HEWLETT·PACKARD JOURNAL

OCTOBER 1985

HEWLETT·PACKARD JOURNAL

October 1985 Volume 36 • Number 10

Articles

Editor, Richard P. Dolan • Associate Editor, Kenneth A. Shaw • Assistant Editor, Nancy R. Teater • Art Director, Photographer, Arvid A. Danielson • Support Supervisor, Susan E. Wright
Illustrator, Nancy S. Vanderbloom • Administrative Services, Typography, Anne S. LoPresti • European Production Supervisor, Michael Zandwijken • Publisher, Russell M. H. Berg

In this Issue

This world has its share of problems whose solutions, if any exist, have so far eluded the collective mind of humanity. On a smaller scale, however, the picture is brighter. In the development laboratory, for example, creative problem solving is alive and well, the routine rather than the exception. I got this message very strongly from the articles in this issue, which cover the design of the HP Integral Personal Computer, a transportable machine that runs a version of AT&T Bell Laboratories' UNIX™ operating system. Another PC, yes, but with enough differences that a parade of problems were encountered for the first time and solved in ways that make interesting reading.

The UNIX operating system was born in the early 1970s at Bell Laboratories. By 1983, according to one study, there were about 100,000 UNIX systems in operation. By 1987, this study projects that there will be about 2,500,000 such systems. There seem to be two major reasons for this popularity. First, the UNIX system isn't tied to any particular computer architecture, so software developed using its extensive facilities can be ported easily to many computers. Second, it is a powerful multitasking system, able to run many applications at the same time.

But putting a UNIX system into a transportable package wasn't trivial. In the operating system article on page 22, for example, we read that UNIX systems use a lot of disc memory, but since a big disc drive is hard to carry around, the Integral PC's designers were restricted to a small flexible disc. Their solution? Emulate a hard disc in main memory. Call it a RAM disc. Use the flexible disc for removable file storage. Another problem: UNIX systems buffer disc writes for a time before updating the disc. What happens if someone removes the disc before the buffered data is written to it? The solution? Update the disc immediately without buffering so it can be removed at any time without losing data. Other operating system challenges included extending the UNIX kernel to provide real-time services for data acquisition and control, essential in an HP technical computer, and developing a BASIC language subsystem to run on top of the UNIX operating system so that HP customers' investments in software for HP Series 80 Computers wouldn't be lost.

The need for a compact package pointed to some sort of flat display instead of a CRT, but there were none that met the needs. The solution? Work with manufacturers of electroluminescent displays to develop sources of suitable displays (see page 12). Develop a resonant drive system to save power. Develop a custom graphics processor to simplify display control (page 10). Make sure the package is rugged, since it's certain to be knocked around (page 18). To make the multitasking operating system easily accessible to the user, develop a user interface that includes windowing capabilities and an easy-to-use applications manager (page 28).

You'll also find an introduction to the Integral PC on page 4 and an article on its electronic design on page 6. Our cover emphasizes the Integral PC's electroluminescent display, its windows, and its transportability.

-R.P. Dolan

What's Ahead

The application of thin-film technology to improve the density and reliability of disc memories is the subject of next month's issue. Authors from HP Laboratories and the Disc Memory Division will describe the development and evaluation of thin-film magnetic disc technology at HP.

A Multitasking Personal Computer System for the Technical Professional

The Integral PC provides high-performance multitasking operation, mass storage, graphics and text output, and instrument I/O in a compact, transportable package.

by Tim J. Williams and Nelson A. Mills

TO BE AN EFFECTIVE TOOL for technical applications, a personal computer system must provide more functions and higher performance than are normally required in an office environment. A versatile operating system, faster computational speed, ability to run more than one task concurrently, an instrument I/O interface, and transportability are some of the features often desired by technical professionals.

These needs led to the creation of the HP Integral Personal Computer (Fig. 1), a transportable computer based on HP's version (HP-UX) of an industry standard multitasking operating system—the UNIX™ operating system. By also creating a version of BASIC that runs in this UNIX environment on the Integral PC, HP provides a growth path for HP Series 80 Computer owners wanting more capability and performance without having to sacrifice their existing software routines and data files.

UNIX is a U.S. trademark of AT&T Bell Laboratories.

Design Objectives

The UNIX operating system in its normal implementations is an excellent system for software developers, providing a range of development tools including editors, compilers, debuggers, profilers, etc. However, it is also a very large and complicated system that is often not suitable for novice users. It is a relatively expensive system that requires substantial memory, usually supported by a hard disc drive. It is often difficult to install and configure and nearly impossible for a novice user to understand and use. Thus, additional objectives for the Integral PC were to reduce the entry-level price of the UNIX operating system by developing a flexible-disc-based version and to make the system as easy to configure and use as a standard personal computer.

To provide proper continuity from the Series 80 product line, there were three objectives:

- Compatibility with the existing software base of Series 80 BASIC programs so that they could be brought forward for use on the Integral PC.

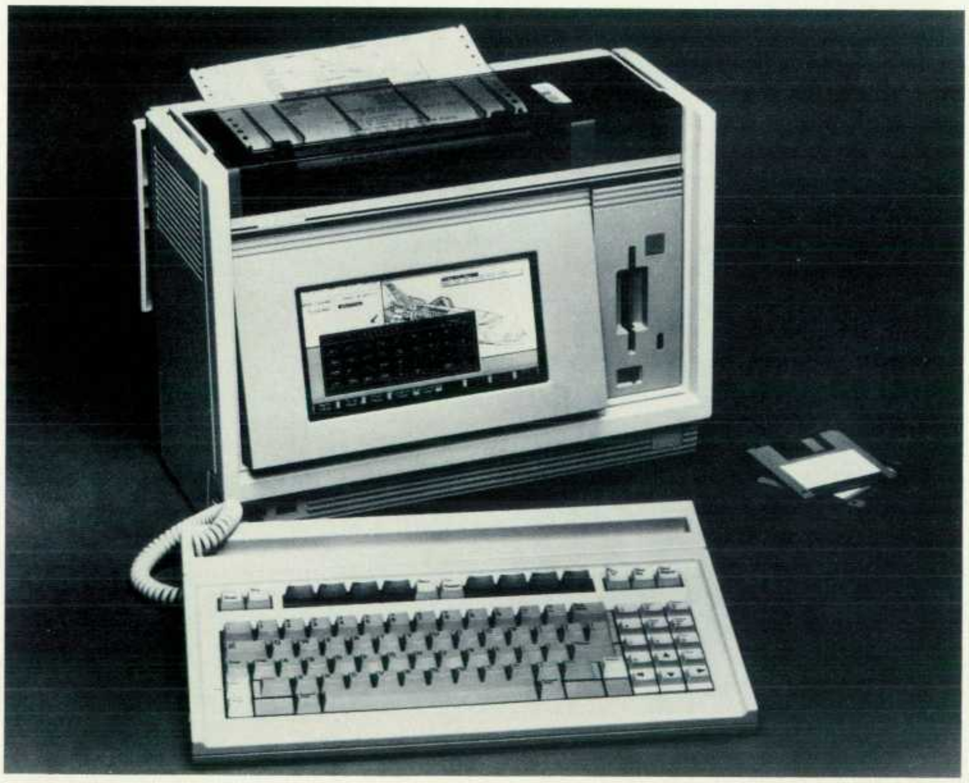


Fig. 1. *HP's Integral Personal Computer is a powerful multitasking computer system in a 25-lb transportable package. Designed for technical professionals, it features a built-in ThinkJet printer, a double-sided, double-density 3½-inch disc drive, a 255×512-pixel, bit-mapped electroluminescent display, and an HP-IB interface.*

- Add value by developing BASIC on top of the multitasking UNIX operating system. This would allow an Integral PC user to have the same instrumentation capabilities as the Series 80 and use the Integral PC for other tasks in a concurrent fashion.
- Develop BASIC in a higher-level language so that it can be easily ported to other environments.

These objectives had to be accomplished while achieving an overall performance three to five times better than that of Series 80 BASIC.

To satisfy the needs of the technical market, a transportable computer should have all the functionality of a desktop computer. Many customers enjoy the ability to move their machines to their required work locations; however, they are not willing to accept fewer features than a comparable desktop computer. In addition to easy movability, the computer system must be highly integrated and require minimal desk space. Hence, the design team decided that a high-quality flat-panel display would be required. After a lengthy investigation, they chose an electroluminescent display (see article on page 12).

Any complete computer system must have a printer, and the ThinkJet Printer mechanism[1] proved to be the best choice. This printer allows quiet printing, is small, and uses little power.

The 3½-inch flexible-disc drive Hewlett-Packard has chosen as a standard is small and rugged and has proved to be an excellent choice for mass storage.

The package should be fully self-contained and possible to set up in less than five minutes. This resulted in design decisions such as having the carrying lid permanently attached to the machine. The mechanical design of the machine (see article on page 18) allows for easy service and quick disassembly.

User Interface

Another primary goal of the Integral PC design was to provide a user interface that the novice user could understand (see article on page 28). The first element is a simplified installation and configuration process. This was accomplished by putting the UNIX kernel in ROM. No installation is required and the system is configured as part of the power-on process. Thus, to the novice user the system appears much as a standard personal computer would.

The second element is the implementation of a window manager to allow easy control of the multitasking environment that the operating system provides. Two basic window types are available in the system, although it is possible to create others. The alpha windows appear as standard terminal displays so that no modifications are required to get standard UNIX applications to run on the Integral PC. The other window type is a graphics window that emulates an HP-GL-based plotter.

Finally, the Integral PC has a user-friendly shell program that provides a visual window into the file system. This shell program is very similar to the Personal Applications Manager (PAM) available on the HP 150 Touchscreen Computer.[2] It includes a file area, a set of softkeys, and a command area for those who know and love the standard cryptic UNIX operating system commands.

Operating System

The Integral Computer runs HP's HP-UX operating system,[3] which is based on the standard UNIX operating system with a few modifications done to reduce the entry-level cost of the system and make it more attractive to personal computer users (see article on page 22). The kernel is in ROM and the system is based on flexible discs, which means that it can be run without requiring a large and expensive hard disc drive. This substantially reduces the cost of the system.

The file system is synchronous and nonswapping so that data destined for the disc drive is not buffered in memory, but is written immediately to the disc. This means that it is possible to remove a disc from the drive any time the drive light is not on. The directories and files on a disc are automatically mounted (listed as accessible by PAM) when the disc is inserted into the drive, and dismounted when the disc is removed.

The user-friendly PAM shell is a visual image of the portion of the file system that is currently mounted. Files are selected via a pointing mechanism using the cursor keys on the keyboard or the optional mouse. The system also includes a RAM disc that is allocated at power-on and whose size is dynamically increased or decreased as required during the execution of the system. This RAM disc contains the root file system and any temporary files that a user may choose to put there. The system includes a window management system that allows the novice user to manipulate the multitasking operating system easily.

The implementation of the UNIX operating system for the Integral PC has been enhanced to include some real-time support that allows the product to be used more effectively as an instrument controller. This HP enhancement (see "Real-Time Extensions" on page 25) includes a priority scheduling algorithm with 255 priority levels, and the ability to lock processes in memory.

BASIC

The BASIC language implementation for the Integral PC is generally source code compatible with the BASIC implementation that exists on the earlier Series 80 products, but there are exceptions that fall into one of the following categories:

- Minor math differences because the Series 80 processor performs BCD mathematics, whereas the Integral PC uses C-language mathematics libraries and does binary mathematics.
- Some minor syntax differences because the Integral PC BASIC implementation uses standard parse generators.
- A few small differences because of the multitasking operating system. Thus, BASIC programs must share resources with other programs that may be running concurrently.
- A few differences because errors in Series 80 BASIC have been corrected in Integral PC BASIC.
- Some differences in the character sets supported on the two products.

BASIC in the Integral PC includes not only the core functions that were built into the Series 80 Computers, but most of the plug-in ROM functions as well. Included are the functions of the Series 80 Matrix, Plotter, I/O, Mass Storage, and Advanced Programming ROMs.

Integral PC BASIC does not support extension of the language through the implementation of binary programs as Series 80 BASIC does. However, the language does support extended functionality through C subprogram calls.

One of the primary goals was easy transfer of existing Series 80 programs. This goal is supported by the upload program which converts SAVEd Series 80 files to Integral PC files. Since SAVEd Series 80 files are ASCII files, the GET command will cause them to be parsed after they have been uploaded to the Integral PC and thus to be converted into the internal format of Integral PC BASIC. Any statements that do not parse correctly will be turned into comments and flagged with the appropriate error message. They may then be edited so that the statements will be acceptable to Integral PC BASIC.

I/O

The Integral PC is designed to be flexible and expandable. An HP-IB (IEEE 488) interface is built-in to allow easy addition of plotters and other mass storage devices. The front of the machine has two HP-HIL ports (Hewlett-Packard Human Interface Link) for the connection of the keyboard and other input devices such as a mouse, optical wand, or graphics tablet. The operating system, which is in built-in ROM, has its own port. This allows users to update their machines with new releases of the operating system without requiring a service call. In addition, the Integral PC has two general-purpose I/O ports for adding more memory, controller, or data communication interface cards.

The space and power available for I/O functions in the Integral PC placed special constraints on the design of the interface cards. The cards had to have all the capabilities of earlier plug-in cards for desktop computers, but had to be more efficiently designed. In general, all cards are designed to comply with a maximum power specification of three watts and are software configured so that no switches are required. This lowers the cost and improves ease of use. The I/O cards for the Integral PC use pin-and-socket connectors instead of printed circuit board connectors to improve reliability. They also have detachable cables, thus enhancing the portability of the Integral PC.

The memory cards include 256K, 512K, and 1M-byte RAM cards. To satisfy data communication needs, an RS-232-C interface and a direct-connect 300/1200-baud modem are available. A general-purpose interface (GPIO), a binary-coded decimal (BCD) interface, a current-loop interface, and an HP-IL (Hewlett-Packard Interface Loop)[4] interface are available to satisfy controller needs.

References

1. C.V. Katen and T.R. Braun, "An Inexpensive, Portable Ink-Jet Printer Family," *Hewlett-Packard Journal*, Vol. 36, no. 5, May 1985.
2. P.S. Showman, et al, "Applications Software for the Touch-screen Personal Computer," *Hewlett-Packard Journal*, Vol. 35, no. 8, August 1984.
3. M.V. Hetrick, "HP-UX: A Corporate Strategy," *Hewlett-Packard Journal*, Vol. 35, no. 3, March 1984, pp. 12-13.
4. 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.

Electronics System for a Transportable Professional Computer

by David L. Kepler and James A. Espeland

THERE WERE MANY CHALLENGES in the electronic design of HP's Integral Personal Computer. As the software development advanced and inputs from marketing were clarified, a number of adjustments in the definition of the electronics were necessary. However, temptations to provide additional functionality and features in some areas had to be overcome to avoid increasing the system cost and running into problems with the limited space available.

A block diagram of the system electronics is shown in Fig. 1. The CPU, RAM, ROM, memory management, I/O buffering, system timing, and keyboard interface reside on one logic board, and all of the other peripheral circuitry (and 14 connectors) reside on another board. Each board is slightly smaller than a sheet of stationery (78 square inches). The interface between the boards contains data, address, and control signals. Each board has its own clock circuitry for reliability and ease of testing. An I/O board with two connectors for optional plug-in cards, a keyboard interface board with two connectors for Hewlett-Packard Human Interface Link (HP-HIL) input devices, and the power supply board are the other printed circuit boards.

System Considerations

The CPU in the Integral PC is an 8-MHz 68000 microprocessor. The 68000 has a 16-bit external data bus, but it can operate with 32-bit words internally. The 68000 provides a 16M-byte direct addressing range. The Integral PC uses this address space for a mix of RAM, ROM, and I/O as shown in Fig. 2. Here, internal refers to those devices that are built into the system, and external refers to those devices that can be plugged into an expansion slot. For example, external RAM refers to any RAM cards that may be plugged into the two expansion slots in the rear panel.

The design goals for the system logic included low cost and simplicity while sacrificing little or no functionality. The use of autovectoring and the DTACK generator reflect these goals. Autovectoring is the process in which the 68000 uses an internally generated interrupt vector which is a function of the interrupt being serviced. This relieves an interrupting device of the burden of generating interrupt vectors. The Integral PC uses autovectored interrupts exclusively.

DTACK (data transfer acknowledge) is an input to the 68000 that indicates when the data transfer is complete. The Integral PC uses a DTACK generator circuit that provides DTACK signals during internal ROM and internal I/O cycles. This circuit eliminates the individual DTACK circuits that otherwise would be required of each internal I/O device.

The initial design contained 128K bytes of RAM using sixteen 64K×1 dynamic RAMs and a TMS4500A dynamic RAM controller IC. A small amount of logic is used to generate DTACK to the 68000 and control signals to the TMS4500A. The dynamic RAM controller chip multiplexes row and column addresses and generates the RAS (row address strobe) and CAS (column address strobe) timing signals to the dynamic RAMs. This IC also refreshes the RAMs.

As the software development advanced, it became appar-

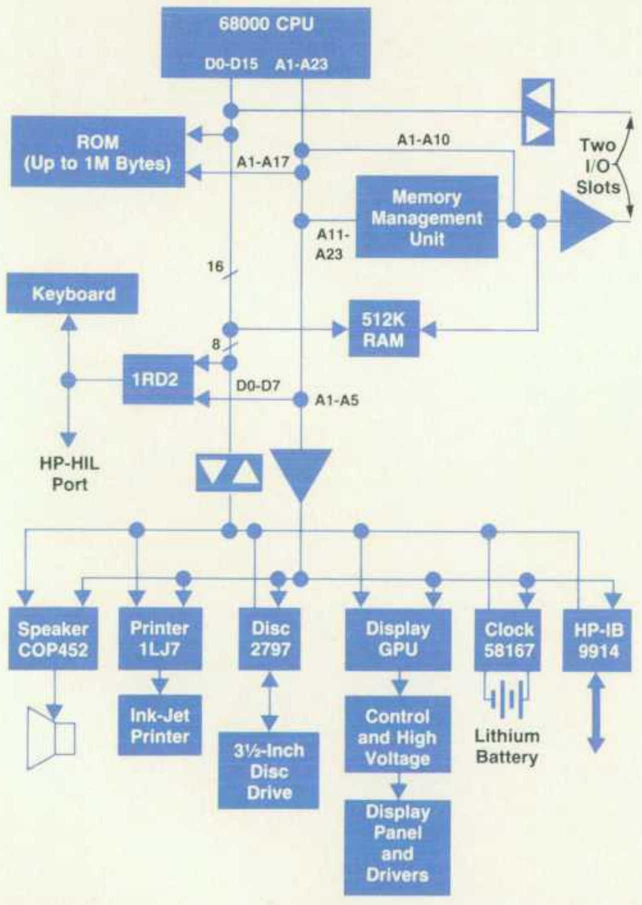


Fig. 1. *Block diagram of electronics system for the Integral PC.*

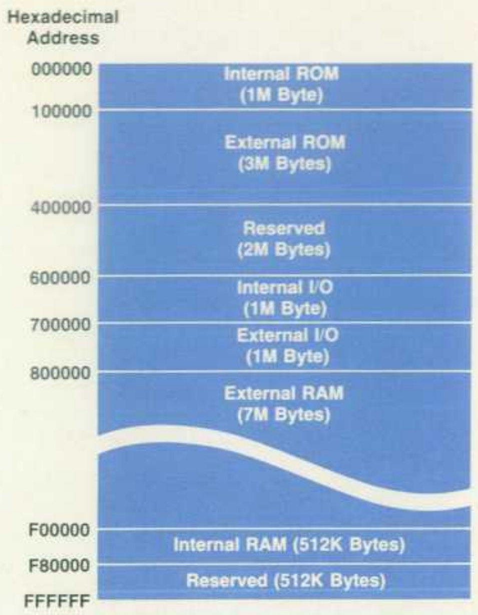


Fig. 2. *Allocation of the 16M-byte address range of the 68000 microprocessor used in the Integral PC.*

ent that 512K bytes of dynamic RAM would be required in the base configuration. Since the TMS4500A cannot address 256K×1 dynamic RAMS, a two-line-to-one-line multiplexer IC (74S157) is used to multiplex the additional two address lines to the most significant address bit (A8) of the 256K×1 RAMs. The TMS4500A is capable of refreshing these RAMs, because the RAS-only refresh cycle for 256K×1 dynamic RAMs requires only that address bits A0 to A7 be valid.

The RAM cycle time is 625 ns. If a refresh collision occurs (i.e., an access is attempted during a refresh cycle), the DTACK signal to the 68000 is held off until the access occurs. This is called an access grant cycle. The access grant cycle time is typically 875 ns.

The Integral PC contains up to 1M byte of internal ROM in a removable module, which is accessible through a door in the back of the product. The signals present at this port allow eight 128K×8 ROMs to be addressed directly.

The system DTACK generator yields a ROM cycle time of 625 ns. This allows the use of ROMs or EPROMs that have access times of 325 ns or less. A 500-ns ROM cycle can be forced by the ROM module when ROMs that have access times of 200 ns or less are used. A signal on the ROM interface bus is asserted when a 500-ns cycle is desired.

Memory Management

The memory management unit (Fig. 3) in the Integral PC translates an address to form the physical address by adding an offset to the logical address from the CPU. Address translation is performed only during RAM accesses, that is, when the most-significant physical address bit PA23 is high. In an effort to conserve costs, hardware bounds checking is not implemented.

The operation of the memory management unit is illustrated by the following example. The internal RAM is physically located at F00000$_{16}$ to F7FFFF$_{16}$ (refer to Fig. 2). If

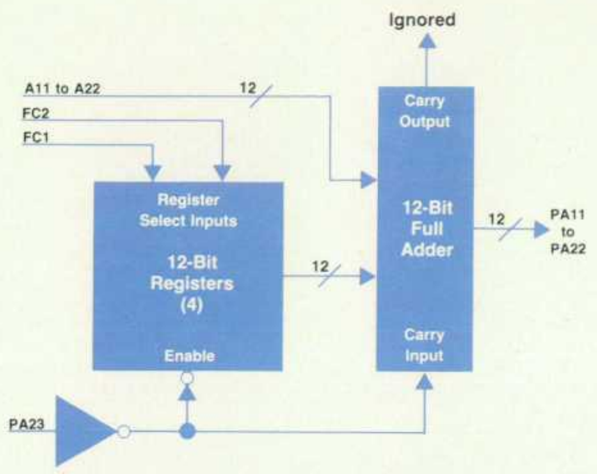


Fig. 3. *Block diagram of memory management unit.*

an offset equivalent to 100000_{16} is placed into the proper memory management unit register, then a logical address of $E00000_{16}$ is translated to $E00000_{16} + 100000_{16} = F00000_{16}$. Hence, logical address $E00000_{16}$ appears to the CPU to be an internal RAM address, although that internal RAM location is physically located at $F00000_{16}$.

The memory management unit consists of seven MSI (medium scale integration) ICs arranged as a 4×12-bit register file (three 74LS170 chips), a 12-bit full adder (three 74LS283 chips), and a PA23 generator (a 74ALS153 chip). During a RAM access, the contents of one of the four 12-bit registers in the register file is added to logical address bits A11 through A22 of the CPU to form the physical address. Since logical address bits A1 through A10 are not translated, the minimum segment size is 1K words (2K bytes). PA23 is not modified by the adder; this means that address translation is performed only within the RAM address space.

During a ROM or I/O access, the open-collector outputs of the 74LS170 chips are turned off and pulled high by pull-up resistors and a carry is input to the first 74LS283 chip. That is, FFF_{16} is added to logical address bits A11 through A22 along with a carry in. If logical address bits A11 through A22 are XXX_{16}, then $XXX_{16} + FFF_{16} + 1 = XXX_{16}$ + a carry. Since the carry output is ignored, this process essentially adds nothing to logical address bits A11 through A22.

The memory management unit register used for the translation function is selected by the FC1 and FC2 (function code) bits from the CPU. These two bits define four processor cycle types: user data, user program, supervisor data, and supervisor program. Each of the four registers in the register file corresponds to one of the processor cycle types. For example, the offset in the user data offset register is used during user data cycles. The contents of each register in the register file are written to the register by the operating system.

PA23 distinguishes translated RAM accesses from untranslated ROM and I/O accesses. PA23 is high during RAM accesses and low during ROM and I/O accesses. In the initial design, PA23 equaled A23 (logical address bit 23 from the CPU). The RAM was always located in the upper half of the logical address space, that is, in the range 800000_{16} to $FFFFFF_{16}$. However, the software designers expressed a desire to be able to translate user RAM addresses from logical address 000000_{16} (UNIX™ applications usually begin at logical address 000000_{16}).

One possible solution to this problem would have been to generate PA23 by inverting A23. If PA23 = NOT A23, then the RAM would be located in the lower half of the logical address space, that is, in the range 000000_{16} to $7FFFFF_{16}$. User RAM addresses could then be translated from logical address 000000_{16}. However, the CPU exception vectors reside at logical addresses 000000_{16} to $0003FF_{16}$. These vectors contain the addresses of interrupt handling routines and error handling routines. If PA23 equaled NOT A23 during exception processing, the 68000 processor would look for the exception vectors in RAM. Since some of these vectors are required at power-up (before the RAM is initialized), this solution was not acceptable.

However, since translation of RAM addresses from logical address 000000_{16} is only required while in user mode, the PA23 generator is designed to invert A23 only during user mode accesses. Hence, in user mode PA23 = NOT A23, and in supervisor mode PA23 = A23. The 68000 always enters supervisor mode before handling exceptions; therefore, it always finds the exception vectors in ROM at logical addresses 000000_{16} to $0003FF_{16}$. The logical address maps for supervisor mode and user mode are shown in Fig. 4.

Peripherals, Controllers, and Power Supply

The HP-HIL is used in the Integral PC as the keyboard interface. It allows other input devices such as graphics tablets and bar code readers to be used without requiring special software or the use of an I/O slot. Two HP-HIL receptacles are accessible on the front of the computer, generally one for the keyboard and one for the mouse (each on its own configurable link). The idea of the link is that multiple input devices can be connected in daisy-chain fashion, but neither HP's mouse nor the Integral PC's keyboard has the receptacle necessary for additional devices. This requires any additional input devices to have that receptacle and the keyboard and mouse to be located at the end of each chain or link.

A custom IC is used for the communication controller that interfaces the CPU to the HP-HIL input devices. This IC resides in the mainframe and controls communication to both interface links. The keyboard contains another custom IC that controls communication with the first chip, and a coprocessor that takes care of the scanning of the

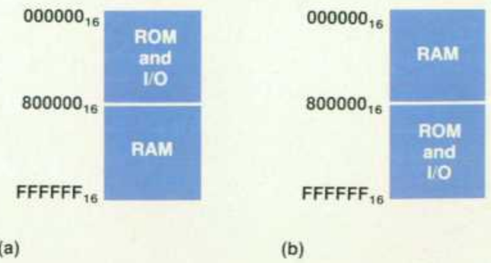


Fig. 4. *Logical address maps for (a) supervisor mode and (b) user mode.*

keyboard.

The main component of the display interface circuitry is the graphics processing unit, which is described in detail in the article on page 10. The dedicated 32K-byte display memory consists of four 16K×4-bit RAM ICs and some timing circuitry. Video shift registers and some more timing electronics needed for the electroluminescent display account for the remaining display interface circuitry. The timing signals required for the display interface also provide the clocks for the other peripheral control circuitry.

A 2797 flexible-disc controller is used to interface the internal disc drive with the system. This disc controller has both the phase-locked loop data separator and the write precompensation logic built into it, eliminating much of the extra circuitry necessary for earlier designs. Potentiometers are used for adjustment of write precompensation time and read clock pulse width. The other adjustment, and the most critical, is the voltage-controlled oscillator adjustment for the disc controller's data separator. Extensive experimentation was necessary to determine the optimum setting for this frequency since it drifts with temperature and has direct impact on the soft error rate of the disc interface.

In addition to the controller, one more system address is decoded to provide control signals for drive select, drive reset, motor control, and side select, and to get feedback from the drive related to disc changes and write protection. The only other electronics necessary for the disc interface is a set of buffer ICs that drive and receive signals to and from the disc drive. Not included in the disc interface for cost and space reasons are sector buffering and DMA facilities for sector transfer. The need for these features was minimized because the operating system does not require the disc/RAM swapping operation that UNIX operating systems normally perform.

The printer electronics are leveraged almost entirely from the HP-IL version of HP's Thinkjet Printer. The HP-IL interface to the printer controller required the use of the custom HP-IL IC[1] for the system interface. The printer electronics include the 48-pin printer controller, a 1K-byte character buffer, a character lookup ROM (all custom ICs from HP's Northwest IC Division), four Darlington driver ICs for motor and printhead drive, and a 22.8V ±1.2% printhead supply derived from the system's 12V supply.

The real-time clock provides timing for the operating system, as well as clock and calendar functions. An MM58167A CMOS real-time clock is used for its low power and high functionality. It has two outputs available to the system in the form of interrupts. One provides the system's heartbeat, the lowest-level interrupt used to initiate system updates. The other is available to provide an interrupt after a specified period of time. A 1.2 ampere-hour lithium battery, which is mounted on a printed circuit board, provides a minimum of six years of life to the clock, and typically will last more than ten years. The crystal circuitry external to the clock chip includes a trimmer capacitor that allows the frequency to be adjusted to obtain a typical accuracy of six minutes per year.

The speaker circuitry provides the user/programmer with a wide range of tones for audible feedback without adding much cost or complexity to the system. A COP452 frequency generator/counter IC was chosen for its small size and low cost. The main drawback is its serial interface, which requires a 74LS175 latch chip and software to simulate that interface off the system bus. The frequency generator drives an operational amplifier that allows the piezoelectric speaker to be driven by a 12V signal, the level necessary to generate the desired volume. By using a speaker that could be mounted on a printed circuit board, the designers eliminated one connector in a system where the number and organization of connectors and cables were continuous concerns.

The built-in HP-IB interface serves two major purposes. First, it is the primary means of interfacing the Integral PC to external peripherals such as disc drives and plotters. Second, when the Integral PC is used as a controller, the HP-IB interface will usually be the primary means of interfacing to the various instruments involved.

Only three ICs are required for the HP-IB interface—one NMOS LSI IEEE-488 talker/listener/controller chip and two TTL IEEE-488 bus transceivers. Since the talker/listener/controller chip has limited intelligence, the 68000 system processor must interact with it closely to perform various HP-IB functions. This circuitry was easily and inexpensively integrated into the system design.

The primary power supply is an off-line, fixed-frequency, pulse width modulated, switch-regulated supply. The supply converts either 230Vac or 115Vac input power (defined by a switch on the rear panel) into five dc output levels: 5V, 12V, 15V, 18V, and −12V. Power status logic is provided to warn the real-time clock when the 5V supply is about to decay on power-down as well as indicating to the system when all voltages are correct after power is switched on. Power supply features include current limiting and automatic shutdown and restart, as well as input and output overvoltage protection.

Acknowledgments

We wish to acknowledge the other members of the electronics design team for the Integral PC—project manager Doug Collins, Eugene Cooper, Bill Eaton, Greg Goebel, Tim Harper, Randy Hergett, Marv Higgins, Ken Lo, Ted Ransom, and Linh Tu. Thanks to Teri Wilson, our project coordinator, for helping the project run smoothly. Thanks also to Leonard Rosi, John Milton, and the rest of the quality assurance department for their support.

Reference

1. S.L. Harper, "A CMOS Integrated Circuit for the HP-IL Interface," *Hewlett-Packard Journal*, Vol. 34, no. 1, January 1983.

Custom Graphics Processor Unit for the Integral PC

by Dean M. Heath

THE GRAPHICS PROCESSOR UNIT (GPU) used to control the orange electroluminescent flat-panel display of HP's Integral Personal Computer is a custom 48-pin CMOS integrated circuit fabricated at HP's Northwest Integrated Circuit Division in Corvallis, Oregon. It is a special-purpose microprocessor designed to be used as a powerful dual CRT and electroluminescent display controller to provide the Integral PC a versatile, easy-to-use graphics subsystem with a bit-mapped display. The GPU is compatible with most commercially available CRT monitors and can rapidly manipulate lines, rectangles, windows, alpha characters, an automatic alpha cursor, and a sprite (pointing device). It also features a user-configurable screen size, monitor sync timing, multiple character fonts and fill patterns, and a display RAM interface.

Architecture

The GPU consists of the following functional blocks (Fig. 1):

- Bus interface logic
- Microcode control logic
- Arithmetic logic unit (ALU)
- 32×16-bit RAM
- Display memory interface logic
- Display timing logic.

The bus interface logic includes a standard asynchronous 8-bit microprocessor interface with one byte of status information, two bytes of output data, four bytes of input data, and one byte of command data. The microcode control logic, ALU, and RAM combine to form a compact 16-bit microprocessor with parallel next-address calculation, a two-level return address stack, a pipelined processor, a 16-bit ALU with special-purpose logic unit design, a barrel shifter, and test logic.

The microprocessor with its parallel next-address calculation, pipelined processor, and ALU allows the GPU to execute its microcode more rapidly than possible in standard architectures where the address calculations are performed by the ALU. This performance is achieved, in part, because the GPU can prefetch the next instruction while processing the current one. Some care must be exercised when writing microcode for a machine with a processor of this type. For example, to perform the function C:= A + B + C, the microcode might be written as shown on the left below:

Incorrect Code		Correct Code	
Step 1:	A := A + B	Step 1:	A := A + B
Step 2:	A := A + C	Step 1.5:	NOP
		Step 2:	A := A + C

Because of the way the pipeline works, the result of step 1 would not be placed into the A register until after step 2 executes. Hence, the result would not be A + B + C, but A + C. Step 1.5 in the correct sequence on the right above must be added to allow the pipelined calculation to complete correctly before continuing. In most cases a NOP step is not used, but some other required operation is performed effectively out of sequence to optimize use of the ROM and microcode cycles.

The special-purpose logic unit allows the ALU to perform concurrently any of the 16 possible logic operations controlled by the mask register. This capability makes many of the GPU commands faster, since many of its window manipulating commands require portions of memory words to be combined logically with portions of other memory words. To exclusive-OR bits 0 through 3 of registers R0 and R1 and leave bits 4 through 15 unchanged, the follow-

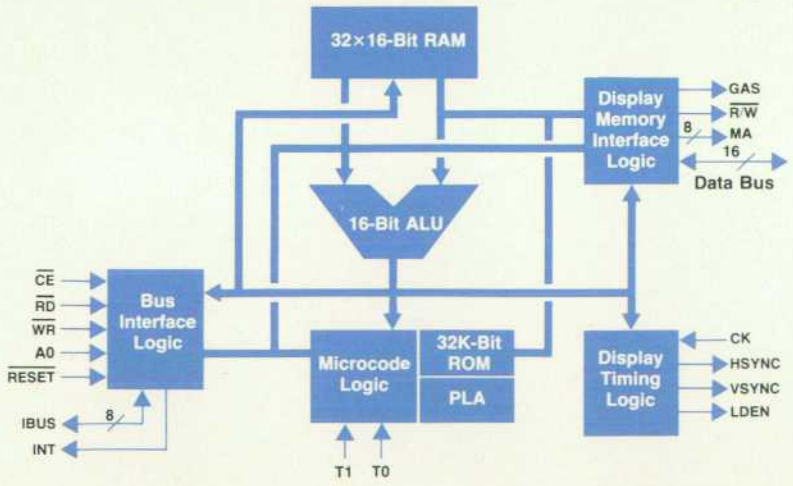


Fig. 1. *Block diagram of custom graphics processor unit for the Integral PC.*

ing microcode might be used:

```
ALU without Mask Register            GPU ALU with Mask Register

MASK  := 0000000000001111            MASK  := 0000000000001111
TEMP1 := R0 XOR R1                   R0    := R0 XOR R1
TEMP1 := TEMP1 AND MASK
TEMP2 := R0
TEMP2 := R0 AND NOT MASK
R0    := TEMP1 OR TEMP2
```

The dedicated two-level return address stack, though limited in size, allows the GPU to return from subroutine calls rapidly without needing to maintain a stack pointer. Because of the small size of the GPU ROM (1536×21 bits) and limited random access memory, massive use of subroutine nesting levels was not considered desirable and thus did not affect microcode development. The barrel shifter allows the GPU to shift two 16-bit registers to the right or left from 0 to 15 bits in one clock cycle. This capability is required since many of the GPU operations move massive amounts of data (organized as 16-bit words) with bit resolution, requiring adjacent memory words to "slide" into each other.

The display memory interface logic allows the GPU to control a display with up to 128K bytes of static or dynamic RAM. When using dynamic RAM, the GPU will automatically refresh the memory since it accesses the memory sequentially during the display read cycle. Because either static or dynamic RAM may be used with the GPU, RAS (row address strobe) and CAS (column address strobe) are not generated. Instead, a more universal signal, GAS, is output which can be used to control either type of memory. The display memory is organized as 16-bit words, which are read sequentially and sent via a 16-bit shift register to the video input to the display device. The GPU interleaves memory cycles to guarantee a flicker-free display.

The timing logic allows the GPU to interface to virtually any type of display that uses serial data, blanking, and sync pulses similar to those used by a CRT monitor. The Integral PC has an electroluminescent flat-panel display and the GPU has all the capability necessary to use this type of display. The maximum screen size is 1024×1024 pixels.

The maximum clock frequency the GPU can use is 5 MHz, which is one fourth the video dot clock frequency. For very large displays requiring dot frequencies up to 40 MHz, the GPU has a double-wide mode that allows the chip to operate at one eighth the video frequency. In this mode, display reads are 32 bits wide, requiring a 32-bit shift register, but display update accesses are still 16 bits wide.

The test logic incorporated into the GPU allows the following test modes: ROM dump mode, single-step mode (output program counter), and single-step mode (output ALU output latch). These test modes were used for initial debug and production testing of the GPU. Since this is a single-chip computer, with portions of the chip virtually inaccessible externally, test microcode is included to allow testing the chip in production. For example, the internal RAM is impossible to test by executing GPU commands, but the test microcode allows the entire internal RAM (512 bits) to be tested.

To simplify the design of the GPU and to minimize chip count, some initial constraints were imposed on its architecture. To simplify the RAM interface logic, it was assumed that no refresh counter would be needed to refresh dynamic RAMs since they would be refreshed automatically by the display read cycles. However, this decision prevents the GPU from performing real-time zoom and interlacing. All ROM for the GPU micromachine had to be on-chip and occupy no more than 25% of the chip area; thus, the internal ROM size was set at 1536 words of 21 bits each.

User Interface

All the functions performed by the GPU are initiated using relative X-Y coordinate commands, with the upper left corner of the displayed screen at 0, 0. The GPU performs all conversions of X-Y coordinates to the memory addresses required by its commands. They can be classified as control, line drawing, window management, character placement, sprite, and cursor commands.

Control Commands. The control commands control the display operation, memory organization, and X-Y coordinate mapping. The CONF command is used to program the size of the display, monitor timing, and memory configuration. There are three segments in display memory: the displayed screen memory, the character font memory, and the data area. The WRSAD, WRFAD, and WRDAD commands are used to set up these memory segments. The WRMEM and RDMEM commands are used to load the font and data segments. The font data consists of 16×16-dot character bit patterns and a proportional spacing table. The data area consists of 16×16 area fill, sprite, and cursor bit patterns.

Line Drawing Commands. The line drawing commands are used to move and draw with the pen (a pointer to the current display location being addressed), using absolute coordinates or relative coordinates. The DRAWPX command is a faster command that can be used to draw single dots. The FRAME command is used to draw rectangles. All line drawing commands are performed using the user-defined line UDL, a 16-bit-wide dotted line pattern. Lines drawn on the screen are combined with data already in display memory using the replacement rule which selects one of the 16 possible logic functions.

Window Commands. The window commands are the most powerful functions the GPU can perform; these commands are used to copy, fill, or scroll windowed portions of display memory. Scrolling can be performed in any of four directions with user-selected step size. The vacated area is then filled with a user-selectable area fill pattern. The GPU checks for overlapping windows in all window operations and adjusts its window move algorithm to prevent losing data in the overlap area.

Alpha Label Commands. The label commands allow user-definable characters to be placed at any pixel position on the screen. Characters are placed relative to the window defined last, with any portion of the character outside of the window automatically clipped. After the character is placed, the pen automatically moves to the origin of the next character according to the width stored in the proportional spacing table.

Sprite and Cursor Commands. A sprite is a 16×16 pixel pattern which can be displayed anywhere on the screen, exclusive-ORed with other display data "underneath." When the display is updated, the following occurs:

1. The sprite pattern is removed from display memory.
2. The display is updated.
3. The sprite pattern is redrawn into display memory.

The cursor operates in a manner similar to the sprite, but is tied to the pen. It is intended for use as an alpha cursor and displays a user-defined cursor character relative to the current pen location. When the display is updated, the cursor operates the same as the sprite. The only difference is that the cursor can be offset relative to the pen so that the cursor can be displayed to the right of the last displayed character as in a standard alpha display.

Speed Performance

Assuming that the GPU is operating at its maximum clock frequency of 5 MHz, the following is a sample of the speed of the chip for a few selected operations:

Draw pixel:	36 μs
Draw line:	1634 μs (200-pixel line) = 8.17 μs/pixel
Draw frame:*	4869 μs = 6.09 μs/pixel
Place character:	179 μs (character height = 15, cursor and sprite are disabled)
Copy window:*	17,516 μs = 0.438 μs/pixel
Fill window:*	11,012 μs = 0.275 μs/pixel
Scroll window down:*	16,509 μs = 0.413 μs/pixel
Scroll window right:*	18,707 μs = 0.468 μs/pixel

*200×200-pixel window size.

High-Quality Electroluminescent Display for a Personal Workstation

by Marvin L. Higgins

THE PRINCIPLES behind today's electroluminescent display technology have been known for nearly 50 years. The phenomenon of electroluminescence was reported in 1936 by G. Destriau.[1] What prevented its commercialization until recently is the requirement for extremely thin and uniform pure layers, or films, of the required materials. The dielectric layers used in what is technically referred to as ac thin-film electroluminescent (ACTFEL) displays[2] must withstand very high electric fields. A foreign particle or thin spot in a dielectric film of an ACTFEL display can cause dielectric breakdown and may ruin the display. Thin-film technology had to progress to its current state of the art to make electroluminescent displays practical and reliable.

A commercially developed electroluminescent display is shown in Fig. 1. Such displays were developed by several manufacturers in cooperation with Hewlett-Packard during the development of the Integral Personal Computer. HP contributed to their design by providing a product definition, and by participating to varying degrees in product development, problem solving, and testing and evaluation.

HP's effort to ensure that a quality display would be ready in time for the introduction of the Integral PC included not only working with several different display manufacturers, but also involved HP's own development project for driving and packaging an electroluminescent

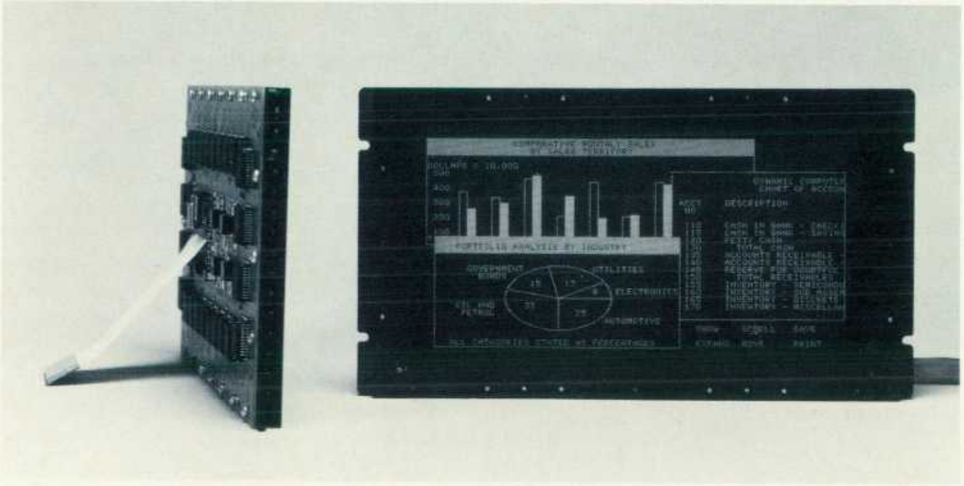


Fig. 1. *Commercially available 256×512-pixel electroluminescent display.*

panel. This in-house program was carried successfully through the prototype stage, but later shelved to focus on outside vendor designs. Although the in-house design was not produced in quantity, the development helped us evaluate and participate in the outside designs effectively.

Choosing a Display

Our first task was to look at existing display technologies in relationship to the design goals and constraints for the Integral PC. A flat-panel technology would be necessary because of the product form factor we had chosen. Because a display is so central to the personal computer human interface, we set very high standards for visual quality. We wanted the display to be big enough for easy readability of twenty-four 80-character lines, but still small enough for easy manufacturing and low cost. A thermal dissipation study of the Integral PC established the display power consumption goal at less than 15 watts. Finally, the display had to be manufacturable and cost effective by the scheduled introduction date for the Integral PC in early 1985.

The challenge was that at the beginning of the Integral PC project such a display did not exist! Liquid-crystal displays were not large enough at that time and did not meet our visual quality objectives. Plasma-discharge panels consumed too much power and were expensive. Existing electroluminescent displays at that time were half the size we needed and required too much power (Sharp Corporation in Japan was then producing 128×512-pixel and 320×240-pixel displays). Other flat-panel display technologies were even farther from being producible.

We chose an electroluminescent display technology because of its potential for excellent visual quality and moderate power consumption. With a concentrated effort, we felt our display objectives could be attained.

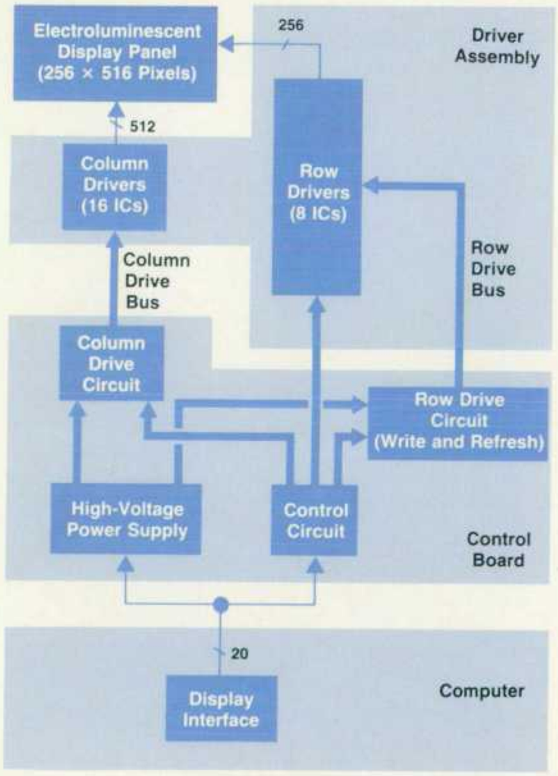


Fig. 2. *Block diagram of Integral PC display system.*

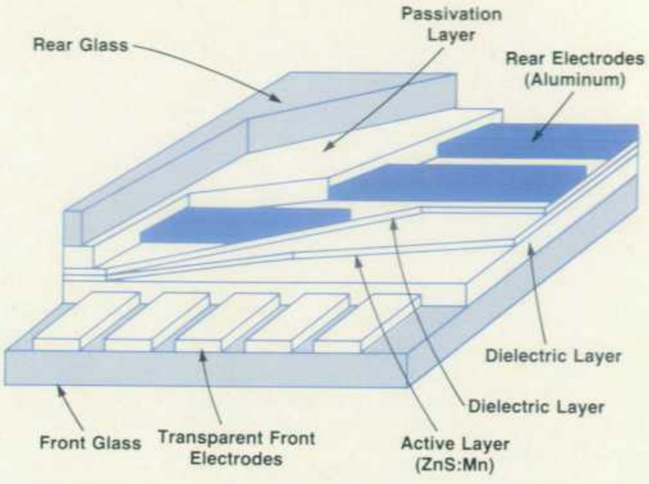


Fig. 3. *Structure of electroluminescent display panel.*

Electroluminescent Display

The Integral PC's electroluminescent display consists of three main parts or subassemblies: the electroluminescent panel, a driver assembly, and a control board (see Fig. 2).

The electroluminescent panel is the light-emitting device. The driver assembly contains high-voltage switches connected to each row and column of the panel. This assembly receives logic control signals and high-voltage pulses from the control board and applies voltage to the display panel according to the desired pattern.

General operation. An electroluminescent display can be thought of as a matrix of light-emitting dots. In the case of the Integral PC's display, that matrix is 512 dots wide by 256 dots high. (However, as a result of software and hardware trade-offs, the 256th line is not used.) The dots, called pixels, are located at the crossovers or intersections of the horizontal and vertical electrode patterns that are built into the display. When a horizontal electrode and a vertical electrode are properly activated, light is produced at their intersection. An image is formed by turning each pixel either on or off in a desired pattern.

The electroluminescent panel consists of a transparent, multilayered thin-film structure formed on a glass substrate (see Fig. 3). The active, or light-emitting, layer is zinc sulfide doped with manganese (ZnS:Mn). This active layer is sandwiched between two insulating layers and the resulting structure is then further sandwiched between two orthogonal sets of parallel thin electrodes. The total thickness of this thin-film structure is about one micrometer.

When a voltage is applied between two orthogonal (row and column) electrodes, a very high electric field is produced at their intersection. When this field is high enough (on the order of 2,000,000 V/cm) in the active ZnS:Mn layer, electrons shift quantum levels and light is emitted. The emitted light is a yellow-orange color (wavelength ≈585 nm), which corresponds to the energy difference between the excited state and the ground state of a manganese atom. Given the thickness of the active layer between the two orthogonal electrodes, the voltage required to light the electroluminescent panel is typically 180 to 230V. Below this threshold, no visible light is emitted (see Fig. 4).

Application of a waveform with a dc component to the active layer causes polarization in the active layer, which develops an internal electric field that opposes the applied field. This reduces light output because of the reduced effective field strength in the active layer. If sustained long enough, the internal dc field can damage the electroluminescent panel. Therefore, for proper continuous operation, the polarity of the applied voltage must be alternated, that is, an ac pulse train is required.

Writing the display. The method of addressing is shown in Fig. 5. The display is addressed one row at a time. Hence, this is called "line at a time scanning" as opposed to the "dot at a time scanning" used in CRT displays. Bits corresponding to the row pattern to be displayed are shifted into the shift registers of the column drivers, then latched. Columns desired on are switched to connect with the column drive bus. Columns desired off are switched to ground. Then, the column drive bus is pulsed to 60V and the row drive bus to −170V. The sum of these voltages (i.e., 230V) is thus applied to the desired pixels in the selected row, turning those pixels on. Dark pixels have only 170V applied, which is not enough for a visible output (Fig. 4).

This process is repeated until each of the rows has been written. The necessary ac cycle is now completed by reversing the polarity of the high voltage applied to each pixel in the entire display at once. This is referred to as a refresh cycle and is done by grounding the column electrodes and applying approximately 200V to all the row electrodes.

Driver Assembly

The driver ICs (Fig. 6) require a special device technology called DMOS (double-diffused MOS) to withstand the high voltages required (up to 250V). The logic portions of these ICs are done with typical NMOS or CMOS device technologies.

High-voltage pulses are received from the control board on the row bus and the column bus. Data is received on two lines: one for the top (TVID) column drives and one for the bottom (BVID). The electrodes on the panel are interleaved from opposite sides so that column one connects at the top, column two at the bottom and so on. Likewise, row one connects at the left, row two at the right. Hence, odd data bits (TVID) go to the top and even data bits (BVID) to the bottom. Interleaving doubles the space between electrodes at the edges of the panel, which makes interconnection easier. It also allows the driver ICs to be placed on all four sides, which uses space effectively and simplifies printed circuit board layout.

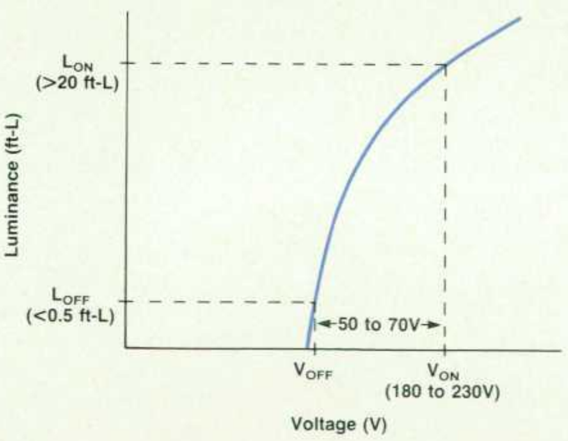


Fig. 4. *Typical luminance-versus-voltage curve for electroluminescent display.*

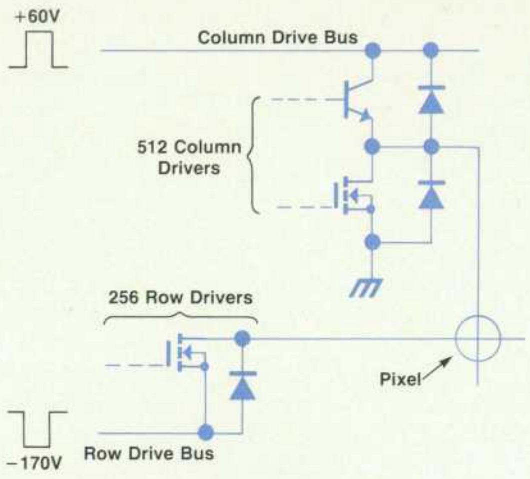


Fig. 5. *Drive scheme used to excite the pixels on the Integral PC's electroluminescent display.*

The shift registers in the column drivers are cascaded, forming a 256-bit register on top and another 256-bit register on the bottom. Odd and even data bits are shifted simultaneously into these registers and latched. The shift registers in the row drivers are also cascaded, and a high bit is shifted down the combined register to turn on the selected row. On command from the control board the outputs are enabled and the voltages on row and column buses are applied to the selected row and columns.

Energy Recovery Drive Scheme

Low power consumption is an important objective, even in line-operated products, mainly because of the problems associated with excessive heat. Our original design estimates, based on then existing drive schemes, were in the 30-to-40-watt range for a 256×512-pixel electroluminescent display. This was too high for our goal of less than 15 watts, so the low-power drive scheme described here was developed.

The basic idea relies on the fact that the electroluminescent panel looks primarily like a capacitive load (Fig. 7). Most of the power required to light the display goes into switching the energy stored in its capacitance, not into the production of light. By adding an external inductor, a series-LC resonant circuit is formed. This allows recovery of some of the energy required to charge the capacitance.

The capacitance C_C presented to the column drive bus varies, depending on how many columns have been selected on. This capacitance will increase when the pixels turn on, but this can be neglected for now. For an individual pixel, capacitance $C_e = 7$ pF, and the row capacitance $C_R = 512C_e = 3.58$ nF. The maximum column bus capacitance

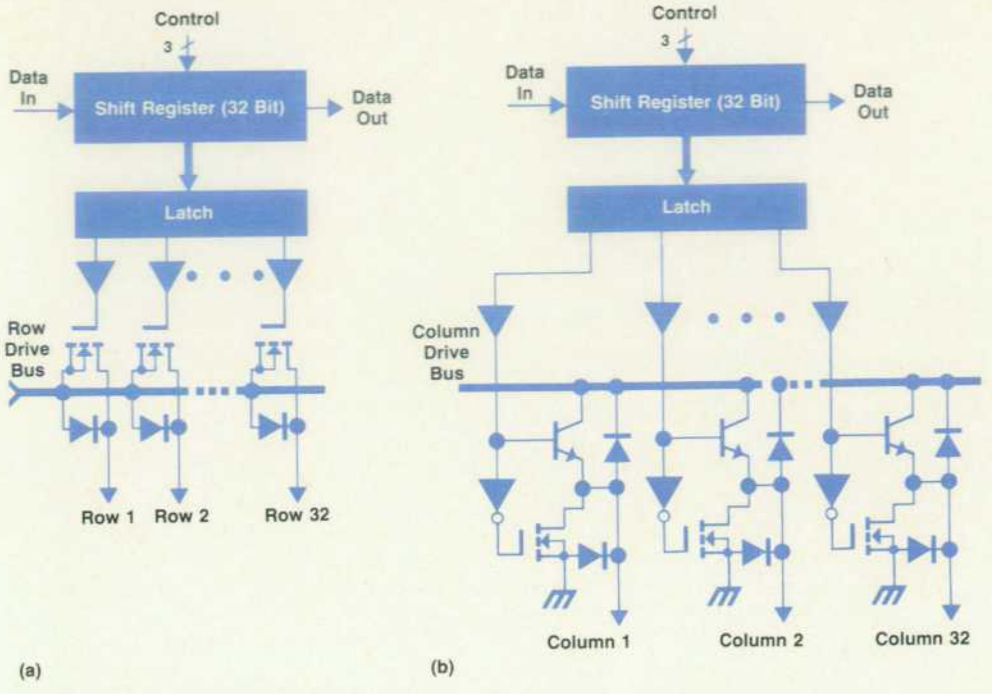


Fig. 6. *Schematic diagrams of driver ICs. (a) Row driver. (b) Column driver.*

occurs when half the columns have been selected on. That is, $C_C = 256C_e(255/2) = 228.4$ nF.

Column drive. Since the column bus capacitance is many times that of the row bus, most of the power will be dissipated in driving the columns (greater than 95%). For this reason we concentrated on the column drive circuit.

In a direct drive application the column bus capacitance is part of a series-RC circuit. Each time the capacitor C is charged to V volts, energy $= CV^2$ is supplied. Half of the energy is stored in the capacitor and half is dissipated in the resistor R. In the case of a series-RLC circuit, again an energy of CV^2 is supplied and half of the energy is stored in the capacitor C charged to V volts. The remaining energy goes to the inductor L and resistor R. The split between L and R depends on their impedance. If $R << X_L$ (inductive reactance), most of the energy is stored in the inductor. In this case, energy can be shifted back and forth between L and C, charging and discharging C with relatively little energy loss. Thus, most of the energy required to charge the capacitor is, in effect, recovered.

The column drive circuit, represented in Fig. 8, takes advantage of the energy storage/shifting characteristics of the LC resonant circuit without allowing oscillation. Note that C equals the column capacitance C_C plus a load capacitance C_L, which will be explained later. Resonant pulses are formed by controlling the current flow through L and C with switching transistors S1 to S4 (see Fig. 9). At t_1, transistor S1 turns on. The capacitor C starts charging toward $2V_M$ (2×60 volts). When it reaches 60 volts, the voltage is clamped by D2 and S1 turns off. S2 then turns on, holding the voltage to 60 volts and discharging the inductor L through D3 back to the power supply capacitor C_{ps}. At time t_3, S3 is turned on with all other switches off. C now begins to discharge through L. Upon reaching zero, the voltage across the capacitor is clamped by D4 and S3 turns off. S4 now turns on, holding the voltage at zero and discharging L through D1 back to the capacitor C_{ps}. In this way an essentially rectangular pulse can be formed while still retaining the energy storage/shifting characteristic of a series-LC circuit.

Timing of the resonant pulse is determined by the values of L and C. The rise time $(t_2 - t_1)$ and fall time $(t_4 - t_3)$ are 8 to 19 μs, corresponding to one-quarter wavelength at the resonant frequency ($\approx$ 13 to 31 kHz), using a 600-nH inductor.

Since, C_C varies widely depending on the number of columns selected, an additional load capacitor C_L is added to limit the maximum resonant frequency. This limits the minimum rise time and thus the maximum peak current, which in this case is about 1.2 amperes.

Row drive. Although the above energy recovery technique could be applied to the row driver circuitry as well, it was not. As explained earlier, most of the power is dissipated in the column drivers. Therefore, the added complexity

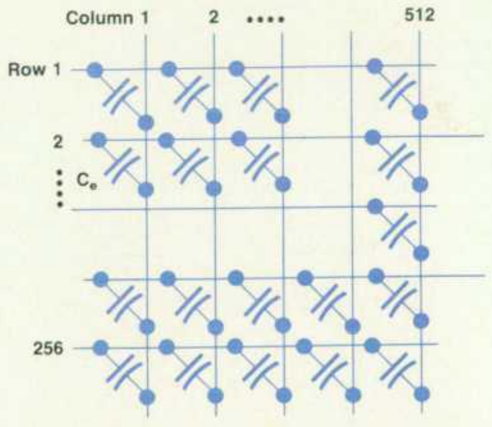


Fig. 7. *Each pixel of the electroluminescent display contributes a capacitance C_e that must be charged and discharged by the driver ICs.*

was felt to be not worth the gain. Instead, the row drive voltage is generated by means of switched current sources (see Fig. 10). The −170V write pulses are formed by turning transistor S1 on until row voltage $V_R = -170V$. Then, after a holding period, S2 is turned on until $V_R = 0$. Refresh pulses are similarly formed by using transistor switches S3 and S4.

Timing is primarily determined by frame rate, number of rows, and a minimum dwell time (the time a pulse is high or low) as determined by the display manufacturer. The current sources required for the desired timing of write and refresh pulses are 50 mA and 3A, respectively. The write current is equal to the nominal row capacitance C_R (3.6 nF) times the desired voltage gradient for the write waveform (dV/dt = 170V/12 μs). The refresh current required is equal to the total row capacitance ($256C_R$) times the desired voltage gradient for the refresh waveform (dV/dt = 200V/61 μs).

Interfacing the Display

The computer interface is located on the control board (refer to Fig. 2). Interconnection is through a 20-lead cable containing five control lines (clock, horizontal sync, vertical sync, and two video signals), power supply connections (5V and 20V), and grounds. All signals are compatible with the low-power Schottky TTL family. The waveforms are shown in Fig. 11.

The display is written at the rate of about 60 frames/second. Horizontal sync HS stays high while data bits are being shifted into the registers at about 6 MHz. When HS goes low, the data is latched. The high-voltage drivers are then enabled and the selected row displayed.

The video clock (VCLK) period allows time to shift in two bits (TVID and BVID) simultaneously. A dual data path is used to reduce the video and clock frequencies on the interconnect cable, thus reducing radiated electromagnetic interference. This also simplifies the computer interface. The horizontal period is equal to 256 VCLK periods plus horizontal low time (greater than or equal to 16 VCLK periods). Horizontal low allows time for latching the 512 data bits and resetting circuits for the next line of data.

The rising edge of vertical sync VS signals the start of a frame or new screen of data. The vertical period of about 16.7 ms allows for 256 lines plus refresh (≈ 300 μs).

Other Considerations

Shadowing. Shadowing is a visual effect that makes the display look streaked and nonuniform. In a given row, the brightness of the on pixels increases with the number of off pixels. For example, at the end of a line of text (dark characters on light background), the pixels in the rows used to display the text will be brighter than in the surrounding area where no text is written. This is undesirable since the human eye is very sensitive to abrupt changes in luminance. The typical threshold of perception is 1 to 3% luminance variation.

In HP's drive scheme and some others, a correct choice of timing for the modulation (60V) and write (−170V) pulses can minimize this effect. In other schemes, a compensation voltage or current that varies as a function of the number of off pixels must be added.

Protection circuit. As mentioned earlier, the display may be damaged by application of a dc voltage. A number of faults, both internal and external, can cause a dc voltage to be applied. For example, a failed component could stop the write pulses while the modulation and refresh pulses continue. This results in an internal dc potential.

Protection against common faults such as losing sync signals is inherent in the controller's design. Protection against a myriad of other faults is provided by a circuit that monitors the column drive bus and the row drive bus. The average voltage of each is compared with a reference. If either voltage is different from the reference, the high-voltage power supply is disabled.

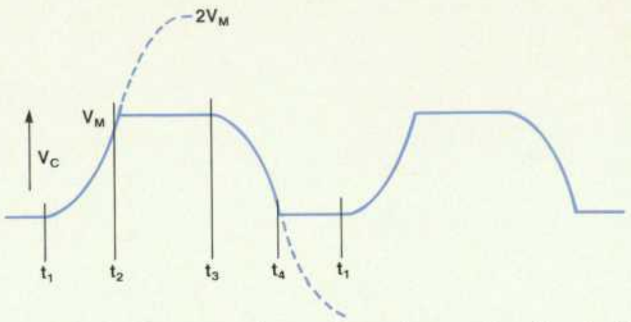


Fig. 9. *Resonant column drive pulse waveform for circuit shown in Fig. 8.*

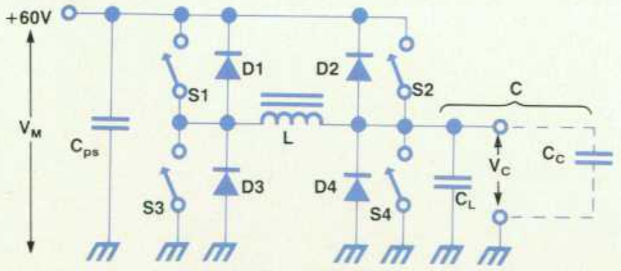


Fig. 8. *Equivalent circuit for the column drive used for the Integral PC display (see text for explanation of its operation).*

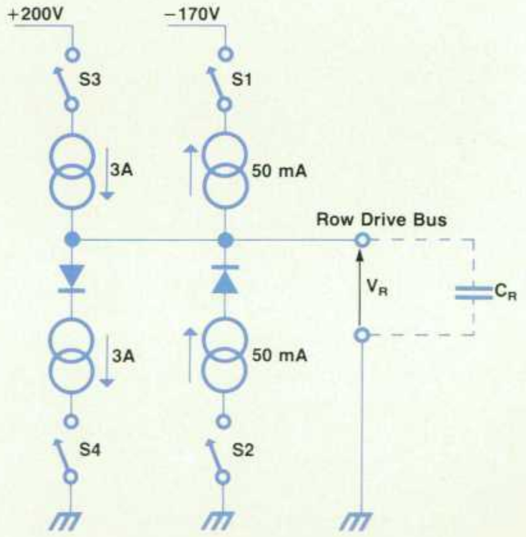


Fig. 10. *Equivalent circuit for the row drive used for the Integral PC display. The current source levels are determined by the desired drive and refresh waveforms.*

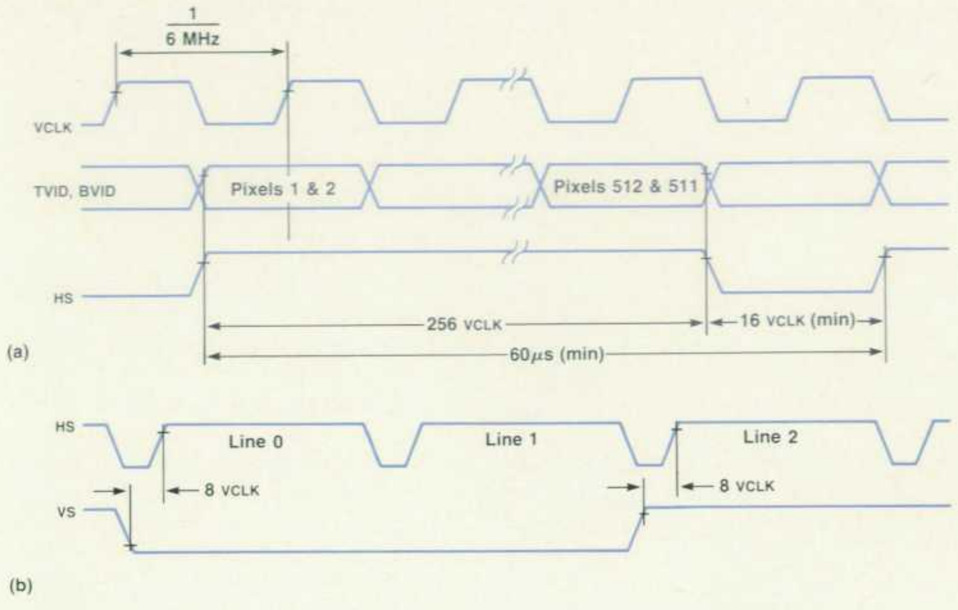


Fig. 11. *Timing diagrams for interface signals between the display system and the rest of the Integral PC electronics. (a) Horizontal sync versus video clock and pixel enables. (b) Horizontal sync versus vertical sync.*

Pattern retention. If a fixed pattern is displayed for a long time, that pattern may be permanently retained by the display. This phenomenon is caused by the on pixels gradually getting dimmer. Electroluminescent displays are more prone to pattern retention when new, since the rate of luminance change is greatest in the beginning. We dealt with this potential problem by:

- Specifying to the manufacturer the maximum allowable rate of change of luminance. (No visible pattern after 100 hours of fixed pattern.)
- By automatically turning the display off (disabling high voltage) after 15 minutes of inactivity. The display can be restored by pressing any key.

Contrast Enhancements

It was desired to have a very high-contrast display. The inherent contrast ratio of the electroluminescent display is quite good—40:1 minimum (i.e., greater than 20 foot-lamberts divided by less than 0.5 foot-lambert—see Fig. 4). The problem is that reflections reduce the contrast to your eye considerably. The mirror-like aluminum back electrodes (rows) are a major cause of contrast reduction.

The Integral PC has a plastic window in front of the display. We chose to reduce reflection off the front window surface by using a matte surface texture. This diffuses incident light, making reflections from the front surface of the window less noticeable. However, too much texturing can cause blurring.

The reflections from the various internal surfaces of the electroluminescent panel are nearly eliminated by a circular polarizing film laminated to the back of the window. This film is formed by a linear polarizing film followed by a thin-film quarter-wave plate. As incoming light passes through the window, it is first polarized in one direction by the linear polarizer and then rotated 45 degrees by the quarter-wave plate layer (see box on page 21). The outgoing reflection is then rotated another 45 degrees by the quarter-wave plate layer, making its polarization angle 90 degrees relative to the linear polarizer. Thus, the reflected light is blocked. Desirable emitted light from the display is attenuated about 50% by the circular polarizer. However, the increase in contrast more than offsets this loss in terms of perceived sharpness and readability.

Acknowledgments

I would like to acknowledge the display development team at HP for their part in making it happen—Bill Eaton, Gene Cooper, Steve Brown, Tim Harper, Neal Andrews, and Brian Spreadbury. I also want to thank the many people outside of HP who made this display their number-one priority.

References

1. G. Destriau, "Recherches Experimentales sur les Actions du Champ Electrique sur les Sulfures Phosphorescents," *Journal de Chimie Physique*, Vol. 34, 1937, pp. 117-124.
2. M. Takeda, et al, "TFEL Displays of Commercial Production Version," *1983 Proceedings of the Society of Photooptical Instrumentation Engineers*, pp. 34-38.

Further Reading

1. I. Linden, et al, "A 256×512 Graphics EL Display Module," *1984 Digest of the Society for Information Displays*, p. 238.
2. C.N. King, et al, "Development of a 256×512 Electroluminescent Display Monitor," *1984 Proceedings of the Society of Photo-optical Instrumentation Engineers.*
3. T. Gielow, "Electroluminescent Driving Techniques: Their Promise & Problems," *1983 Proceedings of the Society of Photo-optical Instrumentation Engineers*, pp. 42-48.
4. *Display Driver Handbook*, Texas Instruments, Inc.

Mechanical Design of the Integral PC: Not Just a Desktop Computer with a Handle

by Thomas A. Pearo

THE MAIN CHALLENGE in the mechanical design of the Integral PC was to design a full-function personal computer in a transportable configuration. We wanted to expand on the tradition of the integrated design used for the HP 85 Computer[1] by updating to an industry standard processor, a state-of-the-art 80-column ink-jet printer, a fast flexible-disc mass storage device, and a state-of-the-art 80-column-by-24-line flat-panel display. We also wanted to retain the capability for removable I/O modules.

To build this feature set into an HP-quality transportable package that would not use up massive amounts of valuable desk space like many desktop systems required rethinking conventional transportable design concepts. After looking at the many ways these components could be arranged, our industrial design department selected the concept of an upright package to meet sound mechanical design goals and make the Integral PC easy to use.

Many of the aspects of the human interface are addressed by the Integral PC's mechanical design, beginning with a detachable, low-profile keyboard. The high-contrast flat-panel display is mounted in a near vertical position (offset 5 degrees) for ease of viewing. The printer is mounted at the top of the package for easy access and paper handling. The whole system folds up into a compact, rugged, easily carried unit.

Ruggedness to withstand shock, vibration, and handling was a primary design goal of the mechanical team. Extensive testing and development of the components and package were required to meet this goal.

Another area of emphasis from the very beginning was electromagnetic compatibility and susceptibility to electrostatic discharge (ESD). Our approach was to suppress electromagnetic interference (EMI) at the source. The traditional concept of a metallized plastic case sealed with conductive gaskets was replaced with good printed circuit layout and metal ground planes and shields both to suppress EMI and to add physical strength to the unit. The switching power supply is completely enclosed in a metal box and the logic assembly uses local metal shielding.

Case Design

To meet the ruggedness and other design requirements,

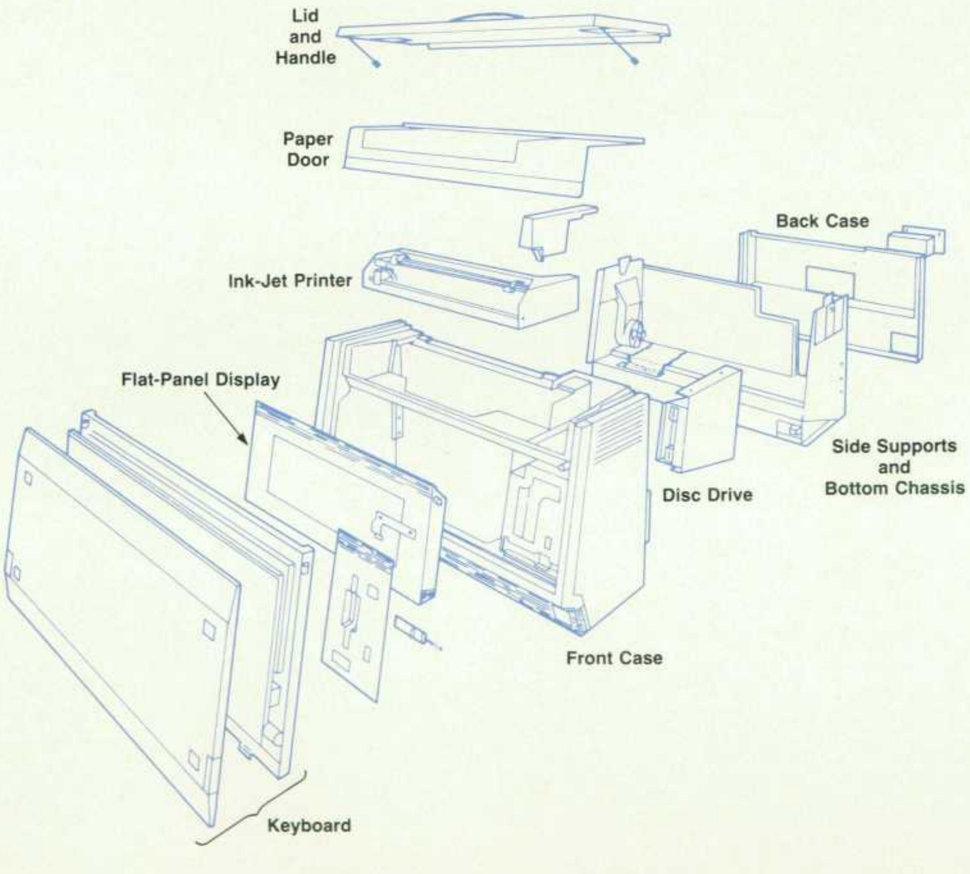


Fig. 1. *Exploded view of package assembly for Integral PC.*

an assembly concept based on a one-piece front case evolved. The front case is a large plastic part (measuring 425×295×184 mm) into which the rest of the computer is built. (See Fig. 1 for an exploded drawing of the Integral PC.) This concept also met our requirement of being able to build, test, and repair the product as a complete electrical unit with the back case removed.

Styling dictated that the large flat sides of the front case have no draft. (Draft is a slight taper on a part which allows its easy removal from the mold.) These no-draft surfaces required a large slide, or pull, on each side of the plastic mold. A slide is a mold surface that moves away from the part surface as the mold opens, usually in an axis perpendicular to the mold's opening axis. The features needed for mounting the printer mechanism at the top of the front case required a third large slide in the mold.

In addition to the large part size and three large pulls, a hydraulic core pin was required near the center of the mold for forming a pivot hole for the computer's display, perpendicular to the mold's opening axis. In addition, critical dimensional requirements, lots of detail, and high cosmetic requirements made this part a plastic tooling challenge. Another complicating detail on this part was an intentional undercut (or reverse draft) on the inside surface of each side to form attachment points for the side supports (see Fig. 2). Although this normally is not good plastic molding practice, the finished front case proved flexible enough to snap over the mold core for removal with no permanent deformation.

This was the largest and most complex molding tool we had ever attempted. The raw tool steel (prehardened P20 steel) blank size was 838×1016×762 mm and the finished mold weighs about 7,500 lb. The mold required about 3,500 hours of work and has performed very well.

Handle and Lid Design

Because the Integral PC is a transportable product, special emphasis was given both to the comfort of the person carrying the unit and to the structural integrity of the unit during transport. Several handle designs were tried before the present special handle and handle mounting method were selected. To support the 25-lb weight of the Integral PC and to assure the customer of a feeling of quality while transporting (no rattles, bounce, or quivering), we designed a weight or load transmitting substructure. The thin metal strap in the interior of the handle is connected through a die-cast end cap to an aluminum lid carrier plate under the plastic lid (Fig. 2). This plate stiffens the lid during lifting and carrying and transmits the static load to a die-cast bottom latch. The bottom latch engages the die-cast lid rocker, which has a pin that goes through a molded hole in the side support.

The 10% glass-filled side supports (one on each side of the unit) carry and transmit the weight to the front case (the large structural plastic part mentioned before) and to the bottom chassis. The bottom chassis is a sheet-metal pan on which the power supply, I/O cage, logic assembly, and part of the disc drive mechanism are mounted. It acts as the back panel, the central grounding point, and a portion of the EMI shielding. The rubber feet that support the Integral PC are also attached to the bottom chassis.

The lid of the Integral PC serves several purposes besides carrying the weight. The lid also protects the printer and printer door and captures the keyboard in the transport mode. A high priority was placed on designing a lid that folded over the top and down the back of the unit and stayed attached and out of the way during operation. A user shouldn't have to waste time looking for a place to store the lid during setup or finding and realigning the lid when closing the unit for transport. To solve this problem, we designed an interesting two-piece mechanical linkage (Fig. 3) using the rocker and the lid bail to guide and support the lid in the open position.

Another interesting area of the lid design is the molded plastic ratchet detail for the lid latch (Fig. 4). This design reduced the part count by using the properties of the plastic polycarbonate material to mold a bending beam with a ratchet surface. To engage the lid latch, the user simply pushes the top latch toward the outside of the package. This engages the moving ratchet teeth on the top latch with the fixed ratchet teeth on the lid to prevent the bottom latch (see Fig. 4a) from backing out of the rocker. To disengage, the user pushes down on the latch button (see Fig. 4b), deflecting the plastic fixed ratchet surface down and disengaging the teeth. While still holding the latch button down, the user then slides the top latch inward, releasing the bottom latch from the rocker and allowing the lid to be lifted and rotated to the storage position.

Display Mounting

One of the most interesting and challenging aspects of the Integral PC design was incorporating the flat-panel display. We knew that because of space, weight, and power constraints we would use a flat-panel display, but what type and size were unknown until well into the mechanical design. After allocating a space (height from one display technology and width from a second), we went on to address problems such as the fragility and the thermal aspects

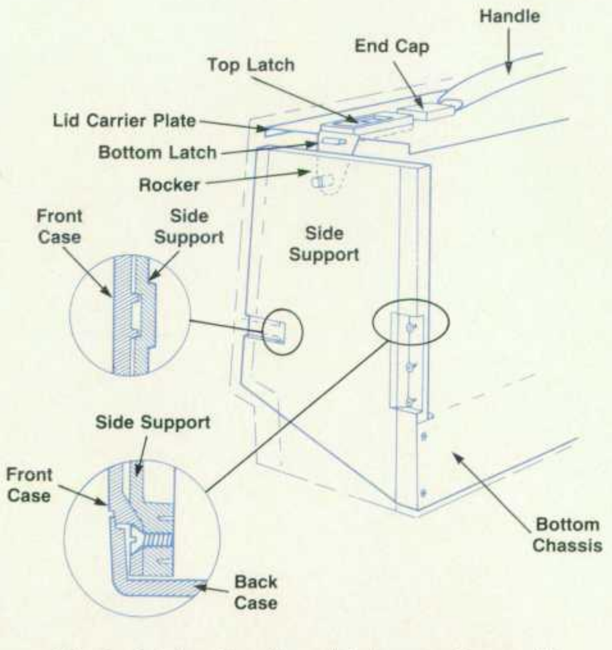


Fig. 2. *Details of handle and side support assemblies.*

of flat-panel displays. We established shock and vibration specifications and then began a process of evaluation, critique, and interchange with vendors.

Since all candidates had a fairly common glass structure, we began to do fragility testing with regular window-type glass. Early testing resulted in several HP designs to evaluate both glass mounting and electrical interconnect (between glass and printed circuit board) technologies. Since glass is strongest in compression, we designed a foamed polycarbonate plastic frame to mount the display. This part minimizes the torsional and tensile forces that could be applied to the glass during rough handling.

The display mount frame also incorporates the details that allow the display to be rotated through 12 degrees (5 to 17 degrees in 2-degree increments) to adjust viewing angle and reduce glare. A simple but effective display latch is produced by a molded plastic blade which is spring loaded in a slot in the front case. This engages a molded ratchet feature on the side of the display mount frame. A spring holds the display in the out position, and a special plastic dashpot assembly provides the required damping to give the desired feel during adjustment.

To protect the display from ESD damage, a plastic window covers the display. It is attached to the cosmetic bezel with a 1/8-inch wide glue joint around its perimeter.

Upon selection of the electroluminescent display (see article on page 12), an extensive investigation was made to enhance display quality and reduce glare. Windows of various materials, colors, and surface finishes, and band-pass and high/low-pass optical filters were evaluated. Although the electroluminescent display provides an inherently bright and sharp high-contrast display, its construction presents a very reflective back surface (like a mirror). We found that a thin-film circular polarizer applied to the back of the polycarbonate display window reduced reflective glare and gave a greater contrast while reducing the apparent brightness only slightly (see box on page 21).

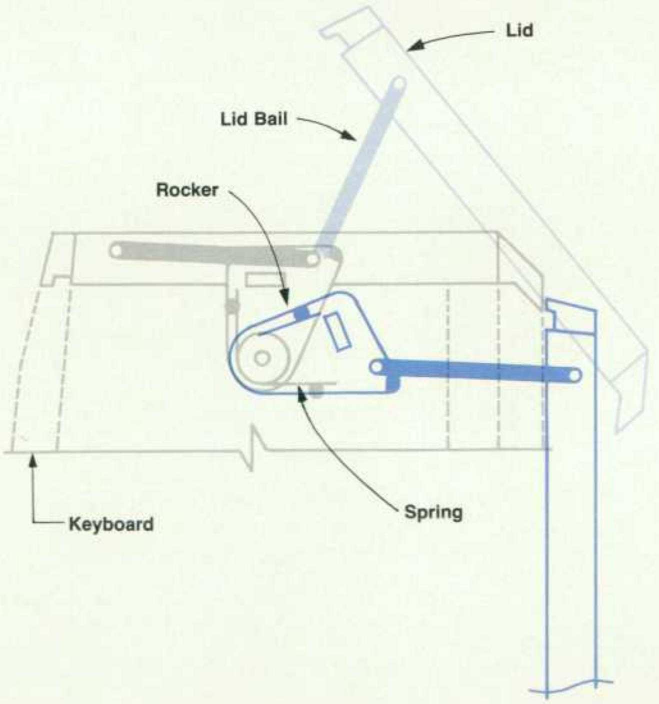


Fig. 3. *Top lid hinge mechanism. (Gray) Closed. (Shaded color) Partially open. (Solid color) Open.*

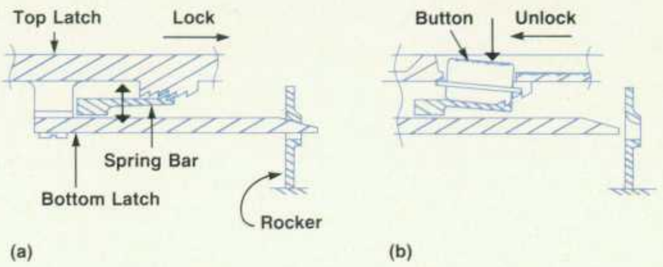


Fig. 4. *Latch operation. (a) Locking, edge cross section of mechanism (button not shown). (b) Unlocking, center-line cross-section of mechanism showing caged release button. See Fig. 2 for perspective view.*

Printer Mounting

Our original concept for the Integral PC was to mount the printer in the center of the package. It quickly became obvious that paper loading and handling required the printer at the top of the package. To cover the printer and muffle noise during operation, a smoked polycarbonate part, the paper door which hinges in the front case, is used. In the open position, the paper door would have been vulnerable to damage if grossly overloaded. To prevent damage, a clever design using a tapered hinge pin allows the paper door to snap out instead of breaking the hinge pin if the door is overloaded. Then the paper door can be snapped back into place easily.

Keyboard

Since the keyboard is the most often used part of a computer system, considerable effort was put into its design. Most important is that the keyboard detaches from the computer and allows users to place it wherever they feel it to be most comfortable, even on their lap. Careful attention went into the design so that the home row keys meet the 30-mm European height recommendation when the keyboard is in the flat position.

For users who prefer a more traditional keyboard position, integral legs fold out to put the keyboard at an 8-degree angle. These spring-loaded legs, which fold into a recess in the side of the keyboard when not in use, are designed to be an inexpensive part, partially assembled at the vendor and easily loaded into the case at final assembly. To move the legs to the extended position, each leg is pulled directly out from the keyboard body to clear it from its recessed position. The spring stop strikes a surface on the bottom case and prevents the keyboard leg from being pulled out of the assembly. The leg is then rotated a little over 90 degrees to its down position, where the spring stop strikes a rib in the bottom case to prevent further rotation. Then the leg is released, and the spring draws the leg back into the recess, which captures the leg in the extended position and prevents further rotation.

Another feature that reduces setup and tear-down time is the cord storage trough built into the keyboard assembly. The coil cord is permanently attached to the keyboard and readily accessible when stored in the trough. Special plastic

Reducing Glare with Circular Polarizers

Without getting too deeply into the several theories used to describe the properties of light, here is a brief description of how a circular polarizer works (see Fig. 1). A circular polarizer is a combination of a linear polarizer and a quarter-wave plate, or retarder. This plate is formed by an anisotropic material that has orthogonal fast and slow directions for light propagation. When a ray of randomly polarized light strikes the electroluminescent flat-panel display of the Integral PC (since the display emits all the light we care about, this random light is considered glare) and passes through the linear polarizer, it is polarized, or oriented, in one plane. When this polarized light passes through the quarter-wave plate, whose fast and slow propagation axes are oriented at 45 degrees to the light's polarization, the light's polarization is given a helical twist. When the twisting light polarization reflects off the highly reflective back surface of the display, the twist is reversed. As the reflected light passes back through the quarter-wave plate, the twist is cancelled and the light exits with a linear polarization 90 degrees from its original polarization plane. This results in the light (glare) being blocked by the linear polarizer and therefore, becoming not visible or bothersome to the viewer.

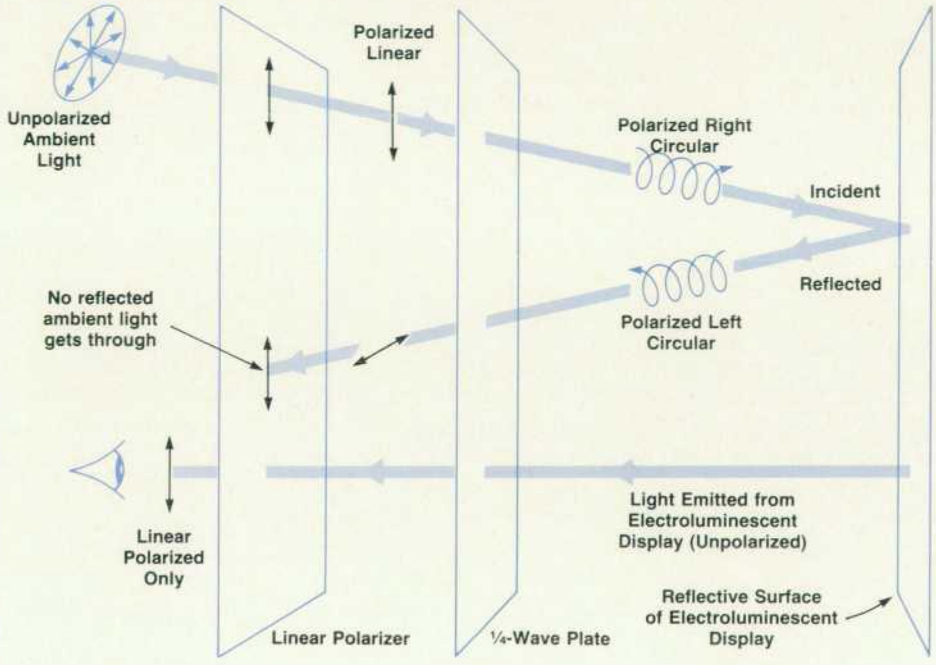

Unpolarized Ambient Light
Polarized Linear
Polarized Right Circular
Incident
Reflected
Polarized Left Circular
No reflected ambient light gets through
Light Emitted from Electroluminescent Display (Unpolarized)
Linear Polarized Only
Reflective Surface of Electroluminescent Display
Linear Polarizer
¼-Wave Plate

Fig. 1. *Reduction of glare reflected from the Integral PC's flat-panel display by using a circular polarizer on the back side of the display window.*

fingers are molded in the trough to retain the coil cord and allow for variations in coil diameter. To set up the keyboard, the user simply reaches down, grabs the connector, pulls out the cord, and plugs it into the mainframe. The procedure is reversed for easy tear-down. The keyboard can now be returned to its storage location, in a vertical position at the front of the unit. The importance of this design is that in addition to using a minimum volume, the keyboard covers and protects the display during transportation.

Since the keyboard stands in nearly a vertical position before being captured between the front case and lid, and because of fabrication tolerances, we developed some injection-molded rubber tolerance adjusters to prevent the keyboard from tipping over or rattling during transport. These are molded in the same color as the plastic case parts and blend into the overall design nicely.

Noise Dampening

Another aspect of a friendly system is minimal acoustic pollution. That was our design goal even though the Integral PC has a built-in printer and flexible-disc drive mechanism. It is fortunate that we are using a version of HP's ThinkJet Printer mechanism, because it is inherently quiet and can be packaged without extensive acoustic insulation.

When we had to add a fan for cooling, we used a small dc motor which runs at about 60% of rated speed to reduce fan-tip noise. Special internal mounting and a custom exhaust duct were designed into one of the side supports. After initial testing, special acoustic mounts were developed to isolate fan motor noise from the rest of the structure.

Special rubber tolerance adjusters similar to those already described for the keyboard are used on both the display assembly and the paper door to prevent rattles and vibration-induced noise. In idle mode, the Integral PC is very quiet, measuring only 35 dBA, using the ISO/DP7779 measurement criteria.

Plug-In ROM Module

An interesting feature of the Integral PC is the customerreplaceable ROM package that contains the operating system. This gives both HP and the Integral PC owner great flexibility in updating the operating system and adding features and has advantages for HP in manufacturing and the easy loading of service diagnostics. This design requires an inexpensive, yet reliable and very compact package. An access door is provided in the back case, and a plastic guide is attached to the main logic printed circuit board to guide the ROM module into place.

For a reliable, yet inexpensive interconnect, we chose a variation of a "tree" connector that is used in several HP calculators. The ROM package consists of a top, a bottom, and two handles and has the capacity for two printed circuit board assemblies. The ROM bottom has a positive latch to hold the assembly in during shock. This latch, a plastic cantilever structure molded into the ROM bottom case, is pushed out of the way by the handle movement to allow the package to be removed.

Acknowledgments

In any project of this size, it's impossible to credit everyone who has contributed to its success. To those who go unmentioned, we send our thanks.

The Integral PC is the work of the mechanical design team of Tim Jondrow, Steve Brown, Bill McDonough, Rick Olson, and Joseph Scheffelin, with help from Brian Spreadbury and Scott Hickman. Our thanks to Chuck Dodge (who originated the upright package concept) and Tony Laidlaw of the industrial design department for aesthetic direction, and to Gene Deshon, Govind Deshmukh, and Horst Irmsher for advice on part design and procurement. Thanks also to Teri Wilson for project coordination and to Carl Hill, Orland Upton, and Colleen Boeger and her production crew for their feedback and prototype building.

Jerry Cornelius, Wayne Downs, Bill Dixon, and the rest of the Corvallis shops did their usual high-quality work. John Milton, Leonard Rosi, Gus O'Donnell, and the PEL group provided support and testing.

Reference

1. Complete issue, *Hewlett-Packard Journal*, Vol. 31, no. 7, July 1980.

A UNIX Operating System Adapted for a Technical Personal Computer

by Ray M. Fajardo, Andrew L. Rood, James R. Andreas, and Robert C. Cline

DEVELOPING THE HP-UX operating system for HP's Integral Personal Computer provided a number of very significant challenges:

- Eliminating the need for a hard disc and allowing the installed flexible disc to be removed whenever a disc access is not occurring.
- Eliminating the need for complex system configuration and user administration of the operating system.
- Simplifying the human interface with the aid of a windowed environment to make the system easier to use.
- Providing a measure of real-time capability for instrumentation control.

The above challenges were to be met, while at the same time preserving compatibility at the programming level with the UNIX™ System III™ operating system and HP's HP-UX-based computers.

Conceptual Organization

The UNIX operating system[1] is conceptually organized as multiple layers of software and thus allows both traditional UNIX facilities and added HP-UX features to coexist without collision. In a traditional UNIX system (Fig. 1), the outermost software layer is called the shell. It provides the primary user interface to the remaining software in the system. The shell is essentially a command interpreter that runs other programs in the system. Since the shell is itself a program, variations of the shell can and do exist. HP-UX systems typically deliver the Bourne shell sh and C language shell csh as the two dominant variations.

Below the shell is the utility library, some 300 individual programs that perform a wide variety of functions. These programs account for the massive size of the UNIX operating system, and the need to have them on-line in the traditional multiuser environment is a principal reason (although not the only one) that UNIX systems require a hard disc. This approximately six megabytes of software generally can be divided into communications, text processing, software development, file manipulation, and system administration.

Below this software layer is the UNIX system kernel. The kernel provides the common run-time services required by UNIX programs and manages the multitasking machine resources available to them all. Within the kernel are the software drivers that control the specific I/O hardware in the system. Thus, any special disc, display, or other peculiar hardware characteristics are generally isolated here.

In the Integral PC, this concept model is expanded (see Fig. 2). The outermost layer of system user interface is provided by PAM—HP's Personal Applications Manager (see box on page 30). This computer user shell is the prin-

UNIX and System III are U.S. trademarks of AT&T Bell Laboratories.

cipal means of user interaction with the system. Other UNIX shells can also be run from PAM, as can any other program in the system.

The programs or applications occupy the next layer of software. New applications made available with the Integral PC include Technical BASIC for instrumentation control and a variety of PC-type application software, including terminal emulation. A number of easy-to-use system utilities (e.g., a format-disc utility and a font editor) are also provided. These are in addition to the previously described standard UNIX software.

The next layer of software in the Integral PC consists of the run-time services commonly available to programmers. Within the HP-UX operating system, this level of programming interface includes not only the UNIX kernel, but also the HP kernel extensions and libraries.

Libraries exist for two reasons. First, together with the kernel, they present a programming interface for applications that can provide source compatibility among HP-UX products. Second, since libraries reside in user memory with the application that needs them, memory is used for them only as necessary. When a particular capability is used by all applications that run on a given system, then it is useful to include the facility within the kernel. Such is the case with HP Windows, an HP kernel extension designed to allow transparent use by applications. Architecturally, HP Windows is structured to include a variety of window types.

One such window type is called TERM0, which supports the programming characteristics of an HP or ANSI-escape-sequence-driven terminal equivalent to a nonblock mode HP 2622 Terminal. A second window type supports the HP-GL plotter language and simulates an HP 7470A Plotter.

Other kernel extensions include certain real-time services such as real-time priority. (This will be discussed later.)

Finally, software drivers are provided within the Integral PC to support the expected complement of peripherals, including built-in and optional devices. These include a wide variety of hard discs, letter-quality printers, and HP plotters.

In the Integral PC, 256K bytes of executable ROM contains the kernel, the peripheral I/O drivers, the windowing system, and PAM. The remaining software resides on disc memory, either flexible disc or hard disc, depending on the software and the needs of the user.

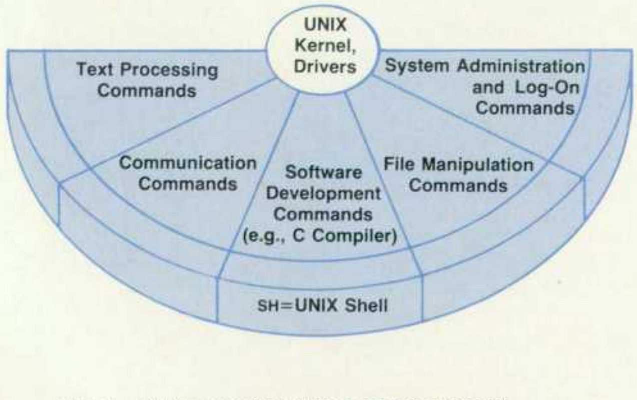

UNIX Kernel, Drivers
Text Processing Commands
System Administration and Log-On Commands
Communication Commands
Software Development Commands (e.g., C Compiler)
File Manipulation Commands
SH=UNIX Shell

Fig. 1. *UNIX operating system concept model.*

Operating System Kernel

The Integral PC uses the HP-UX 1.0 operating system. This multitasking system allows users to solve several problems simultaneously. It provides a uniform logical device interface in which devices are manipulated in the same manner as text files, and a hierarchical file system that allows the user to organize large amounts of secondary storage in a straightforward and convenient manner.

Memory organization. The Integral PC's memory is organized as indicated in Fig. 3. The 68000 CPU has 24 address lines, which enable it to manipulate 16M bytes of address space. The upper 8M bytes are reserved for RAM and the lower 8M bytes are reserved for I/O and ROM. The built-in (internal) 512K-byte RAM address space is located just above the 15M-byte boundary. The plug-in (external) RAM cards are configured to build down from the 15M-byte boundary. The system ROM resides in the 0-to-1M-byte address space and the remaining space (from 1M to 8M bytes) is used for I/O device mapping. Mapped devices include the memory management unit, flexible disc controller, graphics processor unit, keyboard controller, beeper, clock, plug-in memory control registers, and plug-in I/O card control and data registers.

The internal 512K-byte RAM is a critical resource with many demanding uses. This RAM is used by user processes that are created and destroyed during normal operation. To simplify and optimize use of this RAM, the Integral PC has a built-in RAM disc that will be discussed later. The RAM disc blocks are dynamically allocated and freed as files are stored into and removed from the RAM disc. The Integral PC's operating system allows drivers to be configured dynamically into and removed from the operating system, which requires the allocation and deallocation of system RAM. Furthermore, the operating system itself can be enhanced dynamically to emulate other operating system versions or to mask ROM bugs. This also requires dynamic RAM allocation. To satisfy these needs, the operating system maintains two memory pools: a user memory pool and a kernel dynamic memory pool.

The system RAM is organized by the operating system as indicated in Fig. 3. The operating system RAM image occupies the top of physical memory. Immediately below is the kernel dynamic memory pool. This area is used by dynamically loaded drivers (see following section on installable drivers), window creation, and the RAM disc driver and, in general, to fulfill any operating system requirements for dynamically allocated memory. Immediately below this operating system dynamic memory pool is the user process memory pool, which contains all remaining memory. As memory is freed by the operating system, it moves from the kernel dynamic pool into the user process pool. As the kernel requires more memory for its dynamic memory pool, memory is moved from the top of the user process memory pool into the kernel dynamic pool. The operating system automatically manages this movement of memory between the kernel and the user process memory pools. In the quiescent state, all memory resides in the user process pool.

The HP-UX operating system achieves multitasking via multiple processes. A process is an entry in the system process table and possibly some user memory allocated to

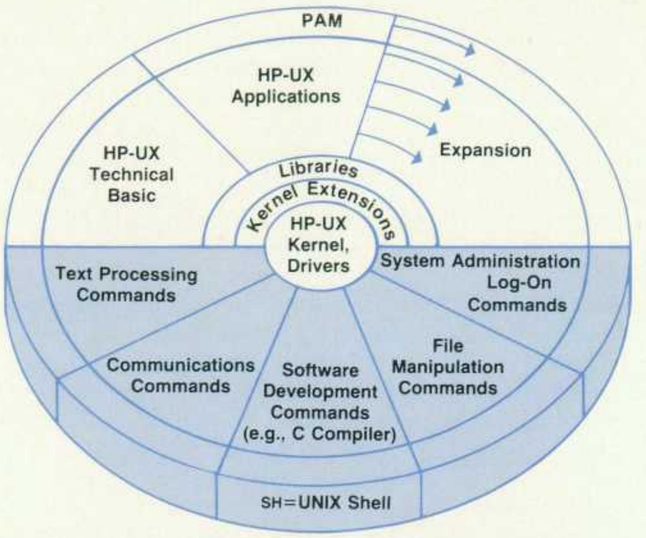


Fig. 2. *Expansion of UNIX concept model in Integral PC.*

the process. As processes are created, they are allocated memory at the bottom of the user process pool. As they are destroyed, the freed memory is returned to the user process memory pool.

The operating system ROM space contains the operating system ROM image. The data portion of this ROM image is copied into RAM during start-up. The code portion of the operating system ROM image is executed directly in ROM. The ROM also contains a ROM disc in which the programs init, daemon, and PAM are located. The ROM disc also contains a small environment file and a small symbol table describing the operating system ROM image.

Processes. Processes are the most important abstraction of the HP-UX operating system. It is through the work of a process that users actually solve problems and generate results using the Integral PC.

The user process memory pool is the largest memory pool, and is the area in which user applications run. A process consists of a user side and a system side. On the user side are a program text area, a data area, and a user stack area. The data area can grow upward dynamically through the use of the sbrk and brk system calls. The user stack can grow downward dynamically as more user stack is required. The operating system also consists of a text area, a data area, and a system stack. The text area is located primarily within the operating system ROM image and is shared among all processes active on the Integral PC. The data area for each process is composed of an area called the u_area, which is unique to the process, and a global data area, which is accessible to all processes and resides in the operating system RAM area. Finally, each process has a dedicated system stack of fixed size.

All processes are created by duplicating existing processes, except the first process (process 0). The first process, the scheduler, is handcrafted by the operating system during the start-up sequence. Each process's system side is similar and provides access to the traditional HP-UX system capabilities. These capabilities are best described as a collection of calls which can be made by the process's user side to the process's system side (i.e., requests to the operating system). These system calls, or kernel calls, are the primary operating system interface mechanism. These calls define the operating system capabilities from the user's program perspective.

Kernel Calls. The complete HP-UX implementation includes many megabytes of object and source code. In the context of the Integral PC, the system is restricted to the kernel calls. However, the Integral PC does offer a majority of the allied commands and libraries available in the complete HP-UX offering.

By restricting the HP-UX system to only the kernel calls (intrinsics), its size is reduced substantially. Application commands that complete the Integral PC software offering are provided on separate discs. By unbundling the HP-UX operating system in this way, the Integral PC is able to provide it in a low-cost microcomputer package that is extensible to include the entire system. The kernel calls are serviced by the operating system ROM. Because this ROM includes only the kernel and a small ROM disc, the Integral PC is able to provide HP-UX capability in just 256K bytes of system ROM.

Compatibility and Flexibility. The most important feature of the operating system is compatibility. In addition to the compatibility with the UNIX System III and HP-UX 1.0 operating systems mentioned above, the Integral PC provides a BASIC language interpreter that is compatible with the BASIC implementation used in HP's earlier Series 80 Computers.

In addition to this strong goal of compatibility, the Integral PC has a strong commitment to flexibility. It can be configured at the operating system level to emulate other operating system versions, allow the dynamic addition and removal of drivers, and accommodate operating system en-

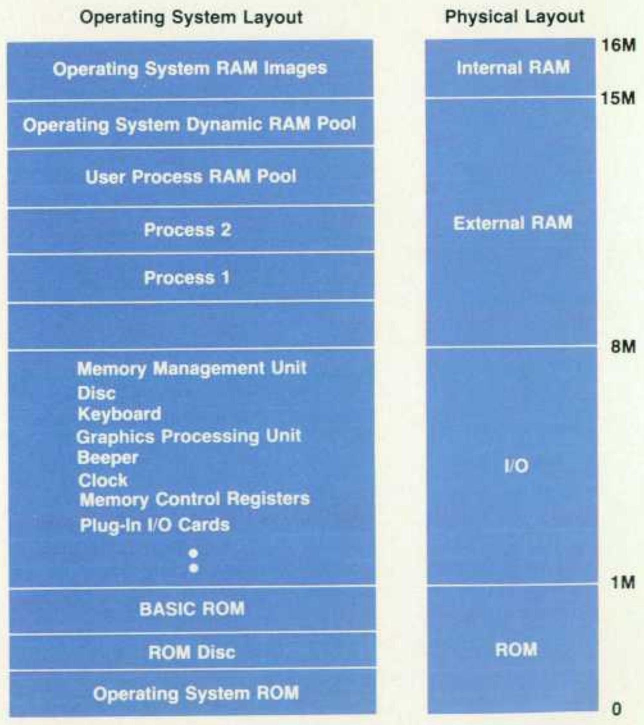


Fig. 3. *Integral PC memory organization.*

hancements and bug fixes.

Installable Drivers. The Integral PC operating system is designed as an open system. All major internal functions within the operating system ROM are linked together indirectly through a RAM jump table initialized during start-up. Function linkage occurs normally except that the actual function transfer of control occurs by jumping into the RAM table, which in turn jumps to the target function.

The jump table contains an array of processor jump instructions. Isolating the operating system ROM function code in this manner allows individual routines to be masked by disc-based patches. These patches modify the jump table to point to replacement functions that have been loaded into the kernel dynamic RAM area and linked to the existing kernel data structures. Hence, any aspect of the operating system may be modified. Perhaps the most radical example of such patching is described in the following section on real-time extensions. For the real-time enhancements, the kernel scheduling function is replaced with a new scheduling function.

This jump table linkage provides a very flexible environment that allows arbitrary operating system customization. The Integral PC goes beyond this jump table flexibility to accommodate installable drivers as well. The installable driver mechanism operates in a manner similar to this jump table linkage.

In the HP-UX context a device driver must conform to a very stylized table-driven interface to the rest of the operating system. This interface is composed primarily of three tables: the character device switch table, the block device switch table, and the interrupt service routine table. These tables all include empty entries that are available for drivers that have not been compiled directly into the operating system. Installable drivers can link to these tables dynamically. When the operating system is up and running, a user process may request that a new driver be installed into these tables, or that a previously added driver be removed.

Many advantages result from this installable driver capability. One large advantage is that a user does not need a source license to add custom drivers to the Integral PC. By conforming to the well-defined driver interface, a new driver can operate with the operating system without being compiled with the operating system source. Another advantage of this installable driver technique is that it optimizes RAM use. Only those drivers actually required for a particular application need be present when that application runs. A later application that requires different drivers can remove the drivers it does not need and add those it does.

An additional benefit of this installable driver strategy is a fast start-up sequence. Because drivers can be installed on the fly, the operating system is not required to do a start-up configuration of all devices and drivers. This, in combination with the ROM-based operating system, yields a very quick start-up. After power-on the Integral PC is ready for use in only 10 seconds. If the user has an external hard disc, that disc is available just 10 seconds later (assuming the disc itself is left powered on), and during this additional 10 seconds the Integral PC is already available for work.

In addition to being able to add drivers when the operating system is up and executing and patching ROM bugs dynamically, the Integral PC can dynamically add kernel enhancements of all sorts. Even new kernel intrinsics (kernel calls) can be added dynamically. Perhaps the best example of this is in the area of real-time control.

Real-Time Extensions. The UNIX operating system by virtue of its multitasking nature has some drawbacks in the real-time control environment. One example of the type of problem that can arise involves high-speed, real-time data acquisition programs. Such a task might require readings from an instrument every millisecond for ten seconds (10,000 readings). Although the Integral PC's I/O is capable of this type of reading frequency, the multitasking scheduling policy can obstruct the data gathering process. In particular, with the standard scheduling policy, each of the currently active processes is given a 200-ms slice of CPU

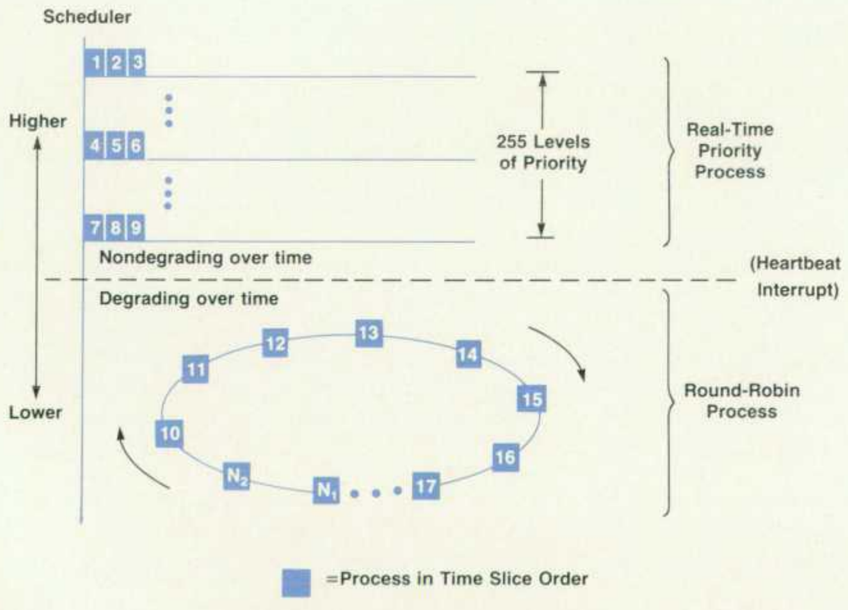


Fig. 4. *Real-time priority and scheduling scheme used in the Integral PC operating system.*

time. After 200 ms, a heartbeat interrupt occurs and the executing process is put to sleep, that is, has its state saved and is made dormant. After each of the other competing processes has had its slice of CPU time, the initial process is again allotted another 200 ms of CPU time, and so on for each process.

To the user, this process delay of several hundred milliseconds is not noticeable and the processes appear to be operating simultaneously. For the data acquisition process, however, this dormant period while other processes execute is not acceptable. During this time several hundred instrument readings can be lost.

To strengthen this area, the HP-UX definition in the Integral PC has been augmented to include a collection of kernel calls that directly address the needs of real-time control applications. The Integral PC provides some of these real-time extensions in the system ROM and others on disc as dynamic kernel enhancements. When enhanced with this collection of real-time kernel calls, the Integral PC can act as a very powerful real-time controller. The real-time extensions include real-time nondegrading priority-based scheduling, reliable signals (modeled upon the signal facility defined in the University of California at Berkeley's UNIX implementation), shared memory (modeled on the UNIX System V™ shared memory concept and provided in system ROM), memory locking, file synchronization, and a high-resolution clock and timers. In general, the real-time extensions are based on industry standard or HP standard solutions to the requirements of real-time programs. This discussion will address only the scheduling enhancement rtprio.

The first step to understanding rtprio is understanding the standard system scheduling technique (see Fig. 4). The Integral PC uses a round-robin scheduling algorithm for its normal operation. In this algorithm all active processes are given slices of CPU time as mentioned earlier based on their current priority. As a process executes, its priority degrades (i.e., it moves to a lower priority). When the currently executing process has executed for its allotted time, that process is "sliced out." The processor state is saved, the process state is saved, and some other process state and processor state are activated. This new process now executes for its allotted time. This occurs in a round-robin fashion where each process in succession is given a time allotment. When all processes have run for their specified time, the cycle starts over; each process in turn is given another time allotment, and so on.

Individual kernel calls are not preemptable. If an interrupt occurs during a kernel call and the running process is to be put to sleep, then that process will complete the kernel call it is currently executing before it is put to sleep. User mode execution is preemptable at any time. Thus, in the normal Integral PC scheduling algorithm, a process is subject to being sliced out whenever it is executing in its user side. The lower portion of Fig. 4 indicates this round-robin scheduling algorithm occurring at the base level of the scheduler.

Some real-time processes may desire to run to completion rather than being subjected to this algorithm. A running process may need to continue running until the process itself is ready to give up the CPU. The Integral PC provides run-to-completion scheduling at each of 255 different absolute (nondegrading) priority levels. By dynamically loading a new scheduling algorithm into the Integral PC operating system when needed, the operating system can be enhanced to perform both round-robin and run-to-completion scheduling.

System V is a U.S. trademark of AT&T Bell Laboratories.

The run-to-completion scheduling takes place at a higher level than the round-robin scheduling. To be placed in the run-to-completion scheduling algorithm, a process executes a rtprio kernel call. This makes the calling process a real-time process. As long as any real-time processes are ready to run, they are scheduled and executed using the run-to-completion algorithm. After all real-time processes have stopped (i.e., gone to sleep), the scheduler resumes round-robin scheduling among those processes that are not real-time. Kernel calls remain atomic. Even with run-to-completion scheduling, the Integral PC does not do preemptive scheduling. Once a process has started a kernel call, that call will be allowed to complete before the process is sliced out. Once a real-time process begins executing, it will run until it blocks, or until a higher-priority real-time process is made ready to run. The real-time processes are organized into 255 separate priority levels. Processes at the higher levels execute before processes at the lower levels, and all real-time processes run before any round-robin processes.

Start-Up Procedure. As the Integral PC powers on, it automatically structures itself for a local language requirement. The language to be used is determined by the attached keyboard. Each keyboard has a key layout that complies with local customs and is identified by its keyboard identifier number. The start-up process simply asks the keyboard driver for its identifier number to determine the language to be used. The Integral PC has softkey menus in English built into the operating system ROM. These menus can be replaced by menus in the local language. The start-up process automatically searches the disc in the internal drive for the localized menus. The Personal Applications Manager is also available in localized versions. The start-up process searches the internal disc for a localized version of PAM. If a localized PAM is found, the start-up process executes it. Then PAM commands are given in the native language. If a localized PAM is not present on the disc, the English version of PAM built into the system ROM is executed.

The Integral PC then executes a process (scan_discs) to look for disc drives attached to its built-in HP-IB (IEEE 488) interface. When it finds a disc, it automatically makes the disc available to the user. First, the process creates a folder for the disc. Then it mounts the disc to the folder. This means that when the user opens the folder using PAM, the contents of the topmost file on the disc will be displayed. The process also creates a device name in the device folder /dev which acts as a handle for the disc. The name is used when the user wishes to find out about the disc, such as how many bytes of unused space are left.

File System

One of the key operating system issues with the Integral PC was making the operating system work without a hard disc and work well in a flexible disc environment. In general, a UNIX-based system requires a hard disc to anchor its file system and to provide swap space. Also, with only a built-in flexible disc drive for mass storage, much had to be done to make using it efficient while providing good means of maintaining the integrity of the information on the disc. This is difficult when one considers how often a user may exchange flexible discs in using the system. Hence, the elimination of a hard disc presented quite a few challenges.

Dealing with Swapping. The first problem was what to do about swap space. This was also the easiest problem to solve. The only guaranteed disc available was the internal flexible disc. For performance reasons, this option got little serious consideration. This left the Integral PC without any disc for swapping. This is when the issue of why any swapping is necessary came up. The Integral PC can support up to 7.5M bytes of memory. Given this capacity for memory, it is easier to acquire more memory for the execution of memory-consuming applications than to allow copying them in and out of swap space.

Providing a Root for the File System. The problem of how to set up the file system was much more difficult. A UNIX system has a hierarchical file system anchored to a directory called / on an associated hard disc. This root file system also contains device configuration information for the system in a directory called /dev. This information provides the sole means of interacting with peripheral devices by applications. The solution to this problem is to build a RAM-based emulation of a disc device and use this device as the home of the root file system and its key components.

As it turns out, the RAM disc is one of the more useful features of the Integral PC. It serves as a root for the file system, as a place to keep the directory /dev that controls the means of accessing devices, and as a place for the directory /tmp where most applications create scratch files. It even provides a nice place to attach other file systems on other disc drives. Whenever a file system on a piece of disc media is mounted (i.e., attached to the root file system for general use), the volume name of that file system is used to create a directory in the / folder and the file system is attached to that directory. This is a special feature of the Integral PC and provides a much nicer means for a user to relate a piece of disc media to the general organization of the system's file system. In a standard UNIX system, it is left up to the user to decide where and under what name a file system on a disc is to be attached to the main file system. We did not wish to burden Integral PC users with such details.

Another special feature of the RAM disc is that all space for it is dynamically allocated from system memory as needed and then dynamically freed when it is no longer in use. This is the first RAM disc on a personal computer to manage its own space. On other personal computers that have a RAM disc capability, the space for the RAM disc is statically allocated and that portion of system RAM is lost to the RAM disc whether it is actually being used or not. On the Integral PC, unused RAM disc space is periodically freed for use by the rest of the system. One further advantage of the RAM disc is speed. Without the time-consuming mechanical head movement of a disc drive, RAM disc performance is exceptional.

A relative of the RAM disc is the ROM disc. This is an emulation of a disc device in ROM. It is mounted as a regular file system to the system root directory /. All applications stored in the system ROM are accessed via this disc device. These include PAM and several programs used in the automatic configuration of the system and the various devices that it controls. This arrangement allows a user to reference these ROM-based programs in the same manner as other programs on any other disc device.

Maintaining File System Integrity. File system integrity is an important issue with personal computers, especially with the common use of flexible disc drives. A typical user frequently moves discs in and out of these disc drives. This makes it quite difficult to manage the integrity of the file space on each piece of disc media. In a standard UNIX system, things are even more complicated. When writes occur in such a system, the data is not physically moved out to disc immediately. It is stored in an intermediate buffer and is only written out when the particular file buffer containing the information is needed for another disc operation. Although this enhances overall disc performance, it makes the file system extremely susceptible to corruption. Unintentional removal of a disc in typical UNIX systems can easily destroy the integrity of all information on the disc. This situation, of course, is completely unacceptable.

In the Integral PC system, all user data is immediately posted to disc. All file system information used to manage the file space on a disc is updated each time a file is closed. This greatly improves the security of the data on the file system while maintaining a low overhead of system disc activity managing the file system. The only time file information is at risk is when a file on a particular disc is open and then only the information in that file is in danger (a situation common to almost all personal computers and many minicomputers as well). Now, even if a flexible disc is accidentally removed from the Integral PC at the wrong moment, little damage is done. Also, any structural damage done to the file system can be easily repaired with a utility program provided with the Integral PC, called verify_disc. This is a friendly version of the fsck UNIX utility. All this puts greater control of file integrity in the user's hands and minimizes the accidental loss of information if a flexible disc is mistakenly removed at the wrong time. Maintaining the integrity of disc information in this manner greatly outweighs the slight file system performance loss caused by not buffering disc writes.

Basing a UNIX machine on a flexible disc drive presented a new challenge to the design team. Most UNIX systems are not friendly in dealing with removable discs. UNIX systems traditionally require the user to first put in a flexible disc, then type the mount command to make the operating system aware that a new disc is available to the operating system. The user is also required to type the unmount command before removing the flexible disc. This is necessary for systems that write information to buffers in RAM, and later write the information to the disc. Many operating systems flush any dirty buffers to the disc only when the unmount command is issued by the user. If a user forgets to issue the unmount command before pulling out the flexible disc, part of the information is left in the machine!

The Integral PC attacks this problem in two ways. First, as mentioned earlier, the operating system writes information directly to the disc. Information is not left in buffers to be written to the disc later. This allows the user to pull out the disc at any time. Second, the mounting and unmounting of discs is done automatically. A special process created at power-on issues the mount and unmount commands for the user. This process is called daemon. When a disc is inserted into the disc drive, the operating system senses it. The disc driver then alerts daemon, which then examines the disc. It senses if the disc is write protected, and what the volume name of the disc is. It then creates a folder for the disc, connects (mounts) the disc to the folder, and alerts the PAM process that a new disc is available. PAM then displays the new disc's files. When a disc is removed, the disc driver also alerts daemon, which then disconnects (unmounts) the disc from the folder, removes the disc from its list of mounted discs, and then deletes the folder. Finally, daemon notifies the PAM process that the disc has been removed and PAM updates its display.

Reference

1. S.R. Bourne, *The UNIX System*, Addison-Wesley, 1983.

A Friendly UNIX Operating System User Interface

by Jon A. Brewster, Karen S. Helt, and James N. Phillips

THE TERM USER INTERFACE is used to describe all interactions between a person and a computer. This includes keyboard layout, display formatting, command structures, and disc handling. The design of the user interface for HP's Integral Personal Computer was driven by the need to make the power of its HP-UX operating system (HP's version of the UNIX™ operating system) available to the novice. Some important constraints were to avoid alienating sophisticated UNIX users or previous HP personal computer users, to allow porting of standard UNIX software, and to allow a novice user to become more sophisticated in using the computer as the user's familiarity with the system grows.

The user interface for the Integral PC consists of three parts: HP Windows (window manager), PAM (Personal Applications Manager), and the inherent user interface of each application. The result is a visually oriented multitasking system that allows each program to run as if it were the only program running.

HP Windows

HP Windows provides a model for easy multitasking.The first part of the model is the desk and paper analogy. The screen can be considered a desk, with each window a paper lying on the desk. This is supported by the fact that windows can overlay each other without modifying each other's information (Fig. 1). Windows also can be shuffled from front to back, and moved around. A window supplies a view of part of a large two-dimensional object. This object can be moved up or down and left or right to change which part of the object is viewed through the window.

Each window has a name. The name, selected by PAM, is usually the name of the program that is running or the file that is being viewed. The name is contained in a small tab called the window title, which sticks up from the upper left edge of each window display. Since windows are allowed to cover each other, some rules for uncovering need to be discussed. In general, whenever a window receives new information, either from a running program or from echoing a user's input, that window shows itself in front of the other windows. A window is also shown when it is selected by the optional mouse for the Integral PC, when it is brought out of hiding (off-screen storage), or when it is brought to the front via the shuffle command (by pressing the shifted **Select** key on the keyboard).

Windows can be told to hide. When a window is hidden, the information part of the window is removed from the screen and the title is placed at the lower left part of the screen (see lower left corner of Fig. 1). When there is more than one window hidden, there will be a title for each hidden window. If an output occurs to a window in this state, the window remains hidden, but when the window is next shown, the new information will be there.

Shuffling a window to the front or back, or hiding it, does not change its location on the screen. Only its depth or visibility relative to the other windows is changed. This supports the paper/desk model by allowing specific papers to move to the top or bottom of a pile of papers on the desk.

The selector is a small image used with the **Select** key to select a place of interest on the screen. When the selector is between windows it appears as a small diamond. When the selector is over the menu it appears as a small arrow. Each window is allowed its own selector image which is in effect when the selector is over that window. The default image is a small arrow. The selector is usually controlled by a mouse, or other pointing device, but can be controlled by the keyboard.

The keyboard is considered to be connected to only one window at a time. When the user types, the letters show up only in this window, which is called the active window. A window is made active by the user in the following ways:

- The optional mouse can be used to make a window active. Simply move the selector over any portion of the desired window using the mouse, and press the left button on the mouse. This is also the method used to retrieve a hidden window from the lower left of the screen. Simply select the desired title.
- If a window is brought to the top with the shuffle command, it is made the active window.
- When an application is placed in a new window by PAM, it is made the active window.

Each window has its own cursor, which is a window-based image that is not allowed to leave its window. If the window type is an alpha window (most windows), the cursor behaves as a normal terminal cursor. If a window is a graphics window, the cursor is associated with the plotting pen.

The user interface is designed around a required keyboard and an optional mouse. The following two techniques are used to allow the user interface to behave in a consistent manner with and without a mouse:

- The keyboard can simulate a mouse movement. In particular, the cursor control keys can be made to control the selector by simply pressing the **CTRL** key with the cursor keys.
- Two keys on the keyboard, **Select** and **Menu**, duplicate the functionality of the two buttons on the mouse.

Menus. A menu is a list of items equivalent to the softkey labels of the Series 80 Computers and current HP 262X Terminals. These items are listed horizontally at the bottom of the screen in an inverse video area. This area "floats" on top of any windows that extend into the bottom area of the screen.

The **Menu** key on the keyboard and the right button on the mouse act as toggle switches, causing the menu to appear or disappear each time they are pressed.

Each window has its own menu called the user menu. This keeps menu items from different programs from interfering with each other. There is also a system menu for window manipulation commands that is shared by all windows. The **User** and **System** keys on the keyboard are used to select the menu type.

The softkeys at the top of the keyboard can be used to access the menu items. It is also possible to access the items by positioning the selector on the desired item with the mouse and then pressing the mouse's select button.

The center area of the menu contains the name of the active window and an identifier that tells the user which type of menu is active (see Fig. 1).

User Control. Window control can be very simple. PAM creates windows automatically. Unused windows get destroyed automatically whenever a new window is created. The only explicit control a novice user needs is the shuffle operation, which brings the window at the very back of the stack to the front, and makes it active. Consecutive shuffles allow a user to review all windows. Even windows that are completely hidden behind other windows can be accessed quickly.

The rest of the user control is performed through the system menu (see Fig. 2). An asterisk appears in the appropriate label block for one of the two states switched by toggle commands to indicate the current mode.

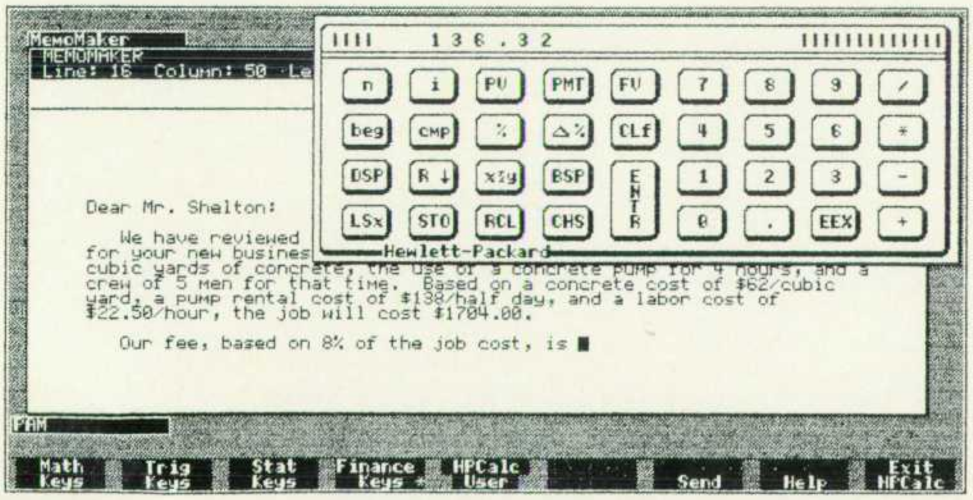
The Hide command removes the active window from the screen without modifying the information it contains. Its title, or banner, will be placed in the lower left corner of the screen.

The Invert command toggles the color sense of the active window. Black on orange will turn to orange on black, and vice versa. The Move command is an interactive command that will move the active window to a specified location without changing its size. When the move operation is invoked, the selector changes its image to that of a bracket in the shape of an upper left window corner (like an upside-down L). This corner selector indicates the placement of the upper left corner of the window. The user can move the selector anywhere on the screen, and press the **Select** key to finish the command.

The Pause command tells the active window to perform no more output until the next pause command is given. When a window is paused, the internal buffers fill up and the program is put to sleep. This is equivalent to **Ctrl-S** and **Ctrl-Q** in a standard UNIX tty driver.

The Save command toggles the save state of the active

(continued on page 31)



```
Line: 16  Column: 50  Le

Dear Mr. Shelton:

   We have reviewed
for your new busines
cubic yards of concrete, the use of a concrete pump for 4 hours, and a
crew of 5 men for that time.  Based on a concrete cost of $62/cubic
yard, a pump rental cost of $138/half day, and a labor cost of
$22.50/hour, the job will cost $1704.00.

   Our fee, based on 8% of the job cost, is █
```

Fig. 1. *Integral PC display showing calculator application window overlying document editor window. The PAM window is hidden.*

Personal Applications Manager

HP's Personal Applications Manager (PAM) is contained in the Integral PC's system ROM and is run automatically when the system is turned on. PAM provides facilities for managing programs and data. It is usually the user's primary interface to the Integral PC's HP-UX operating system.

The PAM user interface has three basic parts: the command area, the folder area, and the function keys. With this interface the user can see the contents of a folder, view the information in a file, and enter commands to run applications and manage files. A folder is a collection of programs, data files, and other folders; it is the same as an HP-UX directory (see Fig. 1).

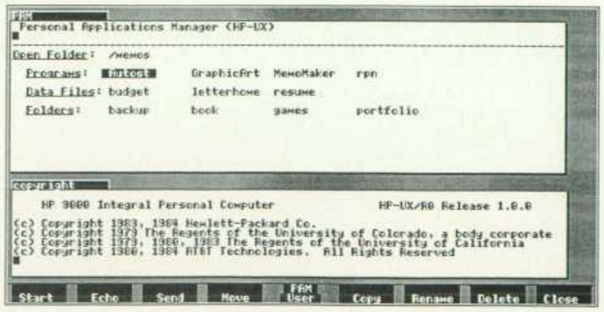
The command area, which is the upper portion of the PAM display, is where a user issues commands and receives PAM feedback. A command may be a function that is built into PAM (requiring no other program to be executed) or it may be a file in the file system. By building into PAM a carefully selected group of commands, the need for a mounted disc with command programs is eliminated in many cases. For an Integral PC without external disc drives, this has the benefit of freeing the built-in disc drive for a disc with the user's programs and data.

The criterion for selecting the commands built into PAM was to have a group of commands that provide for most of the user's general file handling needs. PAM has built-in commands to do the following:

- Copy, move, rename, and delete a file
- View the contents of a file
- Print the contents of a file
- Redisplay the contents of the open folder (the folder displayed in the PAM folder area)
- Change the open folder
- Make a new folder
- Type a file name or a line of text in another window.

Commands not built into PAM are associated with a file in the file system. This is a standard feature of HP-UX shells (command interpreters). With one of the standard HP-UX shells the command must be a program. PAM goes a step beyond this and allows a command to be a program, a data file, or a folder. If the command is a program, it is run. If it is a data file, the contents of the file are displayed one page at a time. If it is a folder, the folder is made the open folder. In addition to simple command entry, which may be adequate for many applications, PAM supports the following command entry features:

- Command parameters
- Specification of the standard input and output of a command
- Specification that the output of a command is to be used as the input of another command



```
PAM
Personal Applications Manager (HP-UX)

Open Folder:  /memos
  Programs:  [illegible]      GraphicArt  MemoMaker   rpn
  Data Files: budget          letterhome  resume
  Folders:    backup          book        games       portfolio

copyright
    HP 9000 Integral Personal Computer              HP-UX/R0 Release 1.0.0
(c) Copyright 1983, 1984 Hewlett-Packard Co.
(c) Copyright 1979 The Regents of the University of Colorado, a body corporate
(c) Copyright 1979, 1980, 1983 The Regents of the University of California
(c) Copyright 1980, 1984 AT&T Technologies.  All Rights Reserved
```

Fig. 1. *PAM window and menu display for Integral PC.*

- Starting a sequence of commands to be run in order
- Specification of the window in which a command is to be run.

Twenty lines of commands are saved by PAM and are accessible to the user for editing and reentry.

PAM always sets up a window for an external command to use for output. This keeps the PAM display and command output separate and allows PAM to be used while a command is running. If the user changes PAM's display font or inverts the PAM window, then command windows are set up with the new characteristics.

The folder area, which is the lower portion of the PAM display, displays the name of the open folder along with the contents of the open folder. The folder contents are displayed automatically and continuously. There is no need to run a command to display information as is the case with other HP-UX shells.

Files of the same type (program, data, folder or device) are grouped together in the display. One of the file names is highlighted—this is the file that is operated on by the function keys and that can be run as a command with a single keystroke or pick of the optional mouse. The file highlighted can be changed from the keyboard or by using the mouse.

When PAM is used to change the contents of a folder or to change to another folder, the folder area display is updated. If the size of the PAM window is changed, PAM reorganizes the folder area display so that it fits within the new bounds of the window. In the event that there is insufficient room to display the entire contents of the open folder, PAM allows the user to display the contents in several parts.

The PAM function keys are used to do one-keystroke execution of the most common built-in PAM commands. These keys operate on the file name that is highlighted in the folder area (this file name is the "parameter" for the function key commands). The labels of the keys and their use change based on the type of the highlighted file. For example, key **f1** is used to start a program when a program is highlighted, to view the contents of a file when a data file is highlighted, and to change the open folder and redisplay the folder area when a folder is highlighted. Functions that are not applicable to a particular type of file are not accessible with the keys when a file of that type is highlighted.

PAM is designed to be localized for use worldwide. PAM has been localized for several languages, including French, German, Italian, and Japanese (Katakana). Messages, key labels, and command names are all translated (using the full Roman8 character set), and the PAM display format is adjusted to meet localization needs. This is done by changing a message file that contains PAM's readable text; the same basic PAM program is used for all languages. PAM localization can be done by a nonprogrammer with an Integral PC in the countries where the localization is needed (i.e., no other development system is required).

Further Reading

1. S.W.Y. Wang and J.B. Lindberg, "HP-UX: Implementation of UNIX on the HP 9000 Series 500 Computer Systems," *Hewlett-Packard Journal*, Vol. 35, no. 3, March 1984.
2. P.S. Showman, et al, "Applications Software for the Touchscreen Personal Computer," *Hewlett-Packard Journal*, Vol. 35, no. 8, August 1984.

Brock Krizan
Development Engineer
Portable Computer Division

(continued from page 29)

window. This command is used to retain old windows that have desirable information. If a window is not to be saved (no asterisk in the label block and default mode) and no program is currently using the window (it is closed, but old output may still be present), then when the next window is created, the unsaved window will be destroyed.

The Stop command destroys the active window if it is not being used (not open). Otherwise, sigquit will be sent to the process group, and the window will be destroyed if and when the final close occurs.

The Stretch command changes the size of a window while leaving its position unchanged. The stretch operation behaves like the move operation except that the selector looks like the lower right corner of a window (a backward L). The selected position becomes the new lower right corner of the window, and the upper left corner (called the anchor) stays where it is. If the selected position is above or to the left of the anchor, then the window will be placed such that the selected position and the anchor form two corners diagonally opposed to each other.

The Alpha and Graphics commands bring up window-type-specific menus. The active window type determines whether Alpha, Graphics, or a future type is indicated in the system menu. These menus contain commands such as Display Functions for alpha windows and Pen Up for graphics windows.

Programmer Control. The programmer's interface for window control allows for complete control. However, all control is optional. The standard UNIX environment is set up mostly for terminals. The window environment is completely compatible with programs that use terminal abilities. In particular, the alpha window type is a pure superset of the normal UNIX tty driver. The technique for controlling windows is modeled after that used for controlling the tty driver. The program makes a call to get a copy of the current state, changes some of the information, and then makes another call to put it back. Some things that can be modified are name, location, and size. There are also bits that control whether the window is on the screen or hidden, has an inverted background color, is connected to the keyboard (is active), etc.

When a program is finished, it exits and goes away, but the window it was using is not destroyed. This is because many programs compute their answers, output them, and then immediately exit. If the window were destroyed, the information would be lost. Instead, the window is destroyed the next time a window is created (unless the user presses the Save softkey). If a program wants its window to be destroyed when it exits, the program should set the autodestroy bit.

Asynchronous events, such as alarms or telephone hangups, are handled by signals. The program merely specifies that a particular function be called when the event happens. Since many programs use the mouse, the Integral PC has a new signal called sigmouse. With this signal and a function that reports exactly what happened, a program can easily detect such things as the left button on the mouse going down or up, or the window being stretched or made active.

Window Architecture

The overriding constraint on the window manager architecture was the need to emulate existing UNIX I/O. It would not be acceptable to require a program to be modified to run it in a window. Because of this, the window system has been implemented as a set of device drivers.

For every window type there is a driver. The standard window manager includes the TERM0 alpha and HP-GL graphics drivers. There is a driver called the window control module, which coordinates all screen and keyboard activity. The drivers mentioned above are all independent of the system hardware. The hardware dependent software is contained in the keyboard driver and display driver. The intermodule connections are shown in Fig. 3.

The keyboard driver receives control from the interrupt handler. It processes its inputs into four types: keystrokes from keyboards, relative motion from devices such as a mouse, absolute coordinates from devices such as a graphics tablet, and ASCII codes from devices such as a bar code reader. Keystrokes are mapped through a RAM-based table into a standard set of 16-bit codes. This table can be modified to customize key layout. This is useful for supporting Dvořák keyboards and foreign language users. After the keyboard driver converts its inputs into a hardware independent form, it calls the window control module for further processing. The window control module processes some of the information (such as the shuffle operation and mouse movement) and passes the rest to the active window. However, the keyboard driver can buffer the keystrokes and make them available for direct reading. This is useful for a program that wishes to capture all keystrokes that enter the system, no matter which window or program would normally receive them. Keystrokes also can be written to the keyboard driver. These keystrokes are treated exactly like normal user input.

The display driver has four main jobs: character placement and font manipulation, line drawing, raster operations, and identification. At system power-up, the identification entry points are used to discover such things as screen size and default font size. Most of the software in this driver is associated with fonts and rasters. The main problem is that fonts and rasters are big, and the Integral PC's hardware resources are limited because of cost restraints. Data space for these objects is allocated dynamically from the main system RAM. However, to use these objects, they must be placed into a special area of RAM

Fig. 2. *View of system menu along lower portion of the display. An asterisk is displayed for one of the two states in label blocks for toggled modes. (Save and Pause in this example).*

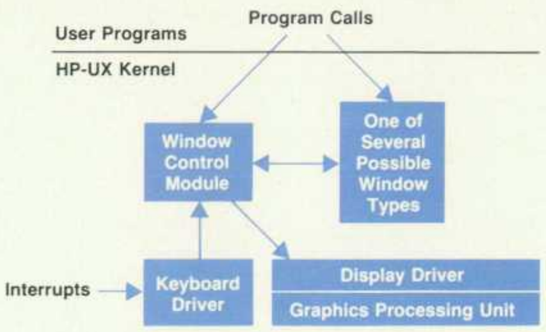


Fig. 3. *Block diagram of intermodule connections for HP Windows in the Integral PC.*

controlled by the screen hardware. This screen RAM is not large enough for all fonts or rasters that might exist in the system at any one time. Therefore, the allocated system RAM takes on the role of a backing store, requiring the design of the screen driver to address many of the classical disc driver problems such as thrashing and fragmentation.

The keyboard driver only calls the window control module as described above. All modules call the display driver. The display driver never calls other drivers. The window control module and the window drivers have a more complicated relationship. Each window driver must have 12 entry points for the window control module to call. Each window type also has at its disposal 10 entry points into the window control module. These entry points are designed to hide the nature of each window type from the window control module, and to centralize screen update and user interface strategies.

New window types may be added to the window manager. The writer of a new type must understand the use and maintenence of the shared data structure, the 22 cross-module entry points, and the abilities of the display driver.

Alpha Window

The main window type is the alpha window TERM0. This is the window that PAM creates for an application when it is started. The application's standard input, standard output, and standard error messages are connected to the new window. Since easy porting of applications was an important goal, the alpha window must look just like a terminal to the application. Applications written for the UNIX operating system expect to talk to a terminal through a serial interface. This serial interface is called the tty interface. The alpha window is a character driver with open, close, read, write, and ioctl entry points. These entry points provide the standard tty interface. The window is opened, read from, written to, and closed just as terminals connected over serial lines are. The ioctl entry is used to access other functions not provided by the read, write, open, and close entry points. The tty and ioctl system calls get and set the terminal's status. They operate just as physical terminals connected over serial lines do.

Since there is not actually a serial interface, some of the bits in the tty data structure are ignored by the alpha window code. However, the tty code will still get and set those bits. The alpha window actually shares the tty code with the serially connected devices. The tty status of the alpha window is initialized at window creation time to the state that most applications expect—9600 baud, signals enabled, line editing mode, echo on, destructive backspace, XON/XOFF pacing enabled, send the interrupt signal when the break key is pressed, map carriage return on input to a new line, and map a new line on output to a carriage return line feed.

Terminal Modes. The alpha window emulates a subset of the HP 2392 Terminal. This subset is known as TERM0. It has both an HP terminal mode and an ANSI terminal mode. The ANSI escape sequences supported are a subset of the ANSI X3.64 1979 standard. The ANSI mode allows access to additional software that was not written to use an HP terminal. The VT100 terminal is one of the more popular ANSI terminals. Thus, in ANSI mode, much of the software written for the VT100 could be used. However, the ANSI mode in the Integral PC is not a complete VT100 emulation.

The HP terminal mode recognizes HP escape sequences and does display enhancements in HP's field-oriented method. TERM0 is a character mode only. It does not contain the complex block and format modes of most of HP's terminals, since most of the UNIX software that is written does not use them. TERM0 does not contain memory lock either. It does contain all the editing functions such as delete line, insert line, clear line, delete character, etc. It also contains cursor positioning, rolling, status query, softkey definition, and display enhancement escape sequences. It has the enter line escape sequence to allow reading back from the terminal buffer, one line at a time. The HP terminal mode of the alpha window provides definable margins, tab stops, and soft character fonts.

Display enhancements in HP terminal mode are field oriented. When a display enhancement is set, it takes effect on all characters from the column the cursor is on to the right until the next display enhancement or the end of the line is reached. If the terminal is placed in transmit functions mode, the keys such as **Insert line**, **Delete line**, **Insert char**, and **Delete char** return the equivalent HP escape sequence instead of performing their function locally. The escape sequences are placed in the read buffer so that the application can read them.

The ANSI terminal mode contains functions similar to those for the HP mode. The ANSI terminal mode recognizes ANSI escape sequences and does stream-oriented display enhancements. It contains the equivalent editing functions such as delete line, insert line, delete character, etc. The editing ANSI escape sequences allow a parameter, so multiple lines can be deleted or inserted, multiple characters can be deleted, or multiple lines can be scrolled. The ANSI terminal mode also contains cursor positioning, scrolling, and display enhancement escape sequences. All characters sent after the display enhancement is set and before it is changed have that display enhancement regardless of where they are placed in the window. If the terminal is in transmit functions mode, the function keys return the ANSI escape sequence for the function.

Soft Fonts. Soft fonts are a useful feature of the alpha window. There can be eight different fonts in use at the same time. These fonts can all be in the same window or they can be used in several different windows. The fonts in an alpha window must all be the same size, but a different window can have a different-sized font. Thus, one window can be using a 6×8 font while a different window can be using 7×11 fonts. When the size of the font in a window is changed, the window is redrawn using the new font size. If the application has been set up to receive the stretch event signal from the window manager, a font size change will generate the stretch event signal. This allows the application to get the size of the font and the size of the window and then reformat its window to the best format.

The Integral PC comes with a collection of fonts, a command to change fonts in a window, and a font editor that allows the user to create new fonts. The font editor displays an enlarged version of the character being edited on the left and half of the 256 characters in a set on the right. First, the user selects the character to be edited. Then the user edits the character by toggling dots selected in the enlarged character. To see how the character appears surrounded by other characters, the user can type and a line will display the characters using the character set as it is at that moment. After the character set looks the way the user wants it, it can be saved in a file.

The font editor will edit fonts containing up to 256 characters. The character cell size can be up to 16 dots wide by 16 dots high. The font editor works either from the keyboard or with a mouse, although mouse operation is easier.

Fast Alpha. The alpha window also contains support for fast alpha, a high-performance functional interface for random-access alphanumeric devices. Fast alpha provides the capability of writing strings of characters with or without display enhancements at any position in the window, placing the cursor and turning it on or off, creating, activating, and removing fonts, filling a rectangle with a character with or without display enhancement, and scrolling a rectangle in any direction. The performance comes from sending many characters in one call, not having to look at every character to parse escape sequences, and not having to send an escape sequence to position the cursor before writing.

There are two fast alpha libraries. One is write only. This library is supported by various HP-UX machines. It contains the functions mentioned above. The second library allows information to be read back from the screen. This library is not a standard library and should be used only if the first library does not provide the functions needed.

HP-GL Graphics Window

During the early development of the Integral PC, we considered how screen graphics should be implemented. We realized that compatibility with other HP products was very important. Equally important was to make external software development as easy as possible by making it easy to port software to the Integral PC. We also knew that we would be supporting external HP-GL-based plotters.

Given the above considerations, we decided to model the graphics window after the HP 7470A Plotter.[1] Many software developers provide configurations for their applications that support the HP 7470A. While we decided that this was the best way to develop graphics for the Integral PC, it did present some problems. Foremost of these is the different graphics resolutions of the Integral PC's graphics window and the HP 7470A Plotter. The graphics window is limited to the screen size, which is 512 dots horizontally by 255 dots vertically, while the plotter's resolution is approximately 10,300 dots by 7650 dots. Therefore, plots displayed on the Integral PC may not appear as smooth as their plotter counterparts.

The HP-GL command implementation on the Integral PC is very close to that of the HP 7470A, but there are a few differences. UC, user-defined character, and VS, velocity select, were not implemented. Finally, we added one command, PG, to clear the graphics display.

Reference

1. M. Azmoon, "Development of a Low-Cost, High-Quality Graphics Plotter," *Hewlett-Packard Journal,* Vol. 33, no. 12, December 1982.

Data Communications

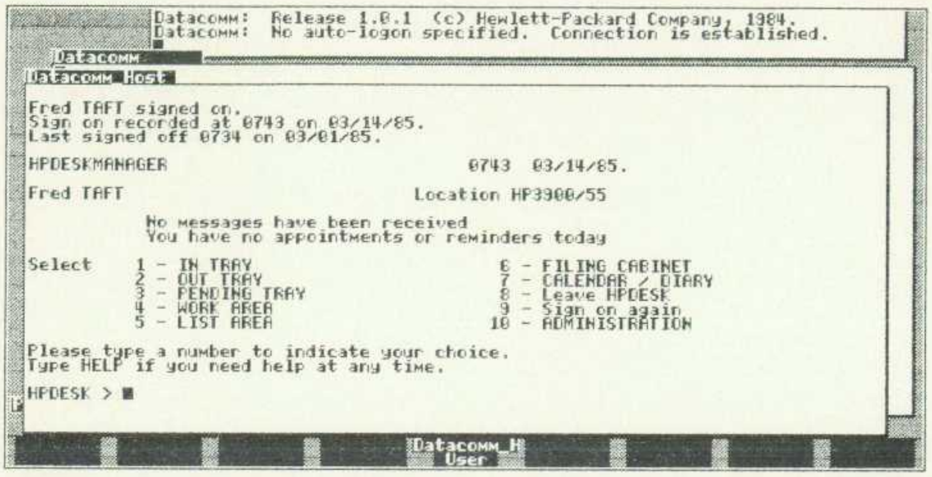
Most data communications programs in existence today were written without the use of windows in mind, primarily because windowed systems were not readily available. This tended to create a cluttered model for the user—the same window had to be used for communicating both with a remote system and with the local terminal emulator program. To change from one mode to the other, the user was required to type a cryptic escape code sequence. A second disadvantage of this method was that while one mode was active, the other was disabled. With the advent of a multiple-window system, both of these problems are easily resolved and a cleaner model is presented to the user.

The data communications package for the Integral PC provides the user with three windows: a host window (see Fig. 1), a command window, and a message window. When a remote connection is active, the host window behaves as the terminal; anything the user types into this window will be transmitted, while all data received from the remote system is displayed here. When the command window is active, everything typed by the user is interpreted locally as a command intended for the datacom program. The message window is used solely for displaying status and error messages.

Whenever the user is using one of the datacom windows, the other two windows continue to perform their assigned tasks. For example, if the user has selected the command window, and is currently viewing a help screen, the host window could be performing a file transfer at the same time. By simply using the optional mouse, or any of the other means provided by the Integral PC for selecting a new window, the user can move quickly from one of the datacom windows to another. No complex user interaction is required.

Terminal Emulation

The Integral PC data communications package emulates a basic HP 2622 Terminal without block or format modes. The terminal emulation functions (processing escape sequences, handling keyboard input, and display management) are all handled by the window manager and the TERM0 window type. What the datacom package provides is a link between the optional



```
Datacomm:  Release 1.0.1  (c) Hewlett-Packard Company, 1984.
Datacomm:  No auto-logon specified.  Connection is established.
```

```
Fred TAFT signed on.
Sign on recorded at 0743 on 03/14/85.
Last signed off 0734 on 03/01/85.

HPDESKMANAGER                              0743  03/14/85.

Fred TAFT                              Location HP3000/55

           No messages have been received
           You have no appointments or reminders today

Select    1 - IN TRAY                         6 - FILING CABINET
          2 - OUT TRAY                        7 - CALENDAR / DIARY
          3 - PENDING TRAY                    8 - Leave HPDESK
          4 - WORK AREA                       9 - Sign on again
          5 - LIST AREA                      10 - ADMINISTRATION

Please type a number to indicate your choice.
Type HELP if you need help at any time.

HPDESK > ■
```

Fig. 1. *Display of host window for Integral PC's datacom package. Portions of the command and message windows can be seen at the top.*

RS-232-C I/O card for the Integral PC and a TERM0 window. Along with providing this link, the datacom package also supplies a mechanism for performing several types of file transfers, configuring the RS-232-C card, and doing a variety of other related tasks.

Many of the commands provided by the datacom package specify configuration options. When the user chooses these options, a form is displayed. Then, using any combination of the mouse, softkeys, cursor keys, and editing keys, the user enters the desired parameters into the form. When the form has been completed, it is removed from the display, and the parameters are remembered until the user changes them.

File Transfers

The Integral PC's datacom package is capable of performing two different types of file transfers:

- Character transfers, where the file is transmitted/received as a string of ASCII characters, and no checking is performed on the data.
- Block protocol transfers, where the file is broken up into blocks and transmitted or received as such. Each block requires an explicit acknowledgment from the receiver before the next block is sent. The block protocol used is the Christensen umodem (or xmodem) protocol.

Each file transfer technique has its advantages and disadvantages. A block protocol transfer is a necessity if the integrity of the file must be maintained, or if the file contains binary information. The disadvantage of this type of file transfer is that it involves more overhead than a character transfer, because each block must be acknowledged before the next will be sent. However, since file integrity is normally of primary importance, the performance penalty is negligible. Character transfers must be used to exchange ASCII files with another machine that does not support the block protocol. Although this type of transfer is faster, it cannot be used to transfer binary files. At the present time, a wide base of machines support the block protocol, including HP's Portable Computer and the HP 3000. Most machines available today are also capable of supporting this type of transfer.

A frequent use for the file transfer facility is for data logging. The user has the option of either keeping a copy of a session in a file, or logging the session on the printer.

Fred Taft
Development Engineer
Portable Computer Division

Printer and Plotter Drivers

There are many different types of printers and plotters on the market today. They are specialized to perform specific functions (e.g., slower letter-quality printers and faster dot matrix printers). Often, a user owns more than one printer or plotter and desires that all of the peripherals be available at the same time.

The UNIX™ operating system traditionally does not recognize this situation. For example, the standard printer is usually /dev/lp and UNIX applications using a printer use /dev/lp. This does not work well with more than one printer. Either the printers must be physically swapped between applications or the second printer must be assigned a different device name.

An ideal solution would be to have all peripherals available at all times and have all peripherals of a given type (i.e., all printers) accessible through the same device name. The solution developed for the Integral PC meets these goals.

Four new variables, printer, plotter, print_flag, and plot_flag, exist in the data structure that describes each process (called the proc structure). The printer and plotter variables contain the device numbers (both major and minor) describing the current printer and plotter, respectively. The print_flag and plot_flag variables contain bit flags used to control the operation of printers and plotters. Because a separate proc structure exists for each process, each process may use a different printer or plotter. Another advantage of this approach is that each process inherits the proc structure from its parent process. This means that when the user sets a specific printer for use by PAM, any applications started by PAM will also use the same printer. The user can subsequently change

PAM's printer without affecting the application already started.

This brings us to the indirect device drivers /dev/lp and /dev/plotter, which operate in a similar manner. When data is sent to /dev/lp, the driver looks at the proc structure to determine which printer to use.

The ability to change printers and plotters is provided via the printer_is and plotter_is commands, which allow the user to specify not only the device to use, but also the baud rate, parity, and handshake method for RS-232-C devices. The plotter_is command also can create graphics windows when that is requested.

Authors

October 1985

4 PC System

Nelson A. Mills

At HP since 1976, Nelson Mills worked on the HP-85 Personal Computer operating system and interpreter. He was project manager for HP-87 and HP-86 PC firmware development and was the section manager in charge of software for the HP Integral PC. After receiving a BS in mathematics from Albion College, Michigan in 1961, Nelson spent five years in the U.S. Navy and then worked for ten years as a systems programmer. He lives in Corvallis, Oregon with his wife and two children. Outside of working hours Nelson enjoys photography, camping, and coaching youth basketball.

Tim J. Williams

Tim Williams started at HP in 1975 as a circuit design engineer on the HP-85 Personal Computer. He has also served as an R&D project manager and section manager for low-end technical computers and was recently appointed R&D department manager for the Handheld Computer and Calculator Operation in Corvallis, Oregon. Tim was born in Camp Chaffee, Arkansas and attended Oklahoma State University (BSEE 1973) and Purdue University (MSEE 1974). He's married and has two children. His hobbies include gardening and fishing, hiking, and skiing in Oregon's beautiful Willamette Valley and Cascade Mountains.

6 Electronics System

James A. Espeland

Jim Espeland was born in Glasgow, Montana and studied electrical engineering at Montana State University, receiving his BS degree in 1981. He has been an R&D engineer at HP since 1981 and contributed to the development of the HP Integral PC. He is currently working on a computer science degree at Oregon State University. Jim lives with his wife and daughter in Corvallis, Oregon. During his leisure time he plays basketball, volleyball, and softball and enjoys photography, videography, and woodworking.

David L. Kepler

A project manager at HP's Corvallis Workstation Operation, Dave Kepler has been with the company since 1980. He contributed to the development of the HP Integral PC as well as providing manufacturing support. He also provided engineering support for HP-85 Personal Computer manufacturing. Born in Wilmington, Delaware, Dave received a BSEE degree from the University of California at Santa Barbara in 1980. He lives in Corvallis, Oregon. During his leisure time he manages his personal investments and enjoys softball, fishing, and golf as well as other outdoor activities.

10 Graphics Processor Unit

Dean M. Heath

Born in Stockton, California, Dean Heath served in the U.S. Coast Guard for five years before earning a BSEE degree from Oregon State University in 1975. Since joining HP in 1977, he has been responsible for IC design for several products, including HP E-Series calculators. He designed the graphics processor for the HP Integral PC and also completed work for an MSEE degree from Oregon State University in 1984. Dean, his wife, and two boys are residents of Corvallis, Oregon. He sings in his church choir, makes stained glass, and is interested in robotics. He also likes camping and on- and off-road motorcycling.

12 Electroluminescent Display

Marvin L. Higgins

Marv Higgins was born in Utah and studied electrical engineering at both the University of Utah (BSEE 1967) and Stanford University (MSEE 1970). At HP since 1967, he has had a wide variety of experience in R&D and manufacturing, including the startup of calculator manufacturing operations in the U.S. and Singapore. He was the project leader for the display on the HP Integral PC and has also worked on color displays. His work on electroluminescent displays has resulted in two patent applications and several published articles. Marv lives in Corvallis, Oregon, is married, and has two sons. He and his family manage a farm on which they raise cattle, sheep, and other animals. He likes music and a variety of outdoor activities.

18 Mechanical Design

Thomas A. Pearo

The mechanical designer of the HP Integral PC, Tom Pearo has been at HP since 1973. His earlier contributions include product design for the HP-86 Personal Computer and design of assembly equipment for a number of calculators, including the HP 41C. He was born in Philadelphia, Pennsylvania, served in the U.S. Navy, and completed work for a BSME degree from the University of California at Davis in 1973. Tom is married, has two sons, and is a member of a Benton County, Oregon citizens group. He likes to sail, read, and tinker with mechanical things.

22 UNIX Operating System

James R. Andreas

At HP since 1980, Jim Andreas designed operating system software for the HP Integral PC and worked on the HP 82905A Printer and the HP 82949A Printer Interface. Before coming to HP he designed hardware and software for digital plotters and says he is interested in hardware/software trade-offs. He was born in Silverton, Oregon, earned a BS degree from the Massachusetts Institute of Technology in 1977, and is working toward a master's degree at Oregon State University. He lives with his wife and young daughter in Corvallis, Oregon and is designing a Japanese-style garden at his home. He also collects Japanese maple trees.

Andrew L. Rood

Andy Rood has been developing personal computer products at HP since 1980. He worked on mass storage for the HP Series 80 Personal Computers and was the project leader for the operating system on the HP Integral PC. Born in San Francisco, California, he earned a BS degree in mathematics from Stanford University in 1975 and an MS degree from Oregon State University in 1978. His professional interests include operating systems and artificial intelligence and he is the author of several computer science papers. He lives in Corvallis, Oregon and lists chess, camping, robotics, golf, and writing as his leisure activities.

Robert C. Cline

Bob Cline was born in Oceanside, California and grew up in Indianapolis, Indiana. He studied mathematics at the University of Massachusetts (BS 1976) and computer science at Indiana University at Bloomington (MS 1978). With HP since 1978, he has worked on distributed systems software for the HP 3000 and on the file system for the HP Integral PC. He is interested in operating systems and networks. Bob lives in Corvallis, Oregon and likes hiking, golf, photography, camping, and music.

Ray M. Fajardo

Born in Chicago, Illinois, Ray Fajardo earned a BS degree in mathematics from Oregon State University in 1969 and came to HP the same year. He has worked on HP 1000 Computer automatic test systems and operating systems and was a section manager for the development of HP 1000 system software. He was also the project manager for HP Integral PC system software. Ray and his wife and two children are residents of Corvallis, Oregon. He likes hiking and fishing and is interested in Indian artifacts. He also plays the guitar and banjo.

28 User Interface

Karen S. Helt

A 1980 graduate of Oregon State University, Karen Helt holds a BSEE degree. She joined HP in 1980, worked on a keyboard controller, and later developed the window manager for the HP Integral PC. She is working on a master's degree in computer science at Oregon State University and is an advisor for the school's chapter of the Tau Beta Pi honorary society. A resident of Albany, Oregon, Karen is married and enjoys reading science fiction and fantasy. She also plays Dungeons and Dragons and is learning to make stained glass.

James N. Phillips

With HP since 1981, Jay Phillips was responsible for the graphics windows for the HP Integral PC. Earlier, he developed a CP/M® card and a terminal emulator for the Series 80 Computer as well as other software. Jay was born in Anniston, Alabama, served in the U.S. Navy, and completed work for a BS degree in electrical and computer engineering from Oregon State University in 1978. Before coming to HP he worked on digital electronic design at Raytheon Company. He and his wife and two children live in Corvallis, Oregon where he is a member of the Jaycees. He is involved in real estate investing and enjoys mountain climbing and bicycling.

Jon A. Brewster

Jon Brewster has been working on calculator and personal computer products since joining HP in 1977. He has provided technical customer support and has been an applications engineer. He was both a project leader and project manager for the HP Integral PC and was responsible for the user interface and the utilities disc set. Jon was born in Seattle, Washington and received a BS degree from Oregon State University in 1980. He lives in Corvallis, Oregon with his wife and three sons and enjoys photography, motorcycles, and astronomy. He comments that the Integral PC makes a fine telescope controller and that he has created a nice control library.

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

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

HEWLETT-PACKARD JOURNAL
October 1985 Volume 36 • Number 10

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

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.

5953-8540