes (an electronic bulletin board system) accounts for a large percentage of the flow. Less often, the X.25 connection is used for simple movement of files from one system to another. This is particularly convenient for sending data overseas, since the physical mailing of a tape could easily take several days, and most likely more. Occasionally, the X.25 link is used for a

Variable Conditions			Hardware Model								
			310			320			550		
load	rtprio	plock	mean	σ	max	mean	σ	max	mean	σ	max
no	no	no	2.249	0.179	3.044	2.031	0.452	3.431	1.855	0.194	3.709
no	no	yes							1.927	0.298	3.286
no	yes	no							1.647	0.044	1.811
no	yes	yes							1.646	0.045	1.806
yes	no	no	5.460	19.41	277.6	5.865	10.37	103.0	46.11	127.7	872.7
yes	no	yes	5.213	18.59	329.6	4.763	4.388	36.19	59.23	148.7	985.0
yes	yes	no	5.309	17.09	343.0	4.850	4.391	38.27	1.678	0.072	2.339
yes	yes	yes	4.268	10.55	186.8	4.288	3.766	46.28	1.680	0.072	2.336

Fig. 2. SRQ *response times in milliseconds.*

remote terminal session. This setup works for many situations, but is not supported since raw mode applications will not function properly unless one is willing to accept the inefficiency of single-character X.25 packets.

The main accomplishment of the HP-UX X.25 connection has been to increase the communications capabilities of the HP-UX system to global proportions, and to reduce the cost of large-scale data traffic applications for systems separated by great distances.

Native Language Support

Native language support (NLS) provides users in the international market access to HP-UX capabilities that until now have been limited to those that speak and understand American English. Internationalization and localization are two other words that are often used to describe the same or similar concepts.

Several objectives guided the development of the NLS product on the HP-UX system:

- To deliver a single product that serves the needs of all international users.
- To save disc space, especially on smaller systems, by separating the resources required for a specific language product from the executable code. The components of these products are tables of data used by the software in the base product.
- To allow multiple users to use the same system, each in his or her own native language.
- To permit the user to specify the desired native language at run time.
- To provide a set of tools so that software can make use of NLS with a minimum of added development effort.

Three main barriers prevent an international user from using the HP-UX system in the user's own language: character set support, local customs, and messages. Bale and Kellogg in their article[9] review in some detail how these limitations are being removed.

Character Set Support. The UNIX system was originally used on machines that use the ASCII character set. The ASCII character set consists of 33 control characters (including DEL), the space character, and 94 printable characters. This is sufficient to write in American English, but is not even sufficient for British English since the British pound sterling character £ is missing.

To provide the additional characters needed to support Western European and other languages, eight bits (one byte) are used for each character. This provides a character set with 67 control codes, the space character, and 188 printable characters. The character set used by HP for Western Europe is called the Roman8 character set.[3] Other 8-bit character sets are also available.

The assumption that characters are always represented by the ASCII character set has led to programs that cannot support 8-bit character definitions. For example, the vi full-screen editor uses the eighth bit of each byte for internal processing. To allow eight bits of character data, the code must be rewritten to store internal processing data elsewhere. Many other commands share this problem.

A number of languages have very large character sets that require more than the 188 printable characters provided by the 8-bit character codes. For example, there are perhaps 50,000 Japanese Kanji ideograms. While 3000 to 7000 ideograms are required for daily use, this is well beyond the scope of an 8-bit character set. Sixteen-bit character codes are available for these languages.

In the standard HP-UX library routines, most of the resources are hard-coded. For example, the ctype (character type) routines which return whether a character is uppercase, lowercase, numeric, et cetera access a table of 129 bytes for the ASCII character set. As discussed earlier, this is not sufficient for European and phonetic Japanese character sets. A new family of nl_ctype (native language character type) routines has been added to the system. These routines access a file that contains the table for the Roman8 character set. An integer value from the table above is passed to an nl_ctype routine to indicate which language to use.

Local Customs. Some aspects of NLS relate more to local customs of a particular geographic location than to the characters used to write the language. Even the seemingly universal Arabic numbers are not identically formatted. The character used to denote the radix of a decimal number is a period in America, but a comma in continental Europe. The indicators used to separate thousands from millions are the reverse—a comma in America, a period in Europe. Thus, 1,432,678.09 in America is 1.432.679,09 in Europe.

The symbol for currency is necessarily different for each country. Symbols for the Japanese yen, the Dutch guilder, and the Italian lira are all required. The symbol may either precede or follow the numeric value. Some currencies allow decimal fractions while others use alternate methods of representing smaller monetary values.

Month and weekday names vary with language (if they are not omitted entirely). Abbreviations may be other than three characters, or may not be allowed at all. Even when a strictly numeric representation is used, the order of year, month, and day and the delimiters that separate them is not universal. For example, the date of October 12, 1986 is represented by 86-10-12 in Japan, by 10/12/86 or 10-12-86 in the U.S.A., and by 12/10/86 or 12.10.86 in Europe.

The HP-UX system clock runs on Greenwich Mean Time (GMT). Corrections to local time zones consist of adding or subtracting whole or fractional hours from GMT. For example, the Cook Islands are 10 hours, 38 minutes behind GMT and Singapore is 7 hours, 30 minutes ahead of GMT. Daylight saving time (DST) in the United State and similar summer time zone adjustments in other countries compound the challenge of knowing the correct time. Each nation selects days to begin and end summer time zone adjustments or whether to institute them at all during a particular year.

Messages. The need for messages to be readable by users is perhaps the most significant justification for implementing native language support. As pointed out by Bale and Kellogg,[9] not only do the words change from one language to another, but the ordering of the words within phrases may be different.

The user language is selected at run time by setting the value of the LANG environment variable. This string value may take on any value shown in the second column of the following table:

0	n-computer	native computer
1	american	American English
2	c-french	Canadian French
3	danish	Danish
4	dutch	Dutch
5	english	British English
6	finnish	Finnish
7	french	European French
8	german	German
9	italian	Italian
10	norwegian	Norwegian
11	portuguese	Portuguese
12	spanish	European Spanish
13	swedish	Swedish
41	katakana	phonetic Japanese

While it is clearer to specify the language in a string, it is often easier to store and manipulate a numeric value within a program. A family of routines has been added to the HP-UX system to access the current language and transform the string value to a numeric value and back.

For each of the languages listed, there is a directory which contains all of the information unique to that language. For example, in the directory /usr/lib/nls/french are located tables for NLS library routines and message catalogs for commands that provide native language support. For the German user, there is another directory /usr/lib/nls/german which has the same structure as the French directory. Only the string german in the path name and the contents of the various files differentiate the German directory from the French directory. Thus the software in the base HP-UX product can uniformly access information for either French or German users.

The string value is also used within a program to access the correct message catalog. For example, the pr command which formats lines of text for printing will look at the value of the LANG environment variable. Based on that value, it will open the message catalog in the proper directory so messages will be in the user's native language. As described earlier, the message catalog for German will appear as /usr/lib/nls/german/pr.cat and that for French will have a path name of /usr/lib/nls/french/pr.cat.

The HP-UX base product includes a number of commands ported from UNIX System V, 4.2BSD, and other sources, including HP. NLS has been incorporated into a number of these commands. For example, as mentioned earlier, the HP-UX version of the pr command is able to handle Italian and other languages while the original version was limited to only ASCII characters and American English messages.

In many cases, a significant amount of development is required to extend the capabilities of a complex program. For example, the Bourne shell (the standard UNIX command interpreter) makes use of the eighth bit of each character that it processes. Therefore, the changes to allow 8-bit characters to be used in Bourne shell directives required a major overhaul of the program.

A number of new commands have been added to those from AT&T or the University of California at Berkeley to maintain and use the NLS facilities of the HP-UX operating system. One set of tools allows you to extract messages from existing C programs (findstr), replace selected messages with calls to the message catalog facility (insertmsg), and build from the extracted strings a message catalog (gencat). As the program continues to evolve, other messages may be added. However, the source for the message catalog can be extracted from the C program source code (findmsg) for retranslation of messages.

HPWindows/9000

HPWindows/9000 is the human interface for the HP-UX system running on the Series 300 family of computers. Its purpose is to provide the user with a convenient yet intuitive interface to the multiprocessing capabilities of HP-UX systems by allowing the creation, manipulation, and control of several graphics and terminal windows on a single bit-mapped display screen. Many of the concepts and models are the same as those found in other window systems.[10] However, there are some contributions special to HPWindows/9000.

These contributions include an architecture that runs entirely as a set of user processes without requiring modification of the kernel. The windows themselves allow complete occlusion where windows can overlap each other and programs can output to obscured windows. Likewise, operations such as movement and size alteration can be done to obscured windows. Applications can alternately choose to redraw previously obscured regions or have the window system do it automatically. The other major contribution is providing a wide range of accesses to the window system, including device emulators that allow existing applications to run without modification, commands for usual HP-UX control of the window system and libraries for output of characters, window control and manipulation, and control of various forms of input from the keyboard or locator.

The underlying architecture is a multiprocess model which is interconnected by two types of communication channels. The primary process, called the window manager, provides two services: support of the interactive user interface and management of resources. These resources include the keyboard and locator (mouse or tablet) devices, the screen area, and the necessary communication paths. The purpose of the secondary processes, called window types, is to emulate an alpha terminal or graphics display device. These window types serve as extensions to the window manager for doing such tasks as drawing the window border and assisting with interactive features unique to the window type. This collection of processes is interconnected by two types of communication channels: shared memory and pseudoterminal devices (pty).

Several HP-UX kernel features are used by HPWindows/9000. The primary communication channel is a pty in which all messages and asynchronous events are passed, either as read/write data, or as ioctl calls to the window manager. A normal terminal tty is a multipurpose device file with the kernel on one side controlling the physical device, and the user process on the other side communicating via the open, close, read, write, and ioctl system calls. pty device files are special in that they provide the features and interface of a tty device file, but user processes can communicate on both sides of the device file. The other communication

channel is shared memory where data is shared between the window manager and servers. Typically this data consists of window images and shared font images that must be consistent between all associated processes. Semaphores are used to ensure the integrity of the shared memory data by preventing other processes from reading a section of shared memory while another process is writing to it. Another necessary kernel feature is the select system call, which allows a process such as the window manager to multiplex I/O from several input devices and many pty files simultaneously, allowing it to respond to whichever channel has data. Since windowing is a real-time application (i.e., used for responsive tracking of the locator, routing of keyboard input, repainting windows after moves or sizes, etc.), real-time process priorities are necessary. The order of priority from highest to lowest is window manager, window types, and user applications.

Since the purpose of windowing is to provide a visual interface, it is only natural that HPWindows/9000 be built in conjunction with the Starbase Graphics Library. The Starbase Library uses a layered architecture with a high-level user interface based on ANSI's Computer Graphics Interface standard and a low-level device handler interface. It is at the device handler interface that windowing is done. Windows required two important extensions to the device handlers. One is the concept of layers, which are required to manage the window images. Layers not only can be obscured (covered by other layers), but also can be written to independent of occlusion conditions. Portions of obscured layers that are not visible either support output by writing to optional memory images or ignore output if repainting of these areas can be done by the window type or application. Whether or not repainting is handled by the user or system, there is no special knowledge required of obscured areas because they are handled by the device handler. The second extension to the device handlers is the addition of raster fonts. This capability allows different fonts such as Courier to be copied in various point sizes on the screen. Since windows are built using the Starbase Library, users can write programs that use either raw Starbase screen devices or the graphics window type by selecting the appropriate driver. HPWindows/9000 works on monochrome or color, low- or high-resolution bit-mapped displays.

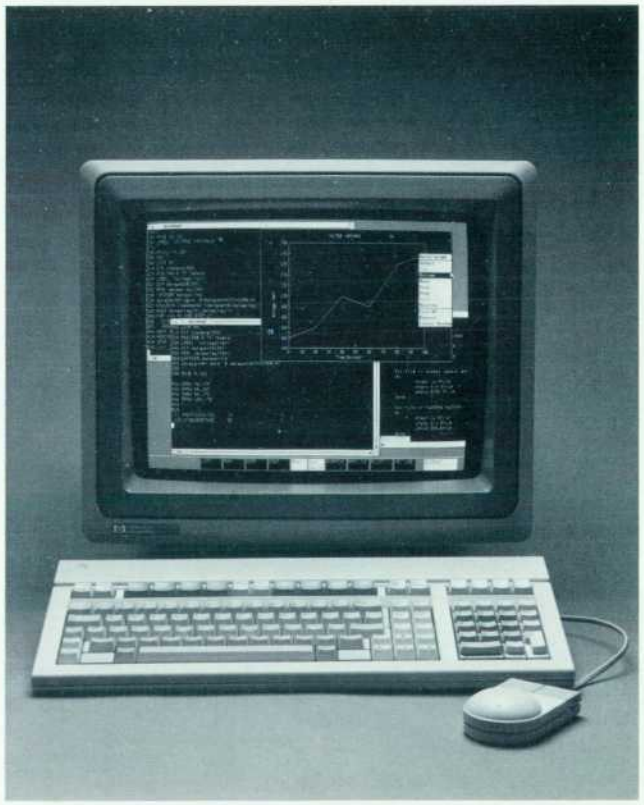


Fig. 3. *Typical display showing alpha and graphics windows on a Series 300 Computer running HPWindows/9000.*

The most natural way to use HPWindows/9000 is through its interactive user interface. The cognitive model it is based on is that of point-do (e.g., select window 1 or move window 3). These actions are done by picking sensitive symbols residing in the window border, each representing actions that can be performed on the window. Such actions include move, size, pause, scroll, or change the window's appearance. The action of associating the keyboard to a window has no touch-sensitive symbol; this is done by selecting the interior area of the window. Changes to the echo representing the locator's position on the screen informs the user that a touch-sensitive area can be picked. A window can be represented in two forms—normal, which is the usual full window display, or iconic, which are small graphic representations of the window or the application running in the window.

There are two types of menus—system and user defined. The system menu not only contains items for interactive operations such as move, but also provides the items to change the window's obscurity, create a new window, destroy a window, or exit the window system. The window system's interactive user interface can be customized to various application requirements (i.e., altering or limiting its capabilities).

Window types are an integral part of HPWindows/9000 because they emulate devices normally accessed by user applications. Window types allow applications that usually deal directly with the device to run in the window system without modification. The first of two window types which are part of HPWindows/9000 is an alpha terminal, which emulates the features of an HP 2622 Terminal, except for its block and format modes, and adds the color capabilities found in an HP 2627 Terminal. The second window type is a graphics window, which provides access via the Starbase Library as if the window were a Starbase display device.

Commands are provided for easy access to the window system from the shell programming environment. Such access includes the creation, deletion, listing, appearance, and manipulation of windows, changing of fonts, and control of the window system.

The library interface provides full programmability of the windowing system. Users not only have access similar to that of the above commands, but they also can control icons, signal detection, and handling of window events, menus, locator input, and echo control.

Fast alpha and font manager library routines are provided

to allow the user to place raster text at arbitrary locations on the display screen or in a window. This capability allows the combining of text and graphics on a single device. Both libraries access the low-level drivers directly for maximum speed and cooperate fully with the window environment. The fast alpha text locations are specified by character column and line. An environment is created so that calls to the various routines cooperate with one another and color and character enhancement information are handled automatically for the user. Intensive text manipulation applications like terminal environments are easily created using the fast alpha scrolling, writing, and cursor manipulation routines. The font manager places text according to a given pixel location. Fonts can be mixed freely with different styles and sizes.

Acknowledgments

The authors would like to thank the many individuals who contributed to the work described here. M.P. Dunn, Hideyuki Hasegawa, Bill McMahon, and our managers, Joe Cowan and Jeff Lindberg, were part of the effort for the Device I/O Library. Xuan Bui, David Gutierrez, Sol Kavy, Gregg Kellogg, Doug Larson, Dave Lennert, Pam Marshall, Donn Terry, and our managers Gary Ho, Steve Kusmer, Dave Landers, and Scott Wang worked on the definition, implementation, and testing of the real-time extensions to the HP-UX system. Mark Laubach, Radek Linhart, and Hal Prince were involved in the X.25 implementation. Duncan Missimer, Nobuhisa Takahashi, Tai-Jue Sue, and Antoni Drudis led by Karen Barnes and Itaru Abe, Takumi Ohtani, and Toshihisa Aoki led by Sotoji Watanabe contributed to the native language support effort, along with many others worldwide. Jack Applin, Keith Fish, Bob Hallissy, Greg Lawson, J.L. Marsh, David Pinedo, Steve Scheid, Michael Schoonover, Alan Silverstein, Jeff Stevenson, and Evelyn Williams, the Integral PC windowing people, numerous understanding and thought provoking users, and our manager Joe Gersch were important to the work on HPWindows/9000. Although we cannot mention everyone by name, practically all of the members of the software lab at HP's Fort Collins Systems Division were somehow involved in the success of the HP-UX system on the Series 300.

References

1. M.V. Hetrick, "HP-UX: A Corporate Strategy," *Hewlett-Packard Journal*, Vol. 35, no. 3, March 1984, pp. 12-13.
2. M.L. Connor, "What is UNIX?," *ibid*, p. 9.
3. S.W.Y. Wang and J.B. Lindberg, "HP-UX: Implementation of UNIX on the HP 9000 Series 500 Computer Systems," *ibid*, pp. 7-15.
4. R.M. Fajardo, et al, "A UNIX Operating System Adapted for a Technical Personal Computer," *Hewlett-Packard Journal*, Vol. 36, no. 10, October 1985, pp. 22-28.
5. J.J. Balza, et al, "A Local Area Network for the HP 9000 Series 500 Computers," *Hewlett-Packard Journal*, Vol. 35, no. 3, March 1984, pp. 22-27.
6. J.L. Bidwell and D.W. Palermo, "Advanced Multilingual Computer Systems for Measurement Automation and Computer-Aided Engineering Applications," *Hewlett-Packard Journal*, Vol. 33, no. 5, May 1982, pp. 3-7.
7. R.M. Lenk, "Real-Time Functionality in HP-UX," *TC Interface*, Vol. 5, no. 1, January/February 1986.
8. V.C. Jones, "Data Communications for a 32-Bit Computer Workstation," *Hewlett-Packard Journal*, Vol. 35, no. 3, March 1984, pp. 24-25.
9. J.E. Bale and H.E. Kellogg, "Native Language Support for Computer Systems," *Hewlett-Packard Journal*, Vol. 36, no. 6, June 1985.
10. J.A. Brewster, et al, "A Friendly UNIX Operating System User Interface," *Hewlett-Packard Journal*, Vol. 36, no. 10, October 1985, pp. 28-33.

CORRECTION

The following paragraph was missing in last month's issue from the end of James Chen's article on page 34.

Continuing Development

The science of ultrasound power and intensity measurement is rapidly evolving and the associated regulatory standards are continually being updated to reflect the latest state of the art. As a result, many changes and improvements have been incorporated into HP's ultrasound power and intensity measurement program since this article was first written and we are continuing to upgrade our measurement process to comply with current regulatory standards using the latest techniques and equipment.

A Protocol Analyzer for Local Area Networks

This new analyzer allows 10 Mbit/s network monitoring, testing, and performance analysis independent of hardware and software composition. It permits a user to view network traffic, simulate node-to-node or network-wide traffic, and derive network statistics.

by Gordon A. Jensen, Stephen P. Reames, Jerry D. Morris, Jeffrey H. Smith, Jeffrey Tomberlin, and James M. Umphrey

MOST USERS of local area networks (LANs) expect their computers and peripherals to exchange data freely through standard Ethernet/IEEE 802.3 interfaces. When the system is composed entirely of equipment from a single vendor such as HP, this is just what they should expect. They know that if the network doesn't perform properly, HP will supply the support services needed to solve the problem.

Unfortunately, few users are able to satisfy all of their networking needs with products supplied by a single vendor. Instead they must choose from a broad spectrum of products from a variety of vendors. Thus it is not at all unusual for users to connect equipment from two or more vendors to the same LAN only to discover that some incompatibility prevents them from working together. Even though the equipment uses correctly designed and operating LAN interfaces, the higher-level protocols often just won't communicate with each other.

Who can users call when two pieces of equipment from different vendors don't communicate? In most cases neither vendor will take on the problem; it will be up to the users to solve it themselves.

Another problem with multivendor networks is network management. Most LAN vendors can supply software that is reasonably adequate for managing the portion of the network supplied by them. Typically each node contains software that allows it to compute statistics on its own activities and report these back to a master station. Because each vendor uses a different system for reporting, however, no single vendor can gather and present statistics for the entire network.

The HP 4971S LAN Protocol Analyzer (Fig. 1) was developed to solve these problems. It is dedicated to monitoring and simulating a network and gathering data for network management independently of the hardware and software that make up the network.

Overview of the HP 4971S

For a protocol analyzer to be capable of trapping and analyzing all of the network traffic for a 10-megabit/s LAN, dedicated high-speed hardware is essential. In the HP 4971S, this hardware includes a LAN interface processor, two hardware trap machines or filters, a high-speed 1M-byte two-port RAM buffer, and a separate LAN transmitter (see Fig. 2).

The LAN interface hardware is based on the Intel 82586 LAN coprocessor. This chip enables the analyzer to capture all traffic on the network independent of address or FCS (frame check sequence) status. Additional hardware was designed around this interface to ensure that the analyzer can capture and process all network traffic in real time under any loading conditions.

Many LAN interfaces can miss packets when large numbers of small frames (<30 bytes) are received at minimum interframe spacing. These small frames are usually collision fragments (runts) found on extremely heavily loaded networks approaching maximum network span. To avoid

Fig. 1. *The HP 4971S LAN Protocol Analyzer is designed for local area network troubleshooting and management. It can monitor and simulate network traffic and gather data independently of network hardware and software.*

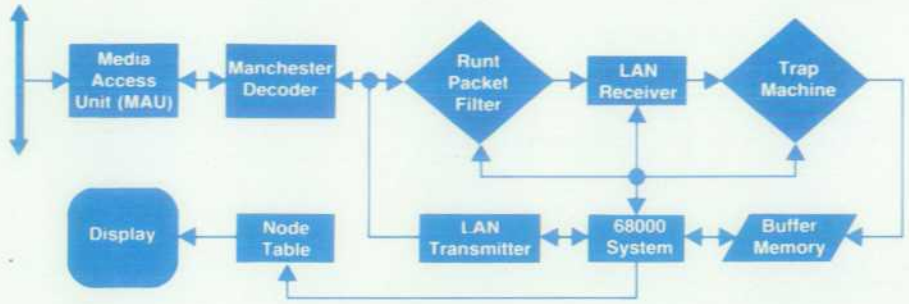


Fig. 2. *HP 4971S block diagram. Dedicated high-speed hardware is provided to trap and analyze all traffic in a 10-Mbit/s LAN.*

this potential problem, the HP 4971S incorporates a custom runt packet filter that removes packets less than 512 bits long. This circuit filters the serial data stream between the LAN attachment unit interface (AUI) and the 82586.

Capturing bad frames on the network permits identification and analysis of errors occurring at levels 1 and 2 of the ISO OSI model, such as marginal connections and MAU (media access unit) failures. Collision counting hardware also permits the user to monitor one of the important indicators of network loading. Although normally only significant at network loads exceeding 40%, the collision rate can indicate faulty hardware on the network or let the system manager know when it may be time to reconfigure the network.

Because of the high data rate and the potentially large numbers of independent logical channels all operating at once, a filtering mechanism that passes only frames of interest to the user is essential. For example, if an IEEE 802.3 network is loaded at 10% of capacity, one megabyte of data appears on the network every eight seconds. Since a conversation between two nodes of interest may take several minutes, collecting all of the data on the network to study that one conversation would be impractical. The filtering function allows the HP 4971S to retain in its capture buffer only those frames that meet a predefined specification.

When a frame is received by the HP 4971S, it is loaded into the capture buffer in a linked list data structure. This loading process also passes the data through some special-purpose pattern recognition hardware called the trap machine. The trap machine outputs the results of a comparison with sixteen different patterns up to 61 bytes long to the system microprocessor. If at least one of the sixteen patterns is matched by the received frame, the system microprocessor knows that this is a frame that the user wishes to keep and no action is taken. If, on the other hand, the received frame does not match any of the predefined patterns, the processor deletes the frame. It does this by manipulating pointers in the linked list to remove the buffers taken by the frame. It places these buffers at the end of the free list where they can be reused. When the HP 4971S operates in a circular buffer mode to observe data continuously, the processor keeps a constant number of buffers between the head and the tail of the list by reallocating the buffers in place.

The capture buffer is a 1M-byte dual-port RAM. The receive hardware writes the captured data into one port and the system microprocessor does its pointer manipulations through the other port. This structure is required because of the high data rate; a single bus cannot provide the needed bandwidth. All received frames are loaded into the capture buffer. This means that if a user does wish to look at every single frame around a trigger point, for example, then the filter mechanism is simply not used.

Another useful feature of the HP 4971S is the ability to view frames as it transmits them onto the network. Because an IEEE 802.3 network is not deterministic, it is impossible to know exactly when a given frame will be sent after a program executes a send message statement. The ability to see the analyzer's transmit frames removes all ambiguity from this situation, allowing the relative timing of transmit and receive frames to be observed during a simulation. This feature requires a separate transmitter and receiver in the instrument, although only one transceiver is needed for attachment to the network.

Software Features Enhance Ease of Use

To minimize the complexity of LAN protocol analysis, the HP 4971S software has been designed with ease of use as a primary goal. Using softkeys to select major functions eases the task of instrument control, while screen editing capability aids data entry and presentation.

To get a novice user started quickly, a two-button network monitoring function is provided. As shown in Fig. 3, the user needs only to select the Monitor Network and Start Monitor softkeys to begin capturing and viewing data on the LAN. Snapshots of the data are displayed as it is being stored in the capture buffer. Once the buffer is full, control transfers automatically to the Examine Data menu, where the user can easily scroll through all of the frames stored.

When dealing with a packet-switched network capable of supporting hundreds of nodes, the difficulty of identifying a single node or connection quickly becomes apparent. Although each node has a unique six-byte address which is clearly displayed in the source and destination fields of a frame, not many users can remember more than a few of these addresses or the devices they represent. To help the user with this problem, the HP 4971S provides a node name table. With it, each node address can be assigned an alphanumeric name to describe the device and/or location on the net. Once assignments have been made, the user has the option of viewing and entering addresses by name throughout the HP 4971S menus.

As previously indicated, it is often desirable to view only a subset of the total network traffic. The dedicated hardware trap machines make this possible, but it is up to the user to specify which subset of the traffic is of interest. This

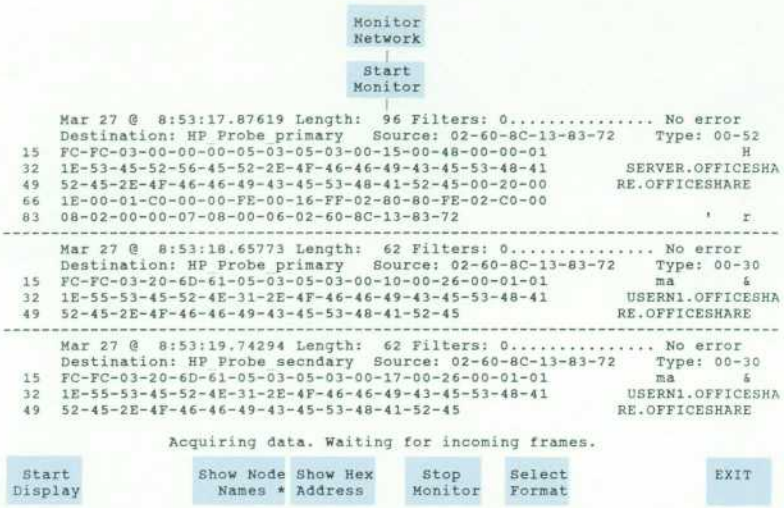


Fig. 3. *Selection of the Monitor Network and Start Monitor softkeys produces a display of network traffic.*

process of data filter specification is made easy by the Edit Filters menu.

A data filter is analogous to a frequency-domain bandpass filter. Only input that is compatible with the parameters of the filter is able to pass through. The HP 4971S data filters are used to eliminate traffic that is not of interest and to select among the frames that remain.

Up to 16 data filters can be defined and used simultaneously. The filters are created by using a softkey-driven filter definition menu. This menu is also used to name the newly created filters and to enter values into their most commonly used fields (see Fig. 4). The 6-byte Ethernet/IEEE 802.3 node addresses can be entered in hexadecimal format, or by using softkeys that display names from the node name table, or by entering node names directly from the keyboard. To permit all defined data filters to be displayed together, this menu does not display the filter data fields.

To use the data filters for capturing frames based on the values of higher-level protocol fields, it is necessary to specify subfields within the Ethernet/IEEE 802.3 data field. An additional menu was designed to allow the user flexibility in specifying these fields (see Fig. 5). The HP 4971S provides 47 byte positions to be defined in the frame data field. These bytes can be located anywhere in the data field and can be formatted into as many as thirteen separate subfields in each data filter. For convenience, subfields can be named. Within each subfield, bytes can be entered and viewed using ASCII, EBCDIC, hexadecimal, or binary data codes. In binary, both the least-significant-bit-on-the-right and the least-significant-bit-on-the-left modes are available. In the hexadecimal mode, individual half bytes can be set to "don't care" and in the binary modes individual bits can be set to "don't care." In all data codes, bytes can be set individually to "anything except" the quantity entered (i.e., the logical NOT of the quantity entered). Thus, for example, a filter can be set to accept any character that is not an AK control character in a given byte position.

A third filter definition menu called the frame traits menu allows the HP 4971S to filter on the presence or absence of some frame reception errors. Frames can be accepted or rejected if they have a good or bad frame check sequence (FCS), if they are misaligned (the FCS is bad and the number of bits in the frame is not evenly divisible by eight), or if they are runt frames (they contain fewer than 64 bytes). In addition, jabbering frames (frames containing more than 1518 bytes) can be recognized by using the Minimum Length and Maximum Length fields in the filter definition menus.

The operator can use a similar set of menus to define up to sixteen messages, which can be used by a softkey-driven programming language to put user-defined traffic onto the network. An overview menu handles the Ethernet/IEEE 802.3 header fields and permits selection of the message length and FCS. The length of each message can range from

Fig. 4. *The filter definition menu is used to define and name filters.*

```
FIELDS of FILTER #0   Label: 802_TCPIP_SYNACK        MINIMUM Frame Length =   57
      4 Filter hardware bytes available              MAXIMUM Frame Length = 2022
 BYTE       FIELD LABEL                    FIELD DATA
 ----   -----------------   ------------------------------------------
    1   DESTINATION         XX-XX-XX-XX-XX-XX       Node Name: -- Not Defined --
    7   SOURCE              XX-XX-XX-XX-XX-XX       Node Name: -- Not Defined --
   13   TYPE                XX-XX
   15   DSAP_SSAP_Control   06-06-XX
   18   TCPIP_SynAck        45
   19   IP_Type_Service     XX
   21   IP_Identification   XX-XX
   24   IP_Flgs_Frgt_Ofst   XX-XX
   26   IP_T_live_Protocl   XX-XX
   28   IP_Checksum         XX-XX
   30   IP_Srce_Dest_addr   XX-XX-XX-XX-XX-XX-XX-XX
   38   TCP_Srce_Des_addr   XX-XX-XX-XX
   42   TCP_Seq_number      XX-XX-XX-XX
   46   TCP_Ack_number      XX-XX-XX-XX
   50   TCP_Data_Ofst_Cfg   XX-XXX1XX1X
   52   TCP_Wndw_Cks_UPtr   XX-XX-XX-XX-XX-XX
              Hexadecimal (0..9, A..F or X) field data entry

Bin LSB Left | Bin LSB Right * | Hex Entry | ASCII 8 Entry | EBCDIC Entry | OTHER CHOICES | EXIT
```

Fig. 5. *A menu is provided to allow the user to specify subfields within the Ethernet/IEEE 802.3 data field.*

one byte to 2022 bytes. If the message length is less than 64 bytes or more than 1518 bytes, it is not a legal Ethernet/IEEE 802.3 frame, so a warning message is displayed. A message can be sent with a good, bad, or selected FCS.

A detail menu expands the data field, if any, of any message that has been defined. The data field contents can be entered from the keyboard using ASCII, EBCDIC, or hexadecimal data codes. A useful feature allows any defined message or any frame in the receive buffer to be copied and edited. Individual bytes can also be inserted or deleted.

A third message menu permits any message to be displayed and edited using the subfield labels, positions, and lengths defined for any filter. This greatly facilitates the entry of values into the fields of higher-level protocols.

Because of the amount and complexity of the traffic that the HP 4971S can collect from a network during a run, particular attention was devoted to ensuring that frames can be displayed in formats that are meaningful to the user. The entire frame, from the first byte of the header through the last byte of the data field, can be viewed, regardless of its length. Address fields from the Ethernet/IEEE 802.3 header can be displayed in hexadecimal code or by using the equivalent node names as defined in the node name table. The Ethernet/IEEE 802.3 data field can be displayed using ASCII-7, ASCII-8, EBCDIC, or hexadecimal character codes. A 32-bit time stamp with a resolution of 32 microseconds is stored with each frame when it is received. Several time stamp display formats are possible, including time of day and time from the end of the previous frame. Additional fields identify the filters that accepted each frame and any error conditions that may have been detected during reception.

If the lengths of the frames permit, multiple frames are displayed simultaneously on the screen, so that a frame can be viewed in the context in which it was received. Optional suppression of the data field, the Ethernet/IEEE 802.3 header, and/or the physical frame data can be used to permit more frames to be displayed simultaneously. Additionally, a one-line summary format is available that shows only the node addresses, frame length, filters matched, and reception errors.

Support for higher-level protocols is also available in the display of received frames. A Filter Format display mode displays received frames using the subfield format and labels that match the filter that accepted it. Key protocol fields can be easily identified in this way.

For complex troubleshooting and traffic simulation, a softkey-driven programming language is included. Based upon the algorithmic language designed for the HP 495X Protocol Analyzers,[1] it provides even inexperienced programmers with the means to automate testing and simplify data analysis.

At the head of the program is the buffer control command line. With it, the user selects which frames (if any) to collect in the receive buffer when the program is executed. The data filters can be accessed here for both filtering and triggering. The filtering process is enabled by choosing to Store frames matching any subset of the 16 defined data filters.

Start	Stop	Increment Counter	If Counter	Go to Block	When (event)	OTHER CHOICES	END EDIT
Frame Matches		Timer	Frame Counter	Collision Counter			END EDIT
Sequence_d_Packet	Packet_Exchange	Routing Info Req	Routing_Info_Resp	Echo_Req	Echo_Reply	OTHER CHOICES	END EDIT
Then Go To Block		And	And Frame Traits	Or			END EDIT
		Else When		‹ Next Block ›			END EDIT
Start	Stop	Increment Counter	If Counter	Go to Block	When (event)	OTHER CHOICES	END EDIT
Display Frame	Mark Frame	Beep	Send Message	Reset	Wait	OTHER CHOICES	END EDIT
			And Then	‹Next Block›			END EDIT
Start	Stop	Increment Counter	If Counter	Go to Block	When (event)	OTHER CHOICES	END EDIT

Block 1:
When frame matches Routing_Info_Req then go to block 2

Block 2:
Display frame and then go to block 1

Fig. 6. *A program to display only those frames that match a single filter can be written with the few keystrokes shown.*

Triggering options are starting with, centered about, or ending with any subset of the 16 defined data filters, although the user may also choose simply to store until full or nonstop.

The remainder of the program is used to manipulate the received data. By softkey choices, a great variety of programs can be easily constructed. For example, a program to display only those frames matching a single filter can be written with the few keystrokes shown in Fig. 6. This program can be used to display only frames sent by a node while storing frames sent and received by it, or to locate those frames in the receive buffer during postprocessing. More complex programs can measure time intervals using up to sixteen timers, count frames or collisions using up to sixteen counters, and transmit frames using the sixteen messages defined in the Message menu. Programs can be written to run in real time so long as the average traffic level through the filters is below the typical maximum network loading of 40%. For traffic levels above this, data will still be captured in real time but must be analyzed by postprocessing.

For preserving the node name table, data filters, messages, programs, and captured data, a Disc Functions menu is provided. Configuration parameters can be saved as network files, and data from the receive buffer can be selectively stored in data files. This information can be loaded back at any time for additional network testing or data postprocessing. Both hard and flexible discs are supported, with a maximum on-line storage capacity of 40 megabytes.

Problem Solving

Users are faced with protocol incompatibilities almost every time they connect equipment from different vendors to the same network. This may happen because the vendors have interpreted the same protocol specification differently or because one vendor has defined an entirely new protocol that simply will not communicate with that of another vendor. It could even be that one device has a software bug that only appears when interacting with another device through the LAN. Finding these problems can be a real challenge, but the HP 4971S makes the job easier.

Suppose, for instance, that a LAN manager connects a new personal computer to a network and suddenly starts getting protocol errors reported by one of the host computers on the network. The PC seems to work fine with other devices on the network, and other PCs from the same manufacturer work with the host, so what is the problem? Using the HP 4971S, a frame from one of the "working" PCs to the host can be captured and compared to a like frame from the "bad" PC. The Filter Format option breaks out the TCP IP protocol fields and makes the problem clear (see Fig. 7). The bad PC is somehow setting its TCP window size to FE-00, a negative number! Further investigation might reveal that the new PC came with a new revision of software, which contains a previously undiscovered bug.

More obscure errors can be located using the special protocol analysis programs. It could be that a bad frame appears on the network at seemingly random times for unknown reasons. Even when the source of the bad frame has been identified, the cause for its unexpected transmission may not be understood. To discover why the error occurs, the user can write a program to store all frames to and from the problem node centered about the appearance of the bad frame. Since the error occurs infrequently, the user can also tell the program to beep when the error is detected. The next time the bad frame appears, the HP 4971S will present a trace of the frames before and after it. With this information, cause and effect can be clearly seen.

In addition to protocol analysis, the HP 4971S can be used to investigate network performance problems. If there is reason to think that a network may be overloaded during certain peak times, a program might be written to measure traffic over one-hour periods for the fourteen prime shift hours of the day (see Fig. 8). The network manager starts the program when beginning work at 6:00 a.m. The next morning, the traffic history of the previous day is waiting. From this data the manager can determine the peak use times of the network and begin an investigation of the traffic during those times. Once the bottleneck is discovered, appropriate action can be taken.

Now suppose that the traffic measurement indicates that there is no general overloading of the network at any time, but the users are still complaining about the response time

```
#1    Dec  5 @  9:58:44.40438 Length:  60 Filters: xxxxxxxxxxxxCxxF No error
      Destination: 08-00-20-01-15-1A  Source: Working_PC          Type: 08-00
TCPIP_Ack_Fld_Sig  45                                            E
IP_Type_Service    00
IP_Identification  B8-B1
IP_Flgs_Frgt_Ofst  00-00
IP_T_live_Protocl  0F-06
IP_Checksum        41-1C                                         A
IP_Srce_Dest_addr  59-00-00-01-59-00-00-02                       Y   Y
TCP_Srce_Des_addr  08-AE-04-6E                                     n
TCP_Seq_number     01-D1-71-C2                                   q
TCP_Ack_number     01-07-79-82                                   y
TCP_Data_Ofst_Cfg  50-10                                         P
TCP_Window_Chksum  02-00-00-99
TCP_Urg_pointer    00-00
================================================================================
================================================================================
#2    Dec  5 @  9:58:44.43949 Length:  60 Filters: xxxxxxxxxxxxCxxF No error
      Destination: 08-00-20-01-15-1A  Source: Bad_PC             Type: 08-00
TCPIP_Ack_Fld_Sig  45                                            E
IP_Type_Service    00
IP_Identification  B8-B4
IP_Flgs_Frgt_Ofst  00-00
IP_T_live_Protocl  0F-06
IP_Checksum        41-19                                         A
IP_Srce_Dest_addr  59-00-00-01-59-00-00-02                       Y   Y
TCP_Srce_Des_addr  08-AE-04-6E                                     n
TCP_Seq_number     01-D1-71-C2                                   q
TCP_Ack_number     01-07-7B-02                                   {
TCP_Data_Ofst_Cfg  50-10                                         P
TCP_Window_Chksum  FE-00-03-18
TCP_Urg_pointer    00-00
================================================================================
```

Fig. 7. *Using the data filters to capture and compare good and bad frames reveals that a protocol problem is caused by attempting to select a negative window size.*

of a particular host. The network manager can write a simple simulation program to send an XID message to the host and measure the time it takes to get a reply (see Fig. 9). If the measurement indicates that there is indeed a slow response, then further investigation may be required. It could be that the host software puts LAN communications at too low a priority and that some optimization can be done. In a multinetwork environment, it may be necessary to readjust the internet routing tables.

Another tool for network problem-solving with the HP 4971S is the remote testing option. This gives the user the ability to test a remote network using the same features that are available for testing the local network. For example, a user can remotely run a program that looks for frames with bad FCS fields. This program would actually be running on a slave HP 4971S attached to the remote network, but the softkeys and results would appear on the master HP 4971S display just as if the program were running on the master unit.

In remote mode, the master and slave HP 4971Ss communicate over an RS-232-C/V.24 link. This link will usually be over a dial-up phone line, but it could be a leased line or even a direct connection if the slave HP 4971S is located close to the master. To ensure error-free communication between the master and the slave, one of two protocols is used: DDCMP or encoded DDCMP.*

More than one slave HP 4971S can be controlled by a master. For example, someone at site 1 could use an HP 4971S as a master to make the slave units at sites 2 and 3 start executing standard test programs that collect some statistical information of interest for their networks.

While the slave units are collecting data, the unit at site 1 can be used locally for some management or diagnostic purpose or as a master to control a slave on some other network. When this task is complete the unit at site 1 can once again become a master to examine the statistics that have been collected by the slave units at sites 2 and 3.

Conclusion

In summary, the ability of the HP 4971S to acquire any network frame independent of the network vendor(s) and network traffic conditions provides the capability for network troubleshooting and management. The softkey-controlled system is extremely user friendly, providing flexibility through an easy-to-use programming language. These features allow straightforward debugging of network hardware and Ethernet/IEEE 802.3 protocols. With relatively simple programs, the user can debug higher-level protocols and gather statistics for use in managing a local area network.

Acknowledgments

Many people played an important part in the success of the HP 4971S. The initial design efforts were led by Dave Bass and Louis Nagode. Jim Quan wrote the run-time code, Joe Peck wrote the software for displaying frames and constructing the node name table, Scott Neal designed the disc function interface and the redirected I/O code, and Rona

*DDCMP is a protocol standard developed by Digital Equipment Corporation.

```
Store: all frames
       nonstop

Block 1:
     Start timer Tot_Time
       and then
     Start frame counter Tot_Frame
       and then
     Start collision cntr Tot_Coll
       and then
     Start timer 1st_hour

Block 2:
     When timer 1st_hour exceeds 3600 seconds then go to block   4
      else when frame matches Any_Frame then go to block   3

Block 3:
     Increment counter 6am_7am
       and then
     Go to block   2

Block 4:
     Stop timer 1st_hour
       and then
     Start timer 2nd_hour

Block 5:
     When timer 2nd_hour exceeds 3600 seconds then go to block   7
      else when frame matches Any_Frame then go to block   6
                                        .
                                        .
                                        .
================================================================================
================================================================================

              COUNTERS                          TIMERS
     ---------------------------------   ----------------------------
     Tot_Frame =  6,671,914              Tot_Time  =  50,400.02028 s
     6am_7am   =     57,585              1st_hour  =   3,600.00168 s
     7am_8am   =    236,455              2nd_hour  =   3,600.00142 s
     8am_9am   =    475,488              3rd_hour  =   3,600.00146 s
     9am_10am  =    672,824              4th_hour  =   3,600.00142 s
     10am_11am =    850,052              5th_hour  =   3,600.00138 s
     11am_12n  =    452,739              6th_hour  =   3,600.00146 s
     12n-1pm   =    341,940              7th_hour  =   3,600.00142 s
     1pm_2pm   =    967,774              8th_hour  =   3,600.00142 s
     2pm_3pm   =    912,371              9th_hour  =   3,600.00138 s
     3pm_4pm   =    827,459              10th_hour =   3,600.00142 s
     4pm_5pm   =    521,978              11th_hour =   3,600.00138 s
     5pm_6pm   =    213,373              12th_hour =   3,600.00146 s
     6pm_7pm   =    121,759              13th_hour =   3,600.00142 s
     7pm_8pm   =     20,117              14th_hour =   3,600.00138 s
     Tot_Coll  =        217              xxxxxxxxx =          0.00 ms

================================================================================
```

Fig. 8. *To check for network overloading during peak hours, a program can be written to measure traffic over one-hour periods for the fourteen prime shift hours of the day.*

```
Store: frames matching XID_Command
       or XID_Response
       nonstop

Block 1:
     Send message XID_Command_Indiv
       and then
     When frame matches XID_Command then go to block   2

Block 2:
     Start timer Duration
       and then
     When frame matches XID_Response then go to block   3

Block 3:
     Stop timer Duration
       and then
     Start display
       and then
     Stop test

================================================================================
================================================================================

                COUNTERS                          TIMERS
        ---------------------------------   ---------------------------
        Counter_0 =          0              Duration  =        215.17 ms
        Counter_1 =          0              Timer_1   =          0.00 ms
        Counter_2 =          0              Timer_2   =          0.00 ms
        Counter_3 =          0              Timer_3   =          0.00 ms
        Counter_4 =          0              Timer_4   =          0.00 ms
        Counter_5 =          0              Timer_5   =          0.00 ms
        Counter_6 =          0              Timer_6   =          0.00 ms
        Counter_7 =          0              Timer_7   =          0.00 ms
        Counter_8 =          0              Timer_8   =          0.00 ms
        Counter_9 =          0              Timer_9   =          0.00 ms
        Counter_A =          0              Timer_A   =          0.00 ms
        Counter_B =          0              Timer_B   =          0.00 ms
        Counter_C =          0              Timer_C   =          0.00 ms
        Counter_D =          0              Timer_D   =          0.00 ms
        Counter_E =          0              Timer_E   =          0.00 ms
        Counter_F =          0              Timer_F   =          0.00 ms
```

Fig. 9. *To test the response time of a particular computer, a program can be written to send an XID loopback message to the computer and measure the time it takes to get a reply.*

Prufer led the software QA process. Bill Marbaker headed the service engineering tasks, Alan Bratshun did the production engineering, and Gary Roberson and Karen Moulton wrote the manuals and tutorial. Dave Couch, Belinda Yung-Rubke, and Dave Novotny contributed to the product definition. Special thanks to Justin Morrill for his many ideas and general support throughout the product's development.

Reference

1. *Hewlett-Packard Journal*, Vol. 36, no. 7, July 1985, pp. 4-29.

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

HEWLETT-PACKARD JOURNAL

July 1986 Volume 37 • Number 7

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 3000 Hanover Street
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529, Startbaan 16
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

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

CHANGE OF ADDRESS: To subscribe, change your address, or delete your name from our mailing list, send your request to Hewlett-Packard Journal, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Include your old address label, if any. Allow 60 days.

5953-8550