MAY 1982

HEWLETT-PACKARD JOURNAL

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

Contents:

MAY 1982 Volume 33 • Number 5

In this Issue:

The HP desktop computer on the cover of this issue is showing you its engineering graphics capability by displaying a diagram of a system that its owner might be designing, or that it might be controlling. The computer, Model 9836A, and its smaller-screen, single-flexible-disc-drive cousin, Model 9826A, are the subjects of this issue. These two new desktop computers are state-of-the-art descendants of the HP 9825 Computer/Controller, which has been HP's top system controller since 1976. Over 28,000 customers now own 9825s.

People like desktop computers because they're friendly and dedicated, and these two new ones are no exceptions. But we've learned a lot since 1976. Using the latest technology, these new computers are up to five times faster than the 9825. They can speak three programming languages—HPL, BASIC, and Pascal—instead of just HPL. And they have much larger memories—up to two million bytes, or characters. They're also reliable, having been designed that way and strife-tested to assure a low failure rate.

Both of these powerful machines make excellent controllers for automatic measurement, test, and control systems. The 9836A, with its larger screen, is especially good for computer-aided engineering, which means helping engineers design and develop new products or systems. The computer can do simulation, prototype testing, production of engineering drawings and printed circuit board layouts, software development, and report generation.

An unusual feature of the new computers is the knob, a rotary control on the keyboard. When editing programs, the user can spin the knob to move the machine's attention rapidly to the item of interest. In computer-aided engineering the knob can be used to change a parameter or component value while the computer simulates how the product's operation will change as a result. In system control the knob can be used for many things, such as varying the speed of a motor.

The article on page 3 will introduce you to the 9826A and 9836A Computers. Articles about their hardware and software designs, how they interface with peripheral devices and instruments, and how they can communicate with other computers are on pages 7, 17, 24, and 33. A new thermal printer family designed to work with them is described on page 22.

-R. P. Dolan

Editor, Richard P. Dolan • Associate Editor, Kenneth A. Shaw • Art Director, Photographer, Arvid A. Danielson • Illustrator, Nancy S. Vanderbloom
Administrative Services, Typography, Anne S. LoPresti, Susan E. Wright • European Production Supervisor, Henk Van Lammeren

Advanced Multilingual Computer Systems for Measurement Automation and Computer-Aided Engineering Applications

Developing and running a test, measurement and control, or computer-aided engineering system is much easier if you have the right tool. These computer systems are designed specifically for such use.

by John L. Bidwell and David W. Palermo

DESKTOP COMPUTERS have provided scientists, engineers, and other noncomputer professionals with the ability to solve computation and instrument control problems in a timely and relatively painless way. The new HP Model 9826A and 9836A Computer Systems (Fig. 1) are fourth-generation desktop computers that blend traditional desktop computer friendliness, powerful computer features, and a state-of-the-art microprocessor with a choice of three programming languages. All of these ingredients take the best of the evolution of desktop computers and add the price/performance advantages of the latest technology.

The 9826A has a 178-mm-diagonal CRT (cathode ray tube) display and a built-in 5.25-in, 264K-byte flexible disc drive. The 9836A has a 310-mm-diagonal CRT display and two built-in 5.25-in, 264K-byte flexible disc drives. At the

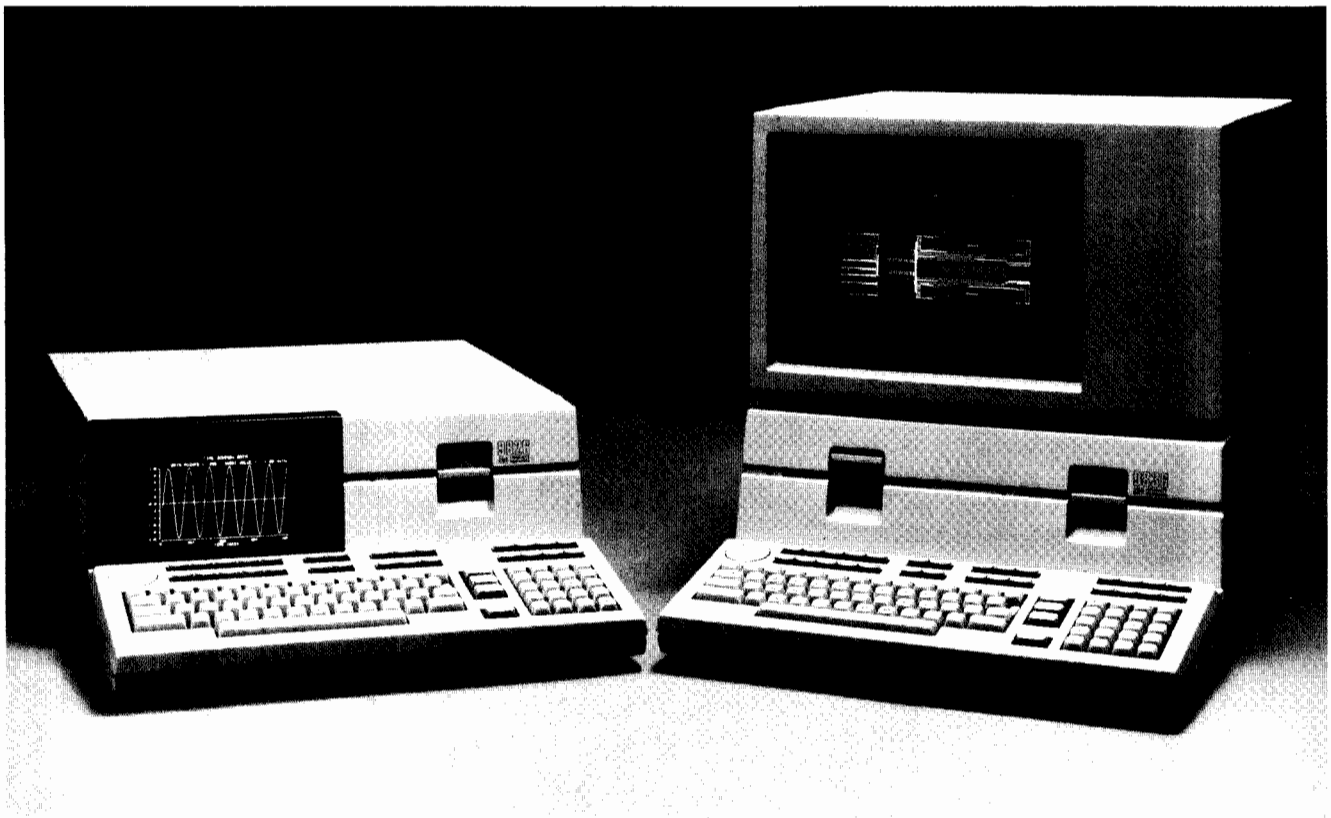


Fig. 1. *The HP Model 9826A Computer System (left) is a multilingual high-performance instrument controller with built-in display and mass storage capability. The HP Model 9836A Computer System (right) is a large-screen-display version of the 9826A with added internal mass storage capability. It supports the same languages as the 9826A and is designed for applications requiring additional graphics and computer-aided design features.*

9826A versus 9836A

The 9826A is designed to be the premier controller of the 1980s, replacing the 9825, the past standard of performance in instrument control. The 9826A is rackable and/or stackable, designed to be visually compatible with other HP rack-mount instruments, and has a high degree of internal peripheral integration within a package that is only 18 cm high. The built-in HP-IB interface, clock, knob, and graphics are all capabilities needed by most instrument control systems.

The 9836A is a large-screen version of the 9826A with two internal flexible disc drives. Computer-aided engineering is the target market for this new machine although it is equally capable as an instrument controller. The powerful architecture of the 9826A also serves the 9836A while the larger screen and second disc drive facilitate programming, graphics display and mass storage operations involving more than one medium. The differences between the 9826A and the 9836A are limited to careful changes in the display and mass storage subsystems. All of the other hardware and software features are the same for both systems.

In adding a second disc drive to the 9836A the primary goal was to maintain high-speed access, especially in the case of disc-to-disc transfers. Two design paths were possible. The existing bus between the disc controller board and the first disc drive could have been extended to the second drive in a daisy chain fashion. With this configuration, two problems would be encountered. First, monitoring media changes on both drives would not be possible. Second, when the drive select is changed a delay is incurred in waiting for the selected drive to become ready.

Instead of daisy chaining, a second drive cable was added. This permits separate monitoring of media changes and readiness for the drives, with the rest of the controller logic being shared. With this design, media changes can be monitored by software and the drive select can be changed without waiting.

-Steve Chorak
-Jon Rubinstein

heart of both computers is a 68000 microprocessor chip interfaced to a 100-pin memory and I/O (input/output) bus. Also standard is a keyboard with ten (20 with shift) user-definable softkeys, a built-in HP-IB* interface, a real-time clock, a programmable beeper, and a graphics display system (400×300 pixels for the 9826A, 512×390 pixels for the 9836A). In addition to the standard ASCII** character set, keycap labels and character sets are available for French, German, Katakana (Japanese), Spanish, and Swedish-Finnish.

A knob on the upper left-hand corner of the keyboard provides both programmers and operators an input device with an analog feel. When editing programs, this device is connected internally to the CRT display's cursor for scrolling program lines or moving the cursor to a particular character within a line. Information about the knob (direction and amount of rotation) can be obtained by program statements and used to control instruments (e.g., gang-tuning a synthesizer/analyzer combination) or computations (e.g., adjusting a component value in an active circuit while continuously plotting the frequency response).

*Hewlett-Packard Interface Bus, HP's implementation of IEEE standard 488 (1978).
**American Standard Code for Information Interchange.

Programming Languages

The 9826A and 9836A Computer Systems can be used with any of three programming languages—BASIC, HPL, and Pascal (see article on page 24).

BASIC provides an enhanced capability set to serve a variety of customers from first-time users who appreciate simple BASIC constructs and friendly program development to advanced programmers who appreciate fast, sophisticated functions and subprograms and the unified I/O and mass storage. Unified I/O means that all I/O operations appear to be the same to a programmer and can be easily redirected to different devices (e.g., to a printer or a disc file; from an instrument, keyboard, or disc file).

HPL is a very compact and efficient algebraic language first used by Hewlett-Packard on the 9820 and later on the 9825 Desktop Computers. The version used in the 9826A and 9836A provides a high degree of compatibility with existing 9825 programs, files, and interface capabilities, a significant improvement in execution speed, and several new HPL features such as a full-screen editor, CRT graphics, real-time clock, and access to the knob functions.

Pascal is a highly structured, modular programming language that provides compiled speed and program development contributions not found in other Pascal implementations. Pascal programs are directly compiled to 68000 link form and are automatically linked by a librarian routine which makes it easy to put together sophisticated, modular systems.

The 9826A and 9836A offer users a flexible set of programming language choices: each language is capable of running on the mainframe and it is possible to order another language as an add-on capability at any time. To provide more user flexibility, some of the languages are available "hard" in ROM (read-only memory) as well as "soft" to be loaded into RAM (random-access memory). For instance, users who intend to write most of their programs in BASIC but occasionally use HPL are able to purchase hard BASIC and soft HPL. The wakeup configuration of the system is determined at power-up by a small ROM (called the boot ROM). After initializing the computer and performing memory tests, the boot ROM determines which language system to wake up. In the 9826A and 9836A, soft language systems such as Pascal have precedence over hard systems, so the first item of business is to check for a disc in the flexible disc drive. If a disc is installed, a front-to-back search of the directory is made to see if there is a system file. If one exists and has a file name beginning with SYSTEM_

```
10    INTEGER L   ! Not possible in HPL
20    Start_time=TIMEDATE
30    K=0
40    DIM M(5)
50    K=K+1
60      A=K/2*3+4-5
70      GOSUB 150
80      FOR L=1 to 5
90        M(L)=A
100     NEXT L
110   IF K<1000 THEN 50
120   Stop_time=TIMEDATE
130   PRINT "Execution time =";Stop_time-Start_time;" seconds"
140   STOP
150   RETURN
160   END
```

Fig. 2. *Program listing in BASIC of benchmark program used for obtaining data given in Table I.*

Table I
Benchmark Performance
Total Execution Time (in seconds):

9825	9826A/9836A	9826A/9836A	9826A/9836A
HPL	HPL	BASIC	Pascal
----------	Interpreted	----------	(Compiled)
11.48	6.95	3.52	0.91

NOTE: Each language used the "best" features available. BASIC and Pascal are capable of running this program using integer math for variable L (the FOR/NEXT loop counter); these are the times shown. HPL used multiple statements per line wherever possible. Two other numbers for BASIC have been run: 4.01 seconds using a 64-bit floating-point real variable for the loop counter, and 2.59 seconds using integer scalar variables (A,K, and L) and integer math wherever possible (including integer divide (DIV) instead of floating-point divide(/) in line 60).

this soft system is loaded into RAM and given control. If no system files are found on the disc (or no disc is installed), the boot ROM searches from low memory towards high memory looking for the ROM header of a language system. If it finds one, it continues searching (on 16K-byte boundaries) to see if there is more than one hard system. If only one hard system is found, this system is given control and the computer wakes up in that language. If more than one hard language system is found, the user is given a choice of which one to turn on. For example, if BASIC and HPL are both installed in ROM, the user responds to a prompt with B for BASIC or H for HPL. If no hard or soft language system is found a message is displayed indicating this fact; the user is told to push **RESET (SHIFT PAUSE)** to try again.

Besides this procedure, the boot ROM also interrogates the hardware and sets up a hardware-attribute table for use by the language systems. This table defines the alphanumeric screen width, the graphics size, and the internal disc configuration corresponding to whether the product is a 9826A or 9836A. This provides the language systems with the necessary information to configure themselves for either of these two products.

In addition to friendliness, excellent I/O, and graphics, fast execution speed was one of the goals for each language implementation. Table I shows the total execution time for each 9826A/9836A language configuration (as well as the 9825) when running a simple benchmark program.[1] (We have only shown the listing of the program in BASIC in Fig. 2. However, it represents the algorithm used by all of the languages).

Versatile Backplane

The backplane of the 9826A and 9836A has eight user-accessible slots. Up to four (every other one) of these slots can be used for any of the many high-performance interface cards that are available (see Fig. 3). The remaining four slots can be used to install RAM, ROM, and/or DMA (direct memory access) cards.

The interface cards (see article on page 17) include the HP-IB card which is essentially identical to the built-in HP-IB port and can be used in conjunction with a dual-port DMA (direct memory access) card to provide data transfer speeds of 200K bytes per second and a complete IEEE-488 implementation. The 16-bit parallel GPIO card provides all functions implemented on its predecessors and supports DMA rates up to 1.5 megabytes per second. A high-speed serial card with on-card buffering and a 19,200-baud transfer rate handles many asynchronous protocols and the Hewlett-Packard Distributed Systems Network Datalink (DSN/DL) protocol. Also available are an RS-232-C card

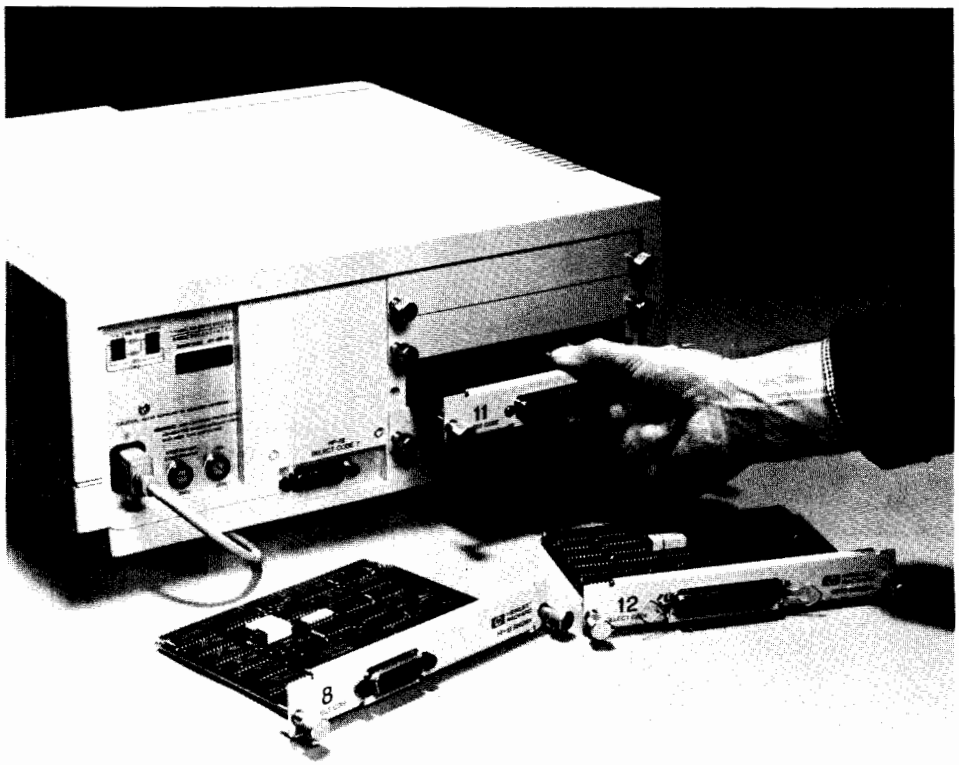


Fig. 3. *The backplane of the 9826A and 9836A mainframes accommodates up to four I/O interfaces and four RAM, ROM, or DMA cards in addition to the internal HP-IB.*

designed to interface terminals and printers, and a BCD (binary-coded decimal) card for reading information from bit-parallel BCD instruments.

I/O Performance

Important operations for controllers include the ability to do short I/O transactions quickly with a minimum of overhead (such as outputting a short alphanumeric string to select a channel on a relay scanner, or entering a reading from a voltmeter). One of the tests made to determine the effectiveness of the implementations was the Nelson Benchmark (see Fig. 4) named after our lab manager Jerry Nelson, one of the inventors of the HP-IB, who devised the experiment. The equipment consists of an HP 3437A Systems Voltmeter, a 3495A Relay Scanner, and a 9826A Computer System. The voltmeter/scanner combination measures the voltage at each node of a biased resistor stick (10 resistors of equal value connected in series). The execution time required to determine expected values, measure the network, and check each measurement against a limit is listed in Table II for each language configuration.

Table II
Nelson Benchmark Performance
Execution Time (in seconds):

9825	9826A/9836A	9826A/9836A	9826A/9836A
HPL	HPL	BASIC	Pascal
	Interpreted		(Compiled)
0.113	0.092	0.077	0.038

NOTE: The timed portion (Fig. 4) is the call to "Get readings" and the loop contained in lines 130 through 170. The time for compiled Pascal includes some hand-tuning of the measurement calls and a specially written routine to convert the DVM readings into integer form so integer math could be used. This is a good example of what is possible in Pascal when used by someone who understands the application and can tailor the program accordingly. Using the standard I/O library, the execution time for Pascal was 0.11 seconds.

Many Peripherals

The 9826A and 9836A have excellent capabilities to control the many hundreds of IEEE-488 (HP-IB) instruments. Also, several important peripherals are fully supported (including full system testing). In the hardcopy area, the systems can use the HP 9866B, 9876A, 2671A/G and 2673A printers. The new HP 2670 Series Printer family is designed to sit on top of the mainframe (see article on page 22). In addition to raster-dot graphics dump supported on the 9876A, 2671G and 2673A printers, high-quality vector plotting can be accomplished on the 7225A and 9872B/C X-Y plotters, and graphical input is supported from the 9111A Graphics Tablet. Current disc support includes the HP 82901 (dual drive) and 82902 (single drive) 5.25-in flexible disc drives, and the 9895A 8-in flexible disc drive (for both single-sided and double-sided discs).

```
100 !
110     Start_time=TIMEDATE
120 !
130     Get_readings(Voltage(*))
140     FOR Node=1 TO 10
150       Delta=ABS(Voltage(Node)-Std_value(Node))
160       IF Delta>Tolerance*Std_value(Node) THEN GOSUB Error_message
170     NEXT Node
180 !
190     Stop_time=TIMEDATE
200 !
210     PRINT "TOTAL TIME =";Stop_time-Start_time;" SECONDS"
220     END
230   SUB Get_readings(Data(*))
240     COM /Inst_ident/ @Voltmeter,@Scanner
250     INTEGER I
260     FOR I=1 TO 10
270       OUTPUT @Scanner USING "ZZ";I
280       OUTPUT @Voltmeter;"T3";
290       ENTER @Voltmeter;Data(I)
300     NEXT I
310   SUBEND
```

Fig. 4. *Partial listing of the Nelson Benchmark program used to obtain data given in Table II.*

Applications and Utilities

Many application programs are available for these computers and many user programs can be adapted from other computers. BASIC utilities are available to do disc backup, cross reference, and secure a program from listing. A training disc demonstrates the machine's capabilities. Programs to translate from other HP implementations to 9826A/9836A BASIC are available. Applications for BASIC include a complete statistics package, a program management package, and VisiCalc™.

VisiCalc is a trademark of VisiCorp.

John L. Bidwell

John Bidwell joined HP in 1970 and has worked on various aspects of the 9830, 9835, 9845 and 9826 designs. He was project manager of the interpreter group for 9826A BASIC. John received the BS degree in mechanical engineering from the University of New Mexico in 1966 and an MS degree in the same subject from Southern Methodist University in 1970. He is named co-inventor on one patent related to the 9845 Computer. A native of New York City, John and his seven-year-old son live in Fort Collins, Colorado. John enjoys bicycling, motorcycle racing, sailing, and playing tennis.

David W. Palermo

David Palermo is a native of Mora, Minnesota. He attended the University of Minnesota, earning the BEE degree in 1967 and the MSEE degree in 1968. With HP since 1969, Dave has worked on computer interfaces and voltmeters. He was the project manager for BASIC I/O and mass memory on the 9826A and the BASIC conversion to the 9836A. Dave is now an interface engineer for HP's Instrument Group. He is the author of a paper on self-test and one of the co-inventors of the HP-IL (Hewlett-Packard Interface Loop). Dave is married, has two children, and lives in Los Altos, California. His outside interests include directing church youth choirs, listening to music, and programming computers.

HPL has several utilities, including disc backup. Pascal has utilities as part of the system library to do HP-IB instrument I/O, other interface I/O, a subset of DGL graphics as used on the HP 1000 Computer, and mass memory management.

Acknowledgments

The development of the 9826A, and later the 9836A, has been a large major program for HP's Desktop Computer Division (DCD) over the last few years. Nearly everyone in DCD has participated in some manner. The leadership of numerous managers and the exceptional performance of many others made these products possible. It was truly a team accomplishment of which we are proud.

Reference

1. T. Rugg and P. Feldman, "BASIC Timing Comparisons.... Information for Speed Freaks," Kilobaud, the Small Computer Magazine, Issue 6, June 1977.

Hardware Design for an Integrated Instrumentation Computer System

by Don D. Stewart, Robert J. Horning, Ken L. Burgess, Ronald G. Rogers, and James W. McLucas

THE DESIGN GOAL for the HP Model 9826A Computer System was to replace the 9825 Desktop Computer with a product having twice the performance at about the same price. This meant not only better than two times the speed of the 9825, but also better than two times the overall functionality, friendliness, and reliability. The 9826A has these features, can be operated with several software languages, and is contained in a convenient package that can be rack mounted or stacked with other instruments.

A block diagram of the 9826A hardware is shown in Fig. 1. The hardware for the 9836A is similar, differing only in having a larger CRT and another flexible disc drive. The CPU used by both systems is the 16-bit 68000 microprocessor. The following attributes made this part an attractive choice:

- Versatile 16-bit-external/32-bit-internal architecture
- 16-megabyte linear address space
- Memory-mapped I/O
- Powerful instruction set with 14 addressing modes and 5 main data types
- Seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register
- High-performance, 8-MHz operation
- Opportunity for 9826A/9836A family growth as the 68000 family evolves.

The input and output signals of the 68000 are shown in Fig. 2. The processor status bits indicate the state and the cycle type currently being executed. The 6800 peripheral control signals are included to allow for backward compatibility when interfacing to 6800-family peripheral devices (e.g., the 9826A display controller). The system con-

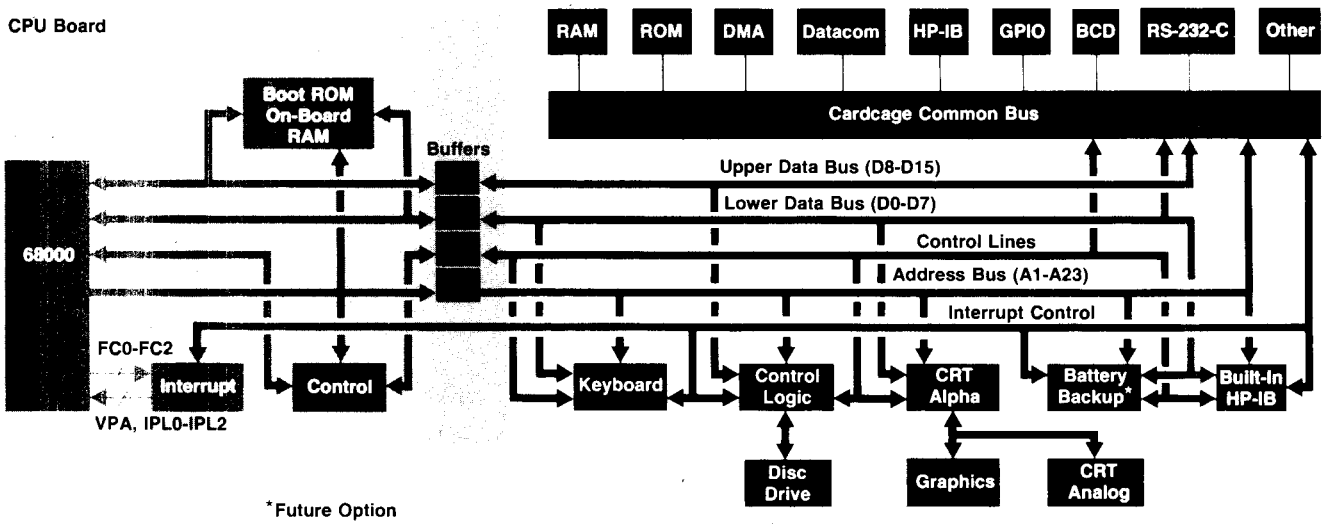


Fig. 1. *Block diagram of 9826A hardware system.*

trol inputs are fairly standard with the exception of bus error. The bus error input $\overline{BERR}$ informs the processor that there is a problem with the cycle being executed. Upon receipt, the 68000 aborts the cycle, places information on the stack regarding the incompleted cycle, and branches to a software bus error handling routine. In this way an intelligent response to an incompleted cycle is possible. The address bus is capable of addressing 8 million words of data and the data bus provides a general-purpose 16-bit data path. The asynchronous bus control lines supply the necessary handshake to coordinate a data transfer. The bus arbitration control lines determine which device in the system will be the bus master. Finally, the interrupt control input lines indicate to the 68000 the encoded priority level of the device requesting an interrupt.

The hardware system's address and data buses are simply the buffered form of the 68000's address and data buses. Except for a few special-purpose additions, the same situation applies to the control and interrupt lines.

Memory Map

The system memory map is shown in Fig. 3. The first four megabytes are assigned to system ROM and the boot ROM. This part of the address space is a synchronous response area. When the CPU board recognizes an access to this address range, it internally sends a $\overline{DTACK}$ (data transfer acknowledge) signal to the 68000 which creates a five-clock-cycle access. This automatic $\overline{DTACK}$ mechanism is used because ROMs are synchronous devices. It avoids the need for duplicating a fixed-delay or $\overline{DTACK}$ circuit on each of the ROM cards. Instead, this function is centralized on the CPU board.

Two ROM cards are available: a 128K-byte ROM board with sixteen 8K×8 ROMs and a 512K-byte ROM board with sixteen 32K×8 ROMs.

The next four megabytes are assigned to the internal and external I/O spaces. Fig. 4 shows the bit assignments for the internal and external I/O accesses.

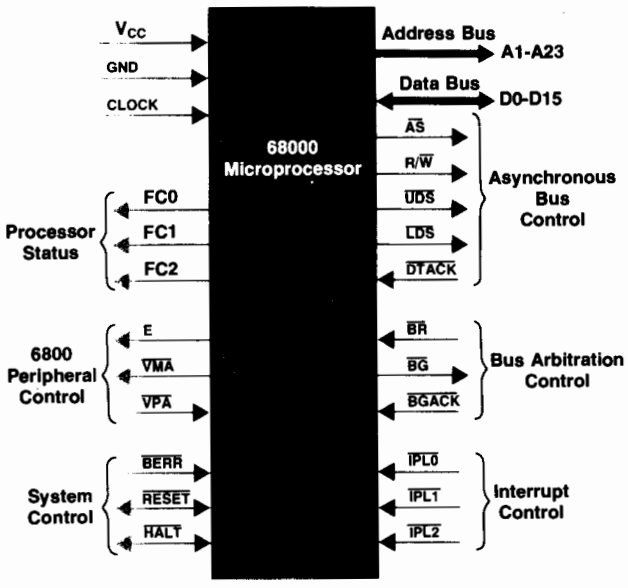


Fig. 2. *Input and output control signals for the 68000 microprocessor.*

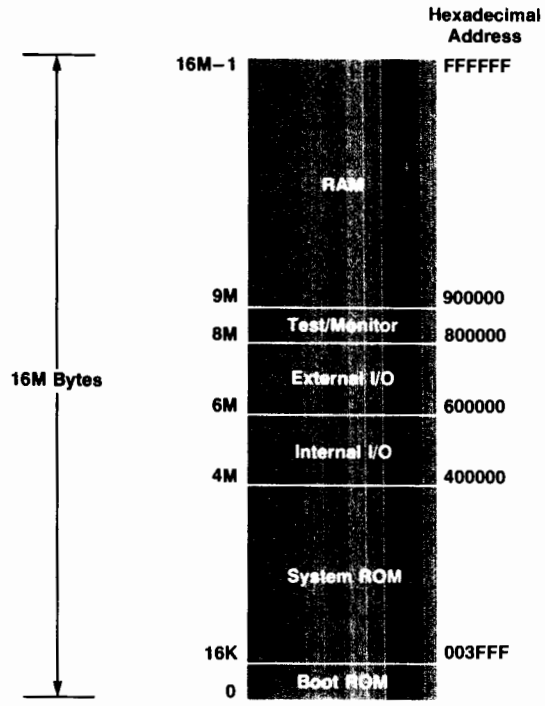


Fig. 3. *Memory map for 9826A and 9836A systems.*

The four megabytes available for I/O are mapped into 32 select codes with 64K bytes of address space each. Hence, only two of the megabytes are actually used. Select codes 0 through 7 are used by internal devices and select codes 8 through 31 are used by external devices. The 9826A and 9836A operating systems automatically convert the user-entered select code into the proper memory address; that is, for select codes 0 through 7, bit 21 is set to zero and for select codes 8 through 31, bit 21 is set to one. Thus, if an external I/O card is set by the user to an internal select code number, the card will never be addressed by the operating system. This eliminates the possibility of an external card's trying to communicate on the bus at the same time as an internal card.

The internal I/O space is broken into synchronous and asynchronous portions. The asynchronous portion requires that the internal peripheral respond with $\overline{DTACK}$ in much the same manner that external peripherals and the read/write memory card do. Internal peripherals residing in the synchronous portion of the address space use the automatic $\overline{DTACK}$ mechanism, much as the system and boot ROMs do. However, in this space, the number of clock cycles for access by the CPU board is programmable from five to eight cycles via address bits 14 and 15.

The next megabyte in the address space is used for HP's internal testing and monitoring. The remaining seven megabytes in the address space are reserved for RAM.

CPU Board

The CPU board (Fig. 1) contains two 8K×8 boot ROMs. These ROMs contain the 68000 reset, interrupt and other exception vectors as well as the necessary information to

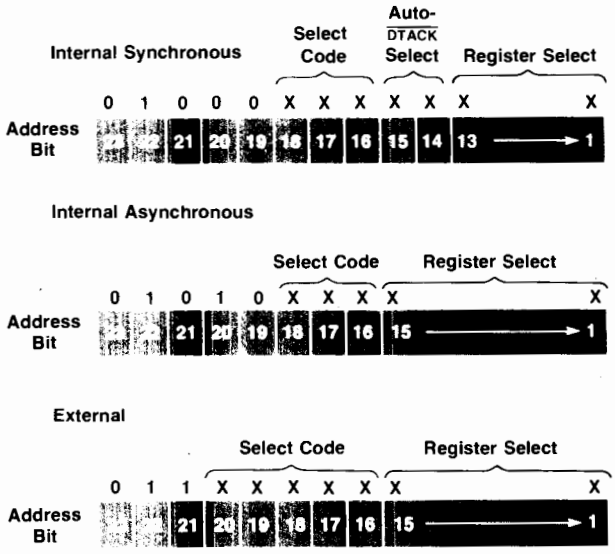


Fig. 4. *Address bit assignments for internal and external I/O accesses.*

initialize the system on power-up and begin execution of the appropriate language system. The 68000 requires that these two ROMs reside at the very bottom of the address space (Fig. 3).

The CPU board also contains a 64K-byte block of read/write memory. This block of memory has the novel ability to locate itself in the address space automatically as the lowest-address block of RAM in the system. This capability is necessary because the memory address switches on the CPU board are not accessible to the user. Furthermore, flexibility requirements of the system necessitate that the CPU board RAM be relocatable and self-configuring. When the system is powered up, the memory block is disabled by the autolocate circuit shown in Fig. 5. This prohibits the memory block from appearing in a random location dependent upon the power-up state of the D-type flip-flops. Fig. 6a shows the power-up signal conditions for the circuit in Fig. 5. Should an access to a memory location occur that would create a comparator match, $\overline{CS}$ is still disabled (Fig. 6b) until memory-block-select enable ($\overline{WAKEUP}$) becomes valid (Fig. 6c). This means that the operating system is free to begin accessing memory for power-up routines.

An early activity for the operating system at power-up is

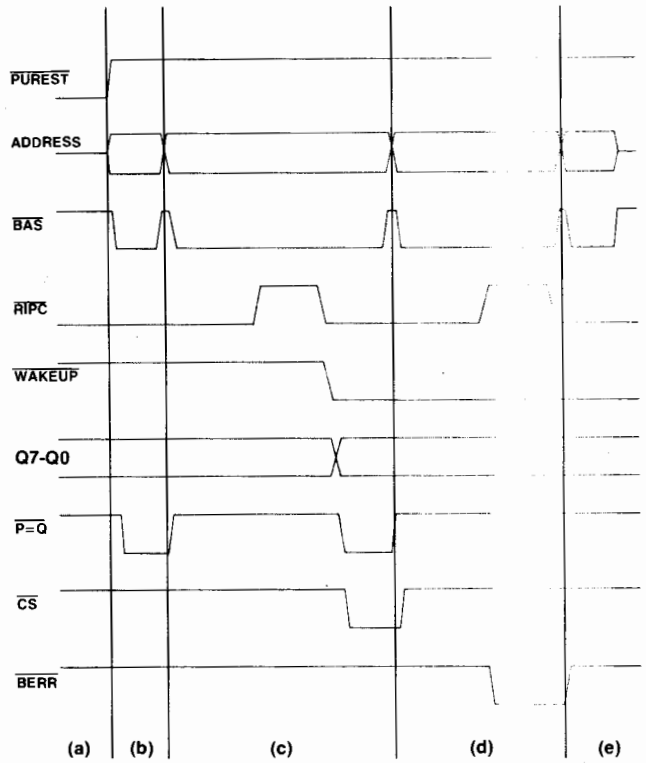


Fig. 6. *Timing diagram for autolocating circuitry.*

to determine the amount of available memory in the system by beginning to search at the top of the address space and then searching down until the first vacant location is found. At this point, a bus handshake timeout $\overline{RIPC}$ occurs, indicating that no memory is present to respond. The circuitry intercepts this signal and fills this first vacant location (Fig. 6c). The memory block controlled by the autolocate electronics responds and the operating system continues the search for another vacant location. Finally, a second vacant location is encountered. This time the circuit sends the $\overline{RIPC}$ signal on through as a bus error $\overline{BERR}$ (Fig. 6d). The system then logs this information and continues normal program execution (Fig. 6e). In this way, the controlled memory block is automatically assigned to the first vacant location in the address space without user or special operating system intervention.

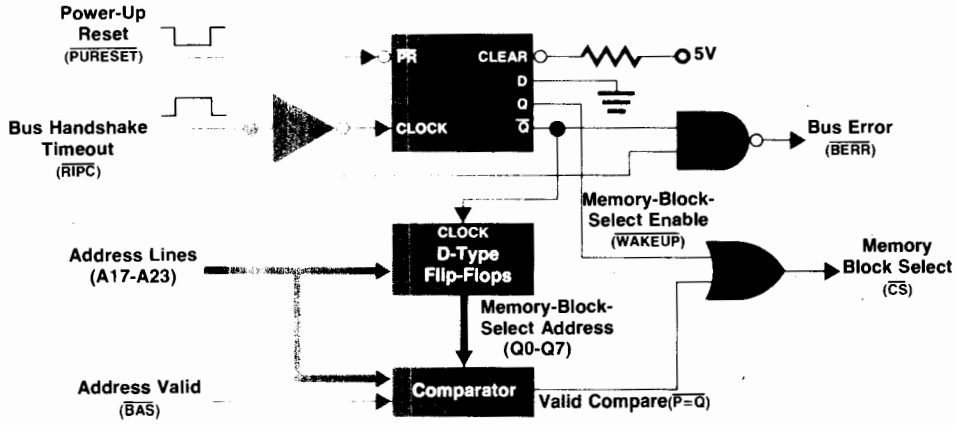


Fig. 5. *Simplified block diagram of autolocating electronics for CPU memory block.*

Product Design for Easy Production

The product concept of the 9826A and 9836A diverges somewhat from that of traditional HP desktop computers. The requirement of achieving a performance/cost ratio up to five times higher than for previous products was not, by itself, unusual. However, the 9826A and 9836A also are intended to become benchmark instruments in other ways such as simplified assembly and servicing, testing, reliability, high-volume production, portability in a highly integrated product, innovations in EMI/RFI protection, and visual compatibility with rack-mounted instruments.

Low Cost—High Volume

Creating a low-cost package was compatible with the expected high volume because higher volumes can justify higher expenditures for tooling. Complex molded plastic parts were designed to combine as many functions as possible and to eliminate sheet-metal parts. Details are molded into the plastic parts so that an absolute minimum of fasteners is required.

In many instances, a standard component is used as a primary structural member instead of adding another part. For example, the cooling fan acts as the structural member of an assembly that mounts the fan to the base, supports the large dc filter capacitor in the power supply, provides guides for four printed circuit boards, and divides the flow of cooling air for the different sections of the instrument.

The rear panel, molded into the structural foam base, is unsupported in a front-to-back direction. The molded printed circuit board guides are unsupported in a side-to-side direction. However, joined together, these elements become a rigid structure. The power transformer depends heavily upon mounting to the fragile rear panel, but imparts massive strength to the assembly by acting as a support member.

A single molded plastic part, mounted with two screws under the keyboard, provides a retainer and guide for the power switch button, mounting (with no fasteners) for the power switch, a hinged safety cover for the power switch terminals, and a red visual indicator to signal the on position to the operator.

The external package is formed by four structural foam-mounted parts held together by eleven screws. Efforts were made throughout the design to eliminate hardware and reduce the number of parts. Where it was practical and useful, modular subassemblies were created to reduce assembly labor.

Reliability

The necessary proof of mechanical reliability is extensive testing. It was presumed that reducing the parts count would increase reliability, but the challenge lies in maintaining structural integrity while eliminating fasteners. It is much easier simply to add another screw or another part. To put it another way, simplified designs are the hardest to execute.

Early in the project it was decided that extensive testing with real molded parts would be essential to achieving both structural simplification and reliability. A year before production release, temporary molds were tooled up for all major plastic parts. This allowed testing and modification using actual parts before committing to production tooling. Sometimes the results eliminated more hardware than anticipated, and sometimes critical weak points were discovered early. But most important, the result is a deceptively simple, reliable product.

One of the major benefits of early testing was in the area of cooling. Some early discoveries and decisions were made, a few of which are controversial.

On the 9826A and 9836A, the cooling fan blows out, rather than in. This disallows the use of a filter because it is not necessary to filter the air going out of an instrument and filtering all of the air entering the instrument is impractical. A filter would also decrease the cooling efficiency of the air flow. A filter can become clogged if not serviced periodically, causing damage through overheating.

The advantages are appreciable. First of all, the heat of the fan is kept out of the product. Also, since the power supply area generates the most heat, this hot air is drawn out immediately, and not subsequently into other areas of the instrument. Cool ambient air is drawn in through natural openings such as the keyboard, side vents, and I/O panels, and it flows over the internal components and circuit boards on its way to the fan. Most of the air is channeled through the power supply area at high velocity, and combined with a custom-designed heat sink, serves to maximize heat transfer from the power supply.

The air flow is allocated to various sections of the product according to tested need. Two methods were used to determine this. Component temperatures were monitored under diverse conditions, and smoke was used to determine air flow patterns and relative velocities. In the smoke tracing, transparent case parts allowed visual observation of air flow.

The individual I/O modules are kept cool by incorporating them into the internal air flow pattern, rather than treating them as external plug-ins, as was done on other HP desktop computers. They remain user-removable from the rear panel.

Industrial Design

The 9826A and 9836A are designed to be rackable/stackable and visually compatible with other HP rack-mount instruments. The products have their own beauty as stand-alone desktop computers, blending into an office environment quite naturally, but the crisp lines were originated with instruments in mind.

Standard rack-mount dimensions are maintained, and the internal peripherals (CRT display, flexible disc drive) are integrated into a package only 18 cm high (45 cm for the 9836A). The power switch is located on the front of the keyboard and the knob is positioned at the upper left corner of the keyboard for convenient user operation. The front panel of the CRT display is slanted back at 10° for proper user viewing angle.

Stacking an instrument on top of the 9826A or the 9836A, the user finds that a standard stacking recess (slot) is provided, matching the special feet on most HP instruments, and locking the two together.

Production Engineering in the Lab

There were certain priorities for the 9826A/9836A project that required more than the usual design function. Early in the project, a separate lab group was created and chartered to design the production process. This included such tasks as assembly procedures, ease of assembly, production testing, simplification of servicing, and the tools and fixtures for assembly. Production changes after introduction were to be avoided by including production engineering during the design phase, rather than later.

Numerous design details were modified early as a result of this approach. What may seem to be easy on paper, or even on a prototype, may not be so perfect in the real world of high-volume assembly. For example, one screw that is accessible but awkward to reach may seem to be a trivial problem to the designer but its awkwardness is magnified many times to the assembly or service person who has to deal with it on a daily basis.

EMI Innovations

Much research and experimentation had taken place previ-

ously on EMI (electromagnetic interference) reduction. It was felt that, it should be possible to design an instrument with "quiet" circuitry, such that the surrounding enclosure need only serve as a container and provide visual and mechanical functions.

Techniques were known for reducing EMI at the source, such as signal shaping, use of ground planes, and elimination of long conductors that act as broadcast antennas. What was not known was how much noise could be eliminated at the source.

The approach taken was a conservative one. It was assumed that the effort to reduce the sources would be ineffective, and so the case parts were designed to include total zinc arc-spray coating on the inside, full conductive gasketing between all parts, and complete grounding everywhere. Then, when all shielding provisions were in place, the entire design group was encouraged to ignore them and independently design for low EMI (output and susceptibility). All boards and subassemblies were range-tested and modified numerous times to eliminate EMI.

The result is that only minimal external shielding is needed. The zinc arc-spray was replaced by inexpensive conductive paint. The full conductive gasketing was eliminated. Only six contact points are needed between case parts—the 9826A and 9836A can pass VDE and FCC requirements without them, but they increase the design margin.

Another cost-saving benefit of early EMI testing was in the area of CRT shielding. A CRT shield is needed to eliminate the potential effect of stray CRT magnetic fields on the flexible disc heads. Instead of assuming the need for a very expensive mu-metal shield over the CRT and flexible disc drive, experiments were performed to establish what configuration and material would do the best job. At the frequencies and field orientations involved, the best results were achieved using a partially open aluminum shield, which happened to be a very inexpensive design.

The success of the EMI design, as well as that of the other innovations mentioned here, rests primarily with the concept of very early testing. There is much to be learned from testing, and it is not necessary to wait for production-ready parts to start it.

Acknowledgments

We are deeply indebted to Perry Pierce and Will Featherolf, who originated many of the 9826A and 9836A product design innovations. Special recognition goes to Don Faatz, one of HP's resident plastics experts, for his contributions to the success of our sometimes "crazy" molded parts. Thanks to Jerry Blanz for providing the impetus and technical guidance for our EMI innovations.

We also want to thank the Desktop Computer Division's model shop, under Harry Hammers, for once again proving that they can fabricate anything if you can draw it on paper. A special thanks also goes out to Evelyn Jones, Lynn Peden, and Wanda Wire for their valuable help in making the 9826A and 9836A easy to assemble.

-Dave Brown
-Pat Balliew
-John Armour

Read/Write Memory and DMA Boards

At introduction the 9826A was able to support about one-half megabyte of RAM. Since introduction new technology has allowed the original 64K-byte RAM board using 16K-bit dynamic RAMs to be replaced by a 256K-byte board using 64K-bit dynamic RAMs. This expands the maximum system memory size to more than two megabytes. Only the 256K-byte board (Fig. 7) will be discussed in detail. However, the 64K-byte board is very similar in architecture.

The RAM boards can be plugged into any of the eight backplane slots. They can run regular asynchronous accesses or a special synchronous access. When a RAM board runs an asynchronous access the data will be valid on the bus for 50 ns before the board pulls the data transfer acknowledge ($\overline{\text{DTACK}}$) line low. During a synchronous access the 68000 CPU does not clock data until one clock cycle after it detects $\overline{\text{DTACK}}$. If $\overline{\text{DTACK}}$ is given after data becomes valid, one clock cycle of access time will be wasted by the internal timing of the CPU. By giving a synchronized early $\overline{\text{DTACK}}$ the synchronous accesses will run two clock cycles faster than a normal asynchronous data transfer (five CPU clock cycles instead of seven). The board is synchronized with the CPU by use of a signal called $\overline{\text{ENDT}}$ (enable $\overline{\text{DTACK}}$). This is a bus signal provided by the CPU that is gated into the CPU's $\overline{\text{DTACK}}$ input by the RAM board to generate a five-cycle synchronous access.

The CPU normally runs synchronous accesses with the RAM board. However, if the dynamic RAMs are being refreshed when the access begins, the access is delayed and synchronization with the CPU is lost. The RAM board must then run an asynchronous access.

The DMA (direct memory access) board always runs asynchronous accesses with the RAM board. This allows for a low-cost, high-performance DMA design. The low cost is achieved by supporting only one type of DMA cycle for both low-speed and high-speed I/O cards. The high performance is realized by starting the memory access in parallel with the I/O card access. This requires that, on writes to the RAM board, the RAM board be able to complete the access up to the point where data is supposed to be valid, and then wait for the data strobe.

Since refreshes are generated by a state machine that is not synchronous with the bus, it is necessary to use an arbitration circuit that prevents glitches or astable signals from getting into the RAM control circuitry (see Fig. 7). Dynamic RAMs can lose data if the control signals are glitched or the address lines are not set up or held properly. The refresh circuit does the arbitration by holding off any new accesses and waiting for any current access to end. Glitches are prevented by use of the open-collector gate, the RC network, and the Schmitt trigger latch shown in Fig. 7. The RC network stretches the pulses to ensure that any signal that comes out of the open-collector gate will be long enough to set the Schmitt trigger latch.

Keyboard and Real-Time Clock

The circuitry for the 9826A/9836A keyboard, knob, programmable audio alarm, and real-time clock is based on an 8041A microprocessor. The main focus of the design was to minimize cost and maximize reliability. The 8041A was chosen as the microprocessor because it is low-cost and designed to be slaved to a master processor. This allows it to be interfaced to the 68000 with minimal circuitry.

The keyboard has 103 keys plus shift and control. The 9826A and 9836A have keyboard options for German, French, Spanish, Swedish/Finnish, and Katakana (Japanese). The system language and configuration jumpers

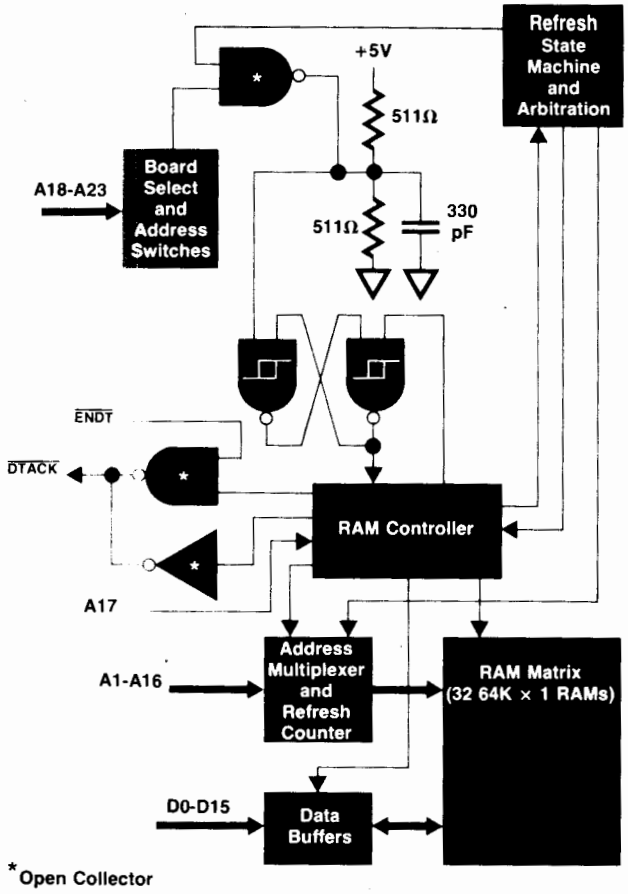


Fig. 7. *Block diagram of 256K-byte RAM board.*

are also scanned as keys. The language jumpers are on the keyboard board so this is the only board that has to be changed when the language is changed. It has to be changed in any case because the keycaps have to be changed. The keyboard has two key rollover and automatic repeat. Both the delay until repeating begins and the repeat rate are programmable. Key debounce is implemented in the 8041A firmware.

The keyboard has a rotary pulse generator (knob) that provides an analog-like input. It has been found to be very useful for program editing and also can be used by the user program as a human input. The maximum rate at which the knob can interrupt the system is programmable from 100 Hz to 0.39 Hz. The number of pulses detected (128 maximum) is passed to the 68000 when the interrupt occurs. About 120 pulses are generated for each revolution of the knob.

The audio alarm is programmable in frequency from 81 Hz to 5167 Hz with 81-Hz resolution. Its duration is also programmable from 10 ms to 2.55 s.

The real-time clock is implemented entirely in the 8041A. The clock has 10-ms resolution and is accurate to 50 ppm. Besides keeping real time, three additional user and two system functions are provided. The user functions are a delayed interrupt, a cycle interrupt, and an on-time match interrupt. The system functions are a periodic interrupt at 10-ms intervals and a fast-handshake timeout interrupt. The fast-handshake timeout interrupt will cause a nonmaskable interrupt and can be used to pull the system out of loops that could hang it up.

The keyboard debounce, the automatic repeat rate and delay, the knob interrupt rate, the alarm duration, and all the real-time clock functions are implemented in a single timer routine. Routines for communicating with the 68000 system processor are shared by all the functions. To keep the functions separate on the system level all data is passed to the 68000 through an interrupt service routine. Data is accompanied with a status byte that describes what the data is for. Defining all the individual functions as one larger function allows all the functions to be implemented in the 1K-byte ROM of the 8041A.

CRT Display

The 9826A display is a 190-mm-diagonal CRT with a raster 120 mm wide and 90 mm high. The subsystem supports both graphics and alphanumeric displays with a graphics resolution of 400×300 pixels (displayed dots) and an alphanumeric organization of 25 lines of 50 characters each. Alphanumeric and graphics pixels can be displayed simultaneously, with pixels from each exactly overlapping in position. The pixel spacing (0.3 mm) is the same in both the horizontal and vertical directions.

Both the alphanumeric and graphics frames have independent dedicated memory buffers which are mapped into the 68000 address space. The 9826A display is refreshed from these buffers at a 60-Hz rate with no use of the system bus. The alphanumeric memory is 2K bytes, one byte per character. The graphics memory is 16K bytes, using one bit per pixel. CPU cycles to the display memory are windowed between display refresh cycles and are synchronous with the display logic. The worst-case access time of this memory is 1.2 μs, about twice that of the system memory.

The alphanumeric display is separated into two functionally different areas: the normal text area and the softkey area. Each of these areas has an independent hardware on/off control for software simplification. In the normal text area, characters appear white on a black background. In the softkey area, characters appear in half-bright inverse video (black on dim white). If the graphics mode is on, the alphanumeric and graphics signals in the 9826A text area are EXCLUSIVE ORed and invert each other. In the 9826A softkey area graphics pixels get ORed with the softkey labels and appear as full bright over half bright, putting softkey labels in the background.

The character lookup ROM contains all 128 ASCII characters (including control), plus the HP standard European extension set and the Katakana set, for a total of 256 characters. In the 9826A the characters are contained in a cell 8 pixels wide by 12 pixels high. The ROM has space for the full cell with no defaulted blanks, either horizontally or vertically. This allows macrocharacters to be created that are formed from two or more character cells. The display timing is generated by a 6845 CRT controller, which is programmed by system software at power-up.

The CRT drive system uses a conventional resonant sweep circuit for horizontal deflection, and a linear amplifier for vertical deflection. The beam accelerating potential (10 kV) is derived from the horizontal sweep circuit with a flyback transformer. The horizontal scan rate is 19.23 kHz for the 9826A and the screen refresh rate is fixed at 60

Hz. Pixel information is shifted out at a 10-MHz rate, with display memory read cycles required every 800 ns.

An intensity control is provided so the user can adjust the display brightness for varying ambient light conditions. The range of this control is from about 15 to 100 candelas per square metre. The control is located just under the left of the screen in the 9826A and can be easily adjusted without tools, even in rack-mount configurations.

A larger (310-mm diagonal) CRT is used in the 9836A with a raster 200 mm wide and 150 mm high. Graphics resolution is 512×390 pixels and text can be displayed in a format of 25 lines of 80 characters each. Because of the larger CRT, several operating parameters are different from those in the 9826A. The beam potential is 12 kV, the horizontal scan rate is 24.9 kHz, and the refresh rate can be set at 50 Hz or 60 Hz. Alphanumeric pixels are displayed at a 25.77-MHz rate and graphics pixels at a 17.18-MHz rate. Also, the character cell is 9 pixels wide by 15 pixels high.

Flexible Disc Drive

The flexible disc controller interfaces a 5.25-in drive to the 9826A bus. The controller supports the HP LIF (logical interchange format) standard. Each disc can store 264K bytes. The average access time is 300 ms if an access has occurred in the last three seconds and about one second otherwise.

The major component of the interface logic is the 1791 FDC (flexible disc controller). The FDC is responsible for seek operations, controlling read and write operations of the soft sectors, and encoding and decoding the MFM (modified frequency modulated) data stream.

The controller board has extended command/status registers. These registers allow the system to control the drive and the on-board transfer state machine. On the drive side of the FDC one block is charged with data recovery on read operations and another block precompensates the write data stream to remove some bit shift.

The overall design goals were to maximize transfer rate and provide testability while minimizing cost. The 9826A and 9836A meet these goals by having the low-level driver in the system boot ROMs instead of an on-board microprocessor. This provides direct access to every register from the system bus and allows system or test-code control of the retries that occur after a soft error. The cost of additional code in the system boot ROM is less than adding a separate microprocessor. The transfer rate is maximized by having the 68000 system processor available to move data to and from a RAM buffer in certain time-critical windows.

This 256-byte RAM buffer is shared by the controller and the system bus. It is controlled by the transfer state machine during read or write operations and is not accessible until the operation is complete. At all other times the buffer is part of the memory-mapped I/O space of the system bus. Each byte can then be randomly accessed. Thanks to the RAM buffer, disc transfers are not time critical. The system waits for the sector read or write operation to complete, but any I/O interrupts or DMA activity can proceed without jeopardizing the disc transfer. The system transfers data out of the RAM buffer in the intersector gaps, the time between the end of one sector and the header for the next sector (see Fig. 8). This allows for noninterleaved operation at a high

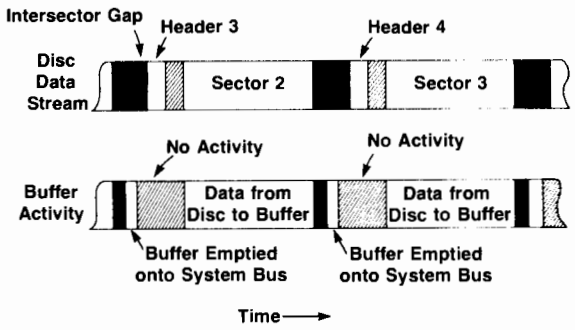


Fig. 8. *Timing diagram of a transfer from internal flexible disc storage at the maximum rate.*

transfer rate. The entire disc can be loaded at a rate of 16K bytes per second.

The soft-error-rate goal for the 9826A flexible disc controller was less than one error per 10^9 bits passed. This goal was met in four phases. First, the timing margin (Fig. 9) was optimized by using a phase-locked loop to create the read clock. Second, the signal-to-noise ratio at the head amplifier output was improved with shielding and by controlling conducted interference from the power supply. Third, exhaustive investigation of each error led to improvements in the lockup controller. Finally, measuring the error rate with the system subjected to 1-kV 50-Hz line transients led to discovery of problems in the drive cable grounding and termination that were easily corrected.

The typical production unit shows a soft error rate of 10^{-10} to 10^{-11} errors/bit. Units with error rates better than 10^{-10} actually exceed the specified media life before the first soft error occurs.

Power Supply and Future Powerfail Option

Primary considerations in the design of the 9826A power supply were reliability and compatibility with the future powerfail option. The power supply has to provide 110 watts of regulated power at +5Vdc, +12Vdc, and −12Vdc from an unfiltered dc voltage which can range as low as

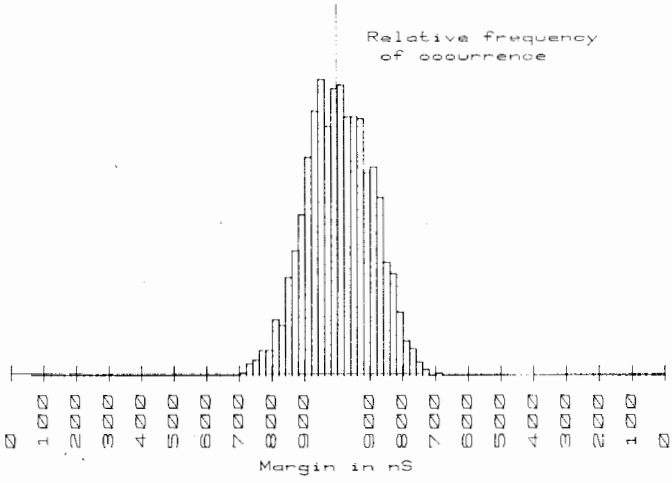


Fig. 9. *Plot of timing margin for 100,000 data transitions between the system and the flexible disc.*

14.5V during battery operation.

To meet these requirements, conventional switching regulators are used to implement the +5V and +12V supplies, and a dc-to-dc converter is used for the −12V supply. The dc voltage to the regulators is provided by a transformer-isolated, full-wave-rectified supply with a nominal voltage of 28Vdc so that transfer to the battery can be simple and practically instantaneous.

Short-circuit protection is provided for all three supplies, and overvoltage protection is provided for the +5V and +12V supplies. The −12V supply does not need overvoltage protection because its design provides inherent protection against shorted switching transistors.

The powerfail option to be available in the near future (Fig. 10) will allow the 9826A to ride out short power interruptions and power down in an orderly manner when power interruptions last longer than a few seconds. With this future option the entire machine can be operated for up to 60 seconds after loss of ac line power, allowing the user to store important data on flexible disc before powering down. The powerfail option also will contain a real-time clock and 112 bytes of memory. The amount of powerfail protection, the amount of time that power must be gone before giving a powerfail interrupt, and the amount of time that power must be back before leaving the powerfail state will all be programmable between 0 and 60 seconds. These programmable features and the real-time clock use another 8041A microcomputer to control the powerfail system. Analog comparators will monitor powerfail and battery conditions and provide this information to the 8041A. An 18-volt, 2-ampere-hour nickel-cadmium battery will provide power to the system for a short time after the power fails and to the real-time clock and CMOS* memory when the power is off.

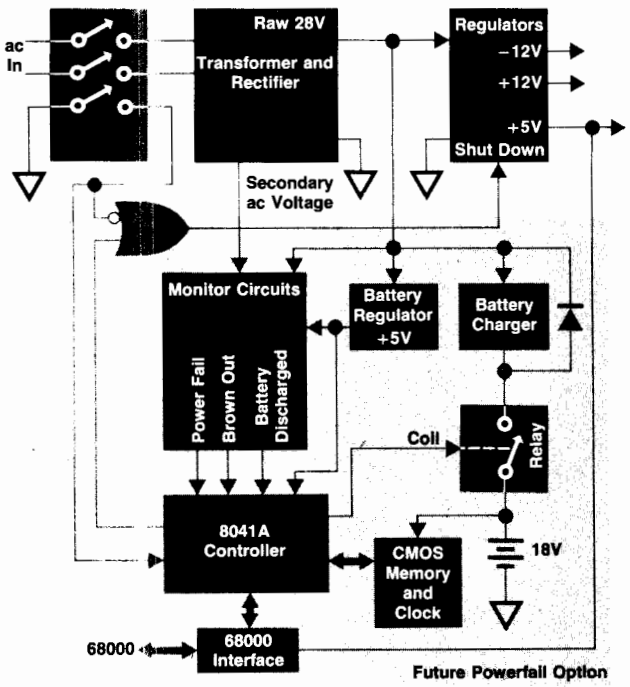


Fig. 10. *Block diagram of power supply and optional powerfail circuit.*

It will be necessary to have a switch ganged with the power switch to tell the difference between a powerfail and the system being turned off. This signal will go to the 8041A powerfail controller but also will override the controller and shut off the system if the power is turned off. Thus, even if the controller fails the power switch will turn off the machine.

The battery will be connected to the unfiltered 28Vdc supply through a transfer diode and relay. The relay will close at power-up, providing nearly instantaneous protection through the transfer diode when power fails. The relay will open at power-down so the controller does not have to be powered up and leakage into the power supply will not drain the battery. The battery will be charged by a constant-current charging circuit powered from the unfiltered 28Vdc supply. Typical recharge time for a fully discharged battery should be 14 hours. After a 60-second backup cycle is provided by the battery it will still be able to keep the real-time clock and CMOS memory operating for a week.

The power will be monitored in three different ways. The ac on the secondary of the transformer will be monitored to see whether any cycles are missed. The unfiltered 28Vdc supply will be monitored to detect reductions of more than 10% in the ac line voltage. Either one of these conditions will be considered a powerfail by the controller. The operating system can interrogate the controller to find out the reason for the powerfail. The unfiltered 28Vdc will also be monitored to check whether the battery is discharged to the point where it will not be able to provide the 14.5V required by the regulators. When this signal is received the controller will shut down the regulators.

The 8041A controller and the monitors will be powered by their own +5V regulator. This will allow the controller to remain operational until the rest of the system is powered down. When power comes up the controller will monitor the power until it is valid (has proper voltage). When power becomes valid the battery charging relay will be switched on and the rest of the machine powered up. The controller will be programmable to give interrupts to the operating system for powerfail or power coming back. The operating system will be able to send a command to the 8041A to shut down the power at any time. This will allow a user to conserve the powerfail battery charge if the user's graceful power-down routine takes less than the 60 seconds allowed. When the power is shut down for any reason (command, switch off, protect time up, or battery discharged) the regulators will be turned off and the charging relay opened after a delay. The delay will ensure that the current through the relay is very small when it is switched, thus improving the reliability of the relay.

Real time will be kept by use of a CMOS counter and a CMOS RAM powered by the battery at all times. 112 bytes of the CMOS RAM will be available to the operating system or the user to store system configuration information.

Quality and Reliability

An early commitment was made to invest resources to

*Complementary metal-oxide-semiconductor.

Instrument Burn-In

The concept of burn-in testing for production instruments has existed for some time. Originally it was introduced to screen out unwanted infant failures in the field, but since has become an accepted part of production testing. In light of recent efforts to increase productivity by optimizing the production process and thereby minimizing work in process (WIP), the existing burn-in or aging philosophy rightfully has come under attack because the cost to repair production failures at the instrument level adversely affects the goal of reduced WIP. An obvious objective would be to test for and eliminate failure mechanisms at an earlier point in the process. The difficulty in achieving this objective is that it requires an understanding of just what is causing infant instrument failures. This in turn demands that exhaustive and accurate failure analysis be performed on failed assemblies to identify what tests could be performed earlier in the production cycle to exercise the predominant failure mechanisms.

In defining a production test plan for the 9826A Computer System, attempts to gather data in support of one test philosophy over another proved to be unproductive. It soon became apparent that additional failure analysis data was required to define accurately what earlier tests would be effective.

Since the ultimate goal is to minimize field failures, it was concluded that instrument burn-in would be the most representative indicator as to what failure mechanisms should be exercised. However, considering the ultimate goal of earlier testing, certain guidelines had to be established that would ensure proper use of instrument burn-in:

- Be comprehensive in developing the burn-in plan. Variation of environmental parameters such as temperature, humidity, and power provide a versatile and powerful test capability.
- Do not let instrument burn-in become a screen. A screen tends to become a permanent and unwieldy part of production. The goal is to eliminate the instrument-level burn-in.
- Failure analysis of failed assemblies and components is vital in determining failure mechanisms and is crucial to identifying whether or not testing earlier in the production process is feasible and/or economical.

The 9826A burn-in plan is an evaluation tool that is used to test the manufacturing process which is made up of several testable parameters: design, material, and workmanship. The source of a production failure will be ultimately identified as one of these parameters and action will be taken to adjust the process at the earliest point possible. This may result in adjustments being made by the component vendor, or at incoming inspection, board loading, or board test areas.

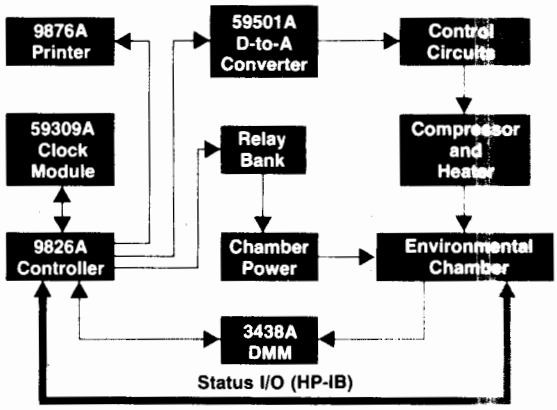


Fig. 1. *Instrument burn-in monitoring system.*

Assuming that this philosophy is administered properly, it naturally will result in reduced need for instrument-level burn-in as more and more problems are eliminated.

The resulting 9826A burn-in program incorporates several test environments:

- Temperature cycling is used to verify proper operating margin in electrical and mechanical assemblies[1,2,3]
- Power cycling is used to impose thermomechanical stress on the internal construction of encapsulated components such as integrated circuit and other semiconductor devices [4,5]
- Vibration testing sifts out early problems associated with instrument integrity and workmanship.[6,7]

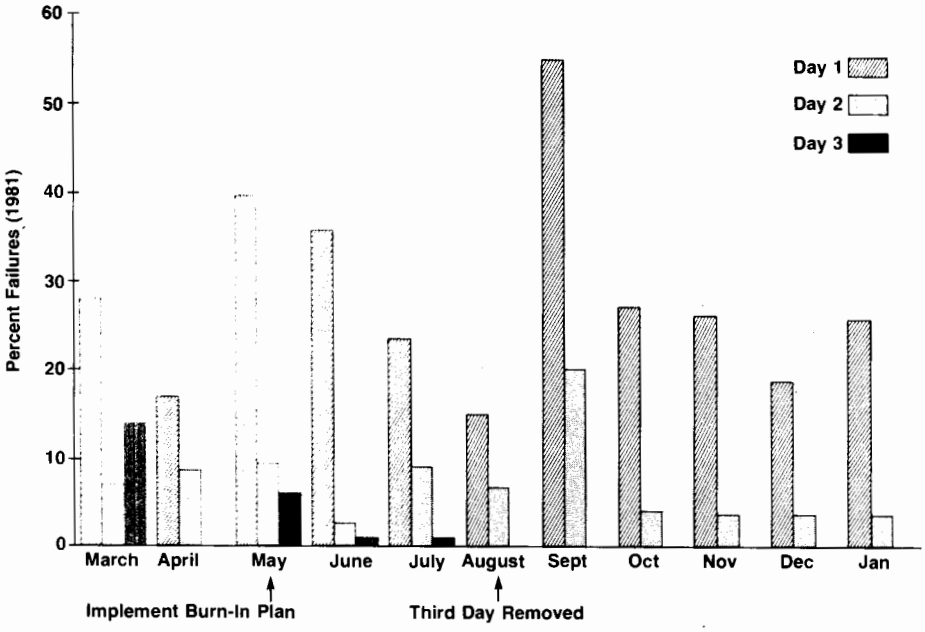


Fig. 2. *Failure rate per day of burn-in over an 11-month period. The burn-in plan was implemented in May 1981 and the third day of burn-in was eliminated that August. (Note: September's failure rate increase was caused by a single vendor problem.)*

In addition to these environmental tests, a monitor system (Fig. 1) is used to evaluate the status of each 9826A. The instruments in Fig. 1 perform the following functions:

1. Thermotron WP-1261-TCM-3-15 Chamber: 3.35 m× 3.35 m× 2.44 m walk-in temperature chamber designed to accommodate up to seventy 9826As for testing. With a load of 14 kW and 907 kg of mass, the chamber will provide a transition rate greater than one degree Celsius per minute.
2. HP 9826A Controller: provides chamber temperature and power cycling control, coordinates data flow from each 9826A under test, generates and formats failure reports, and backs up fail data on its internal flexible disc unit.
3. HP 59501A D-to-A Converter: converts computer programming outputs to an analog control signal for the chamber control circuitry.
4. HP 3438A Digital Multimeter: measures chamber temperature by sensing a thermistor and then outputs the measurement via the HP-IB to the system controller.
5. HP 59309A Clock Module: provides battery backed-up real time for the system.

Data collected during the first eleven months of 9826A testing indicates tremendous progress has been made toward the failure rate goals. Fig. 2 shows a definite contrast in May 1981 failure rates as a result of implementing the burn-in plan. The data shows that the first two days of burn-in were so effective that we were able to reduce the burn-in time to two days in August 1981.

A breakdown of the failure types found during failure analysis indicated that electrical failures far exceed other contributors.

Table I
Breakdown of Electrical Failures
(Data through January 1982)

Failure Mechanism	Percent of Total
Semiconductor-Hard Failures	37%
Temperature Sensitivity or Margin	46%
Other	17%

Further evaluation of the electrical failures (Table I) indicates that the major portion of the failures were identified with varying temperature and were intermittent in nature.

References:

1. D. Moss, HP Product Reliability Seminar, 1979.
2. R. F. Powell, "Temperature Cycling vs Steady State Burn-in", Circuits Manufacturing, Vol. 16., September 1976.
3. A. Coppola, "Thermal Cycling Makes Better Burn-in", Evaluation Engineering, Vol. 18, No. 2, March/April 1979.
4. M. R. Strange, "Power Cycling", Quality, September 1980.
5. G. F. Kujawski and E. A. Rypka, "Effects of ON-OFF Cycling on Equipment Reliability", Proceedings of 1978 Annual Reliability and Maintainability Symposium.
6. D. O. Patterson, "The Navy's Environmental Stress Screening Program", Proceedings, Institute of Environmental Sciences, 1979.
7. W. A. Mayers, "Applying Environmental Screening to Build Military System Operational Success", Proceedings, Institute of Environmental Sciences, 1979.

-Ken Fedraw

make the 9826A and 9836A very reliable products of high quality. Quality was not perceived simply as a set of tests to pass at the conclusion of the design cycle, but rather as a viable design parameter to be factored in during the development from the first hand-drawn sketch to the final product coming off the assembly line.

An early decision was made to use multilayer printed circuit logic boards that include a +5V bus and ground plane. Even though this conflicted with our goal to keep the factory cost as low as possible, the gain of having a system immune to noise and generating minimal EMI (electromagnetic interference) was worth it. Next, a decision was made to minimize the number of shared clocks in the system. Furthermore, the cardcage does not receive any clock and has an asynchronous bus. Had this not been done, the 8-MHz CPU clock would have had to leave the CPU board and travel about 20 cm across the motherboard and up the cardcage board. Then the CPU board and up to eight cardcage boards would all have been pulsing currents and switching voltages on the same clock edges. This was not consistent with our desire to have an EMI-quiet system. Finally, starting with our first wire-wrap system, we tested our units on the EMI range to determine if our theories were indeed yielding the results we intended.

An infrared thermal scanner was used to identify hardware components operating at elevated temperatures. This is important because semiconductor component reliability is related to temperature. The results of this test caused several component/circuit design changes to be made. Although these weak areas probably would have shown up in later tests, the thermal scan quickly identified them early in the design process.

Other design activities concerned with quality included early environmental testing and margin testing to failure of parameters such as power supply voltages and clock frequencies. These tests helped solve the most serious design problems early and provided the time to attack less frequently occurring problems. This procedure allowed us to

Ronald G. Rogers
Ron Rogers designed the flexible disc subsystem for the 9826A and now is a production engineer for the 9826A and 9836A systems. He came to HP in 1979 with experience as a development engineer designing test equipment and circuit breakers. A native of Topeka, Kansas, Ron received the BS degree in electrical engineering from Kansas State University in 1977. He lives in Fort Collins, Colorado, and enjoys sailing, hiking, and cross-country skiing.

Ken L. Burgess
Ken Burgess joined HP in 1978 after receiving the BS (1976) and MS (1978) degrees in electrical engineering from the University of Nebraska. He designed the CRT subsystem for the 9826A and now is a production engineer for the system. Ken was born in Lincoln, Nebraska and now lives in Fort Collins, Colorado with his wife, daughter, and son. He enjoys woodworking, boating, electronics, and listening to his hi-fi.

go into our major strife test program with fairly reliable machines.

The strife test program was an extension of similar testing done on the 9915A Modular Computer.[1] The purpose of the strife testing is to identify and eliminate sources of failure before the product is released. Thirty-six 9826As were tested for about 20 days of test time per unit. The tests resulted in 311 failures that were categorized into 173 design, 99 material, 26 process, and 13 test-related deficiencies.

Initial field warranty data indicates that the failure rate for the 9826A has been reduced by 33% as a direct result of design and process changes attributed to the strife test program.

Reference:

1. K. Watts, "A Unifying Approach to Designing for Reliability," Hewlett-Packard Journal, Vol. 32, no. 7, July 1981.

James W. McLucas

Jim McLucas joined HP in 1966 with R&D experience working with frequency synthesizers. He has worked in IC and thin-film production engineering at HP and designed the power supply and powerfail subsystems for the 9826A and 9836A. He is co-author of a paper on the use of variable-bandwidth active filters and a veteran of three years in the U.S. Army Signal Corps. He was born in Apalachicola, Florida and attended the University of Florida where he earned the BEE degree in 1963. While working at HP he completed the requirements for his MS degree at Colorado State University in 1974. Jim lives in Fort Collins, Colorado with his wife and five children, and likes hiking, camping, and skiing.

Robert J. Horning

Rob Horning is a native of Great Falls, Montana and attended Montana State University where he earned the BS (1976) and MS (1978) degrees in electrical engineering. Rob came to HP in 1978 and has worked on the hardware design of the 9826A and 9836A. He is currently concerned with future enhancements for these systems. Rob lives with his wife, son, and daughter in Fort Collins, Colorado. Outside of work, he plays softball, enjoys camping, and does improvements to his home.

Don D. Stewart

Don Stewart earned the BS (1976) and MS (1977) degrees in electrical engineering from Virginia Polytechnic Institute and State University. He joined HP in 1977 as a design engineer. Don has worked on the hardware design and production of the 9826A and now is a project manager for 9826/36 family enhancements. He is named as inventor on a patent pending on the memory autolocate concept used in the 9826A and 9836A, is registered as a professional engineer in Colorado, and is a member of IEEE. Don was born in Arlington, Virginia and lives in Fort Collins with his wife and their new daughter. He leads a youth group and sings in the choir at the local Baptist church, plays baseball and basketball, and enjoys hiking, hunting, skiing, and photography.

I/O Philosophy and Architecture for Instrument Control

by Loyd F. Nelson

ONE AREA OF APPLICATION for the HP Model 9826A and 9836A Computer Systems is as smart controllers. Fundamental to this application is the ability of the controller to be connected to all kinds of devices. In considering the I/O (input/output) system for the 9826A and 9836A, the first question that arises is, "Why not use the current HP 980xx series of I/O cards which already provide this facility?"

Since the method of connecting external devices to the 9826A and 9836A directly impacts the electrical and mechanical design of the machines, the question of whether to develop a new I/O series or use the existing 980xx series needed an answer. An initial investigation showed that on the average the cost of the I/O cards could be reduced approximately 40% by developing new designs. At the same time, performance would be improved. In the 1980s great competitive pressure is expected from low-end personal computers and the price of hardware will continue to decline. Six-year-old hardware developed for higher-priced products will not be competitive with the newer lower-cost

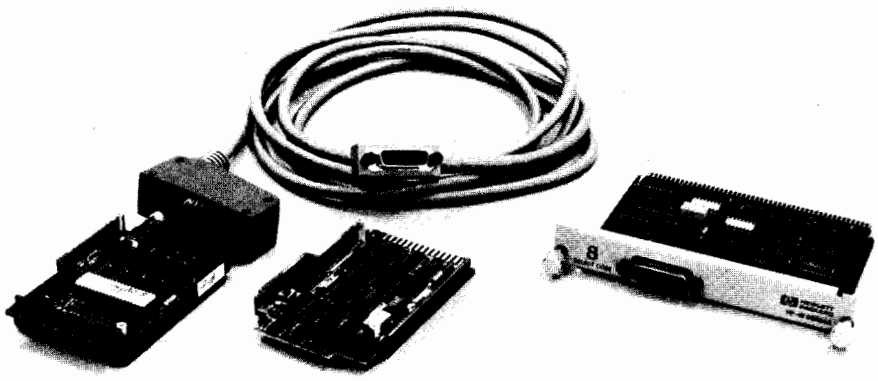


Fig. 1. *Physical comparison of the older 980xx series I/O card (on the left) and the new I/O card (on the right) developed for the 9826A and 9836A.*

generation.

Another advantage of designing new I/O cards is the opportunity to improve the reliability of the cards. The 980xx cards are housed in their own cases with no cooling. Their form factor is such that the circuitry has to be divided onto two small boards (see Fig. 1). The resulting design uses two very densely packed boards and requires many interconnections as well as being troublesome to assemble and test. A new design should be able to solve these problems and contribute to the improved performance and reliability and would be only a small part of the total system cost.

Physical Architecture

Once the decision was made to develop new I/O cards, the physical implications were quickly decided. Rather than enclosing the cards in their own case, it was decided that the 9826A and 9836A should be designed so that the cards reside totally within the mainframe. This has the twofold advantage that the cards can be cooled by the mainframe fan for an increase in reliability and that more power can be dissipated on the cards. The power supply and cooling for the 9826A and 9836A are designed to allow a power dissipation of 5.5 watts per card, an increase of 30% over the earlier 890xx series. The important +5V power supply current is increased from 450 milliamperes to 1 ampere. Added to each card is a metal backplate that connects directly to the metal backplate of the mainframe to provide an excellent safety ground.

The metal backplate, in conjunction with the attached metal connectors and mating metal connectors on the cable, also provides a significant improvement in limiting EMI (electromagnetic interference) radiation. Since the cable can be disconnected at the backplate, the I/O card is now independent of the cable length and the system can be configured easily. The space chosen for the cards allows an increase in printed circuit board area from 174 square centimetres for the older 980xx series cards using two boards to 206 square centimetres for the new cards using a single-board design (Fig. 1).

Electrical Architecture

One of the keys to providing high-performance I/O for a lower price is the scheme by which the 68000 CPU (central processing unit) communicates and interacts with the I/O interface electronics. There are basically two schemes in use at HP today. The first is microprocessor based. In this method each interface board contains standard interface hardware and a microprocessor or special-purpose IC (integrated circuit) with enough processing power to handle I/O-related operations. The interface between the I/O card and the mainframe CPU is on a relatively high level. The machine-level I/O code of the mainframe CPU is the same for all I/O interfaces. The microprocessor on each interface handles the peculiarities associated with the particular I/O protocol of the external device to which it is connected.

The second interface method between the mainframe CPU and the I/O interface electronics is nonstructured. In this method the standardization of the I/O interface hardware is kept to a minimum. The I/O hardware is concerned more with the peculiarities of its particular external device and is tailored to provide those functions in an efficient manner. The interface between the mainframe CPU and the I/O card is generally on a more primitive level than the other method. Each signal line on the bus tends to have only one meaning and does not mean different things at different times. If there are four interrupt levels, then there are four interrupt-request signal lines. The software I/O driver is different for each interface and tailored to the requirements and features of the interface hardware.

Which type of I/O interface method should be used on the 9826A and 9836A? This question might be best answered by looking at the hardware and software costs for each approach. In the software area the microprocessor solution appears to have an advantage because it provides a standard interface and a single standard set of I/O drivers for all interfaces. When a new I/O card is developed, the card can be inserted in the machine without having to change any operating system code. However, a closer look shows that the software problem is not really solved. Each I/O card is different and has peculiarities that must be controlled by the software. The software task has essentially moved from writing code for the mainframe CPU to writing code for the microprocessor on the I/O interface card. Tools are needed to handle development and debugging of software not only for the mainframe CPU but also for each I/O interface microprocessor. Because the mainframe driver and protocol must be defined and completed long before the requirements of any future I/O is known, what happens when a new I/O card has some feature or requirement that cannot be handled with the defined protocol?

The nonstructured interface hardware approach does not

prohibit the standardization of the mainframe I/O routines; it merely moves them to a higher level in the operating system hierarchy. The I/O drivers that would have been in the microprocessor code of each interface move to the mainframe CPU and are invoked by standard I/O routines that are above the drivers in the operating system hierarchy. Thus, the nonstructured approach provides a more flexible solution because one set of development tools can be used that allows the software to take full advantage of the I/O hardware features.

Another consideration is how fast the software runs. The microprocessor on the interface card generally does not have the instruction execution speed of the mainframe CPU. In smaller machines such as the HP-85A Personal Computer the discrepancy in speed is not so great and, therefore, the extra overhead associated with the microprocessor on the interface card does not affect the overall performance of the I/O. To overcome the speed problem on higher-performance machines, the I/O microprocessor usually is not a true microprocessor but a custom IC developed for that particular computer family. For the 9826A and 9836A, the schedule and the risk associated with developing a custom I/O microprocessor did not make it feasible.

On the hardware side, the 9826A/9836A I/O project wanted to take advantage of LSI (large-scale integration) devices currently in the marketplace. For example, the 9914 GPIB controller chip integrates a number of the HP-IB* functions and provides a flexible, cost-effective HP-IB interface. For the I/O microprocessor approach the question arises as to how the LSI device and the microprocessor can be made to share the I/O tasks. Since the microprocessor would be located functionally between the mainframe CPU and the external I/O control logic, it would be awkward to take advantage of the control functionality of the LSI device. The mainframe CPU would talk to the I/O microprocessor and the microprocessor, in turn, would talk to the LSI device. Of course, the design could be made such that the mainframe CPU could also talk to the LSI device directly. However, this would cause the standard I/O routines to become partially specialized, a deviation from the philosophy of a single set of I/O routines for all interfaces. Also, it would tend to make the I/O microprocessor redundant. With the nonstructured approach the mainframe CPU can always talk directly to the LSI device and take full advantage of its capabilities.

Finally, for the 9826A and 9836A marketplace, one of the goals is to keep hardware cost down. The I/O cards should take advantage of current LSI devices because they provide more functionality for the dollar as well as greater reliability and less power consumption than discrete MSI (medium-scale integration) devices. The interface approach should minimize the overhead circuitry required.

The BCD (binary-coded decimal) I/O card (HP Model 98623A) was designed and breadboarded using each approach. For the microprocessor approach, an 8048 (the same as the series used in the HP-85A I/O) was used as the I/O microprocessor. It was found that because of the interface circuitry required for the 8048, the design needed a significant amount of printed circuit board space, consumed more power than the nonstructured approach, and cost more. For the GPIO card (HP Model 98622A) the amount of printed circuit board space required for the external interface 16-bit registers made it doubtful that all the circuitry including a microprocessor could be made to fit on one board. Furthermore, all the timing and control logic necessary to communicate with the external interface 16-bit registers was still needed to provide twice the speed of the 9825, and it was not clear that a microprocessor could be used to advantage. For the HP-IB card (HP Model 98624A), the 9914 chip had to be used to take advantage of its performance and maintain compatibility with the internal HP-IB provided by the mainframe.

Thus, after considering the advantages and disadvantages of the two approaches and the practical problems of cost and printed circuit board space, the nonstructured approach was chosen as the basic I/O philosophy for the 9826A and 9836A.

I/O Interface Details

The 9826A and 9836A support three types of I/O: programmed I/O, interrupt I/O, and DMA (direct memory access). Fig. 2 shows the signals on the backplane associated with each type. Most of the signals are the signals of the mainframe CPU which have been buffered to provide the drive necessary for the bus. Because the bus is an asynchronous bus (no system clock), the presence of information on the address or data lines is determined from associated control signals. Such a design is susceptible to skew problems between the information and the associated control signal caused by differences in capacitive loading and differences in logic path delays of the respective signals. A significant amount of engineering time was used to determine the worst-case skews that could arise in the sys-

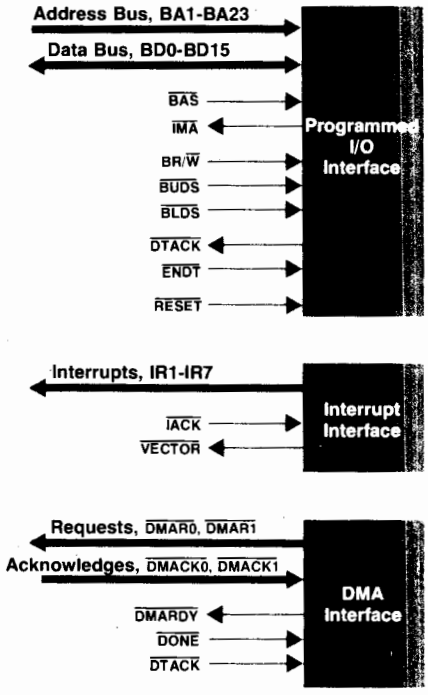


Fig. 2. *9826A/9836A backplane bus interface signals.*

*Hewlett-Packard Interface Bus. HP's implementation of IEEE Standard 488 (1978).

tem. The study resulted in a standard set of requirements for each entity to be connected to the bus in terms of the loading allowed, the drive required, the devices to use, and the timing to be adhered to.

Programmed I/O

The programmed I/O interface provides the basic communication between the mainframe CPU and the I/O. In keeping with the capabilities of the 68000 microprocessor, the I/O is accessed programmatically via memory address mapping. Consequently, both memory and I/O cards conform to the same timing specifications. Fig. 3 shows the signals and timing of the read and write cycles. The simplicity of the design will be evident from the following brief description of the read cycle. Just prior to the beginning of the cycle, the buffered read/write signal BR/$\overline{W}$, the buffered upper and lower data strobes (represented in Fig. 3 as one signal, $\overline{BDS}$), and the data transfer acknowledge signal $\overline{DTACK}$ are actively pulled high to ensure their proper polarity at the beginning of the cycle. The bus master (the entity currently controlling the bus) then defines the memory address to be read by driving the address bus, bits BA1 through BA23. Note that bit BA0 is not present on the bus. The data bus is 16 bits wide. BA0 determines whether the upper or lower byte of memory is desired for byte-oriented instructions. It is converted by the bus master into the appropriate upper and/or lower data strobe.

After allowing sufficient setup time for the address, the bus master asserts the buffered address strobe signal $\overline{BAS}$ to indicate that the address is now valid. All entities in the memory space then decode the address using $\overline{BAS}$ as an enable to determine if they contain the address being accessed. The addressed card responds by asserting the "I am addressed" signal $\overline{IMA}$ and holding the $\overline{DTACK}$ signal nonasserted while being accessed. The $\overline{IMA}$ signal is used by the bus expander logic to determine the location of the memory participant so that the data bus buffers for the expander interface can be turned on if required. Concurrently with asserting $\overline{BAS}$, the bus master also asserts the appropriate buffered upper and/or lower data strobes to indicate which bytes are involved in the data transfer. The addressed card can now enable its data bus buffer since it knows that it is to be involved in the memory cycle and that the data bus direction is correctly defined. For the write cycle, the buffered read/write signal is not defined until the data strobes occur. Therefore, if $\overline{BAS}$ alone is used to enable the data bus buffers, a bus conflict could result.

Once the addressed card has placed the data on the bus and allowed sufficient setup time, it indicates that the desired data is now valid on the bus by asserting $\overline{DTACK}$. The bus master detects the $\overline{DTACK}$ signal, latches the data at its convenience, and ends the cycle by negating $\overline{BAS}$ and the data strobes. The addressed card detects that the cycle has ended by detecting that $\overline{BAS}$ and/or the data strobes have gone away and releases the data bus and cancels the $\overline{IMA}$ and $\overline{DTACK}$ signals.

The I/O write cycle is similar. The address and $\overline{BAS}$ signals occur in the same way to start the cycle. All entities in the memory space decode the address and the addressed card then asserts $\overline{IMA}$ and holds $\overline{DTACK}$ in its nonasserted state. The bus master then asserts the buffered read/write signal to indicate that a write cycle is in process. Next it places the data to be written on the data bus, and after sufficient setup time, asserts the upper and/or lower data strobes to indicate that the data is valid. The I/O card detects that the data strobes have occurred and stores the data in the proper location. When the storage operation is complete, the card asserts $\overline{DTACK}$ to indicate that the cycle can end. The bus master detects that $\overline{DTACK}$ has occurred and removes $\overline{BAS}$ and the data strobe signals. The I/O card then cancels its drive of $\overline{IMA}$ and $\overline{DTACK}$.

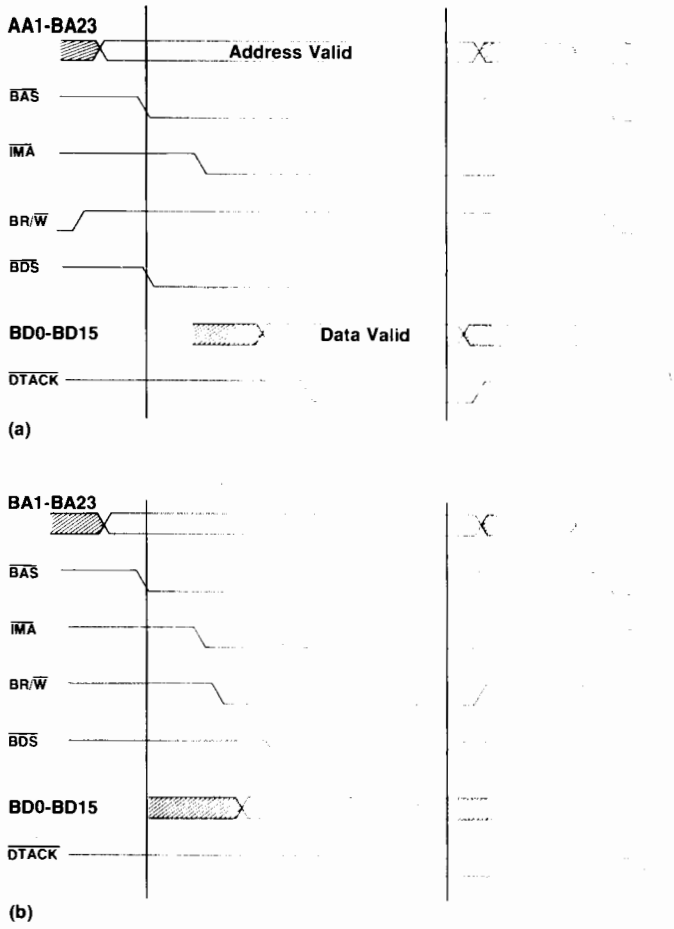


Fig. 3. *Read (a) and write (b) cycle timing for I/O and memory cards in the 9826A and 9836A.*

Thus, the programmed I/O protocol is essentially a version of the common asynchronous handshake scheme. The address and data strobes provide the sourcing information. The $\overline{DTACK}$ signal provides the response of the receiving entity. Since the I/O card may or may not be present, the bus cycles are provided with a timeout that automatically ends the bus cycle with a bus error trap should an I/O card not respond within a specific amount of time.

Interrupt I/O

The interrupt I/O scheme provides the ability to interrupt the normal instruction execution sequence of the mainframe CPU. The 68000 provides for seven priority levels of hardware interrupt. The levels have been distributed among the hardware as follows:

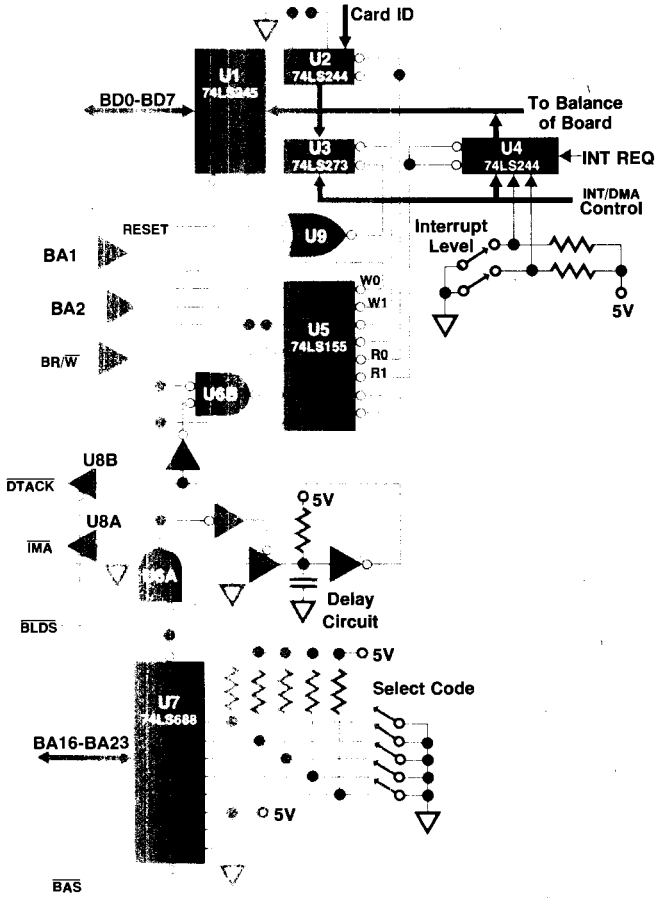


Fig. 4. *A typical 9826A/9836A I/O interface design.*

Interrupt Level	Device
0	Quiescent level
1	Keyboard and real-time clock
2	Internal flexible disc unit
3 through 6	External I/O
7	Reset key, powerfail

The internal peripherals have a lower priority than the external peripherals for two reasons. First, the human interface is felt to be secondary in priority to the user's being allowed full use of the mainframe CPU for control purposes. Second, the design of the 9826A and 9836A is such that the internal peripherals have drivers that can be interrupted without causing improper operation whereas the user may have an external device that requires fast interrupt response and whose service routine cannot be interrupted by internal peripherals. On the high-priority end, the highest interrupt level is reserved for the reset key so that the user has full control over the machine. Thus, the external I/O cards have four interrupt levels, more or less by default. This is twice the number of levels available in the 9825, and the external I/O cards do not have to share an interrupt level with the internal devices as required in the 9825. Also, the setting of the I/O card's interrupt level and select code has been divided to allow the user more flexibility in assigning the interrupt levels to the 32 available select codes.

DMA

DMA provides the mechanism for high-speed data transfers between memory and external devices. The 68000 does not contain DMA hardware as provided in the I/O controller chip of the 9825. A suitable LSI DMA controller was not available, so the choice was made to provide a discrete DMA controller in the form of an option card. The DMA card provides two channels of DMA capability with full address range per channel and a maximum buffer size of 64K bytes. The card provides a burst mode capability of greater than 1.2 million transfers (bytes or words) per second. Actual transfer rates are dependent on the performance of the particular I/O card involved in the transfer.

Hardware Design

To get an idea of the simplicity of the interface hardware, Fig. 4 shows a functional schematic of a typical I/O interface. The memory location of the card is defined by the Q side of address decoder U7. When $\overline{\text{BAS}}$ occurs the decoder is enabled. Its output enables the $\overline{\text{IMA}}$ and the $\overline{\text{DTACK}}$ drivers U8A and U8B and gate U6A. When the lower data strobe $\overline{\text{BLDS}}$ occurs, the output of gate U6A turns on the data bus buffer U1, enables the register decoder U5, and starts the timing delay. The register decoder U5 determines which register is involved in the cycle and whether a read or write operation is to occur. There are two standard registers that all I/O interfaces must provide. The first is a read-only register (buffer U2 in Fig. 4) which provides a unique identifier for the card. By reading this register, the operating system can determine which I/O cards are at which select codes. The second register, represented by U3 and output buffer U4, provides status and control information. A write to the register can enable or disable card interrupts. A read of the register indicates which interrupt level the card is on and whether or not the card is currently interrupting. Decoder U9 connects the card's interrupt request to the appropriate interrupt level. The second register also determines which DMA channel the card is assigned to. The DMA interface logic is not shown in Fig. 4. Only four low-power Schottky TTL (transistor-transistor logic) IC packages are needed to implement the interface for a simple DMA timing delay.

Loyd F. Nelson

Loyd Nelson earned the BS (1967) and MS (1969) degrees in electrical engineering from Colorado State University. After a few years designing air traffic controller displays, he joined HP in 1973. Loyd designed the architecture of the MFG/250 application package and wrote part of the code. He also designed the CPU and other boards used in the 9825 Computer and did production engineering for it. Loyd was born in Cushing, Oklahoma. He is married, has three daughters, and lives in Loveland, Colorado. He enjoys backpacking, jogging, bicycling, and sports of all kinds.

Low-Cost Printers for the 9826A and 9836A Computers

This family of compatibly packaged thermal printers provides quality hard copy of alphanumeric text and graphics displays.

by Michael J. Sproviero

LIKE ALL COMPUTING SYSTEMS, the 9826A and 9836A Computer Systems require a companion printer. The printer not only has to do alphanumeric printing at medium-performance speed and print quality, but also must be capable of duplicating graphics displays. Secondary needs dictate that the printer be easy to use and conveniently located when connected to the system. As an alternative to building the printer mechanism directly into the computer, a separate printer package was designed to sit on top of the 9826A or 9836A (Fig. 1).

The thermal print mechanism (also used in HP's 2620 series of display terminals) uses a printhead containing 15 thin-film resistors arranged in a vertical column. Characters or graphics are printed by moving the head across thermally sensitive paper and selectively activating the resistor elements at appropriate locations. Characters are formed within a high-resolution 18×15 dot matrix to produce character quality beyond that previously achieved by thermal printers. Raster graphics resolution is 90 dots per inch both vertically and horizontally and up to 720 dots are available across the page. Fifteen raster dot rows are buffered by the printer before each graphics print line scan, producing high-throughput graphics at 16,200 dots/s. Alphanumeric characters are printed bidirectionally at 120 characters/s. A microprocessor-controlled algorithm optimizes character throughput.

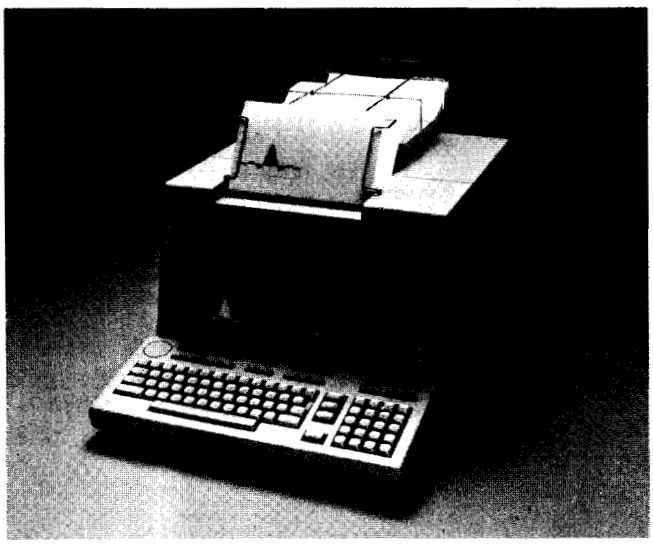


Fig. 1. *The HP Model 2673A Intelligent Graphics Printer is the top of the line of HP's 2670 series of thermal printers. Like the other printers in this family, it sits conveniently on top of a 9826A (as shown above) or 9836A Computer System to provide quality hardcopy of alphanumeric and/or graphics data.*

The print mechanism is controlled by a microcomputer located on an attached printed circuit board. The microcomputer accepts and buffers characters, commands, or graphics data over a simple 8-bit parallel interface. Data handshake lines are used for interrupt and acknowledgment of data transfers. Besides accepting input data, the microcomputer controls two step motors and the 15-element printhead. One motor controls movement of the printhead and the other controls paper advance. Phase information is maintained by the microcomputer for each motor and an on-board timer interrupt is used for speed control. A 64K-bit ROM stores the definitions of up to 256 characters. The microcomputer transforms the ASCII*-coded characters to dot information for the printhead. Graphics data is received in row format and transposed to column format for the printhead. To achieve 120 characters/s peak printing speed, the internal timer runs at 2160 Hz (460-μs period). With each timer interrupt, the microcomputer updates phase information for the step motor and outputs dot information to the printhead. The printhead resistors are driven from a 16V power supply and energized for 420 μs to print a dot of information.

Consecutive dots for a given resistor are energized for only 160 μs since the resistor is still at an elevated temperature from the previous dot. This technique maintains consistent intensity of dots and increases resistor life.

Thermal printing requires contact of the printhead and the paper. Appropriate pressure is maintained by a spring-loaded guide. To achieve accurate paper advances, the printhead is lifted during line feeds. This is done by a solenoid which depresses the paper guide spring, relieving head pressure and lifting the head slightly. A single line feed cycle lasts 66 ms. Mechanical time delay for the solenoid requires 16 ms, 15 motor phase advances take 33 ms, and 17 ms is allowed for the head to settle after the solenoid is deenergized. Consecutive line feeds are performed with the same solenoid cycle and therefore require only an additional 33 ms. A photocell is used to sense out-of-paper and top-of-form conditions. When using perforated paper, a small hole located in the perforation between pages is read by the photocell as the paper passes. The microcomputer automatically moves the paper past the perforation to the

*American Standard Code for Information Interchange

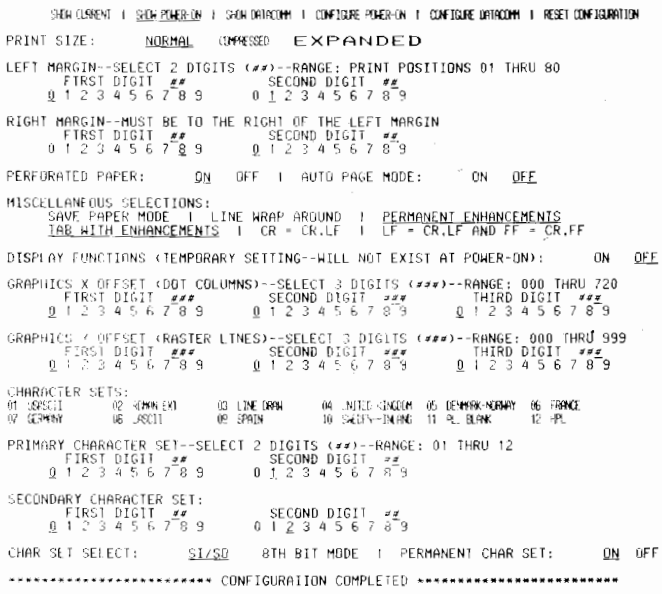


Fig. 2. *Configuration menu for the 2673A. This menu is printed by the 2673A in the configuration mode. Current settings are underlined. Each may be changed to any of the other settings shown by pressing the appropriate keys on the front panel of the 2673A or by program control.*

next top-of-form position.

Printer Design

Given a general-purpose thermal printing mechanism, the design problems involved overall package and product design, including a user control panel, a power supply system, a flexible external interface system, and overall control logic. The control logic is accomplished by a second microprocessor system whose primary function is interface control, data buffering, including raster graphics, escape-sequence control decoding, printer configuration control, and interfacing to the thermal print mechanism.

As mentioned earlier, the primary product design requirement was compatiblility with the 9826A. This includes not only industrial design style, but also environmental and regulatory agency approvals. To do this, sufficient margin is designed in to guarantee success in all system tests. Because the printer sits on top of the CRT display of the 9826A or 9836A, adequate shielding of magnetic devices such as the power transformer is required to prevent the printer from affecting display quality. Internal metallization of the plastic housing and special gasketing is used to shield the entire enclosure. The results are plenty of margin from electrostatic discharge faults and certification to FCC Level B for RFI emission with a 10-dB margin.

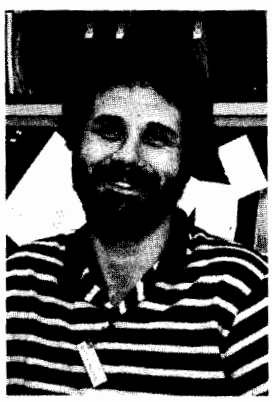
Printer Family

A family of printers, including the 2671A Alphanumeric Printer, 2671G Graphics Printer, and the 2673A Intelligent Graphics Printer, is available within this same package. At the top of the line is the 2673A (Fig. 1), a superset of all the features of the other members of the family. In addition to the normal print mode that prints 10 characters/in at 120 characters/s, alphanumeric characters may be printed at 16.2 characters/in (132 columns per line) or 5 characters/in (40 columns per line). Print enhancements, such as underlining, framing, and triple-pass bold printing are accessible in all three print modes. All combinations of these print styles can be intermixed in a single line, character-by-character, while still maintaining the high throughput of bidirectional printing. Graphics features include autocentering, windowing, and offsets. Standard page format features are also available such as margins, tabs, and vertical page formatting. Character set flexibility is an advantage of the 2673A. European language and general-purpose line-drawing character sets are available.

A key contribution of the 2673A is that features are accessible both programmatically through escape-sequence commands from the host controller and from the friendly front control panel. A menu-driven configuration mode (Fig. 2) is accessed by four keys on the front panel. The configuration mode controls and retains the settings in the printer's memory. A state-of-the-art electrically alterable read-only memory (EAROM) that can store 128 bytes of data is used to remember the state of the machine when the power is removed. This novel printer feature allows permanent user customization of the printer's power-on state. Features can be also temporarily invoked by escape-sequence commands from the host controller. Thus, features can be turned on and off without altering the printer's memory.

The 2673A has a multilevel self-test and self-diagnostic capability. Separate tests for print capability, data communications, memory, and I/O can be invoked from a menu that is presented when the self-test key is depressed. Any faults detected are printed, or if printing is inhibited by fault conditions, a hexidecimal LED (light-emitting diode) indicator located on the main processor board displays a code identifying the fault area. Memory failures can be reported to the chip level.

Interface flexibility is achieved not only in the 2673A, but also in the 2671A and 2671G. Each interface type has a custom printed circuit board that can be changed by the user. In addition to the HP-IB interface, which is standard, an RS-232-C serial interface and a general-purpose 8-bit parallel interface are also available for all printers in the family.



Michael J. Sproviero

Mike Sproviero was born in Gustine, California. He attended the University of Santa Clara, earning the BSEE degree in 1972, and the University of Idaho, earning an MSEE degree in 1978. Mike has been with HP since 1972 and has worked in production engineering and did development work on the 2631 Printer. He now is a section manager for thermal printers. Mike is married, has a son and daughter, and lives in Vancouver, Washington. He is interested in woodworking and has built three custom homes, including doing the landscaping.

Acknowledgments

Many people contributed to the design of the 2670 Series Printers. Don Bloyer and Bruce Yano did the product and industrial design. Mark Lund and Tom Pritchard designed the I/O PCAs and Mark Elworthy did the 2673A main processor assembly. Wally Thrash designed the power supplies. Mike Ard, Mitch Forcier, and Mike Warden did the firmware design for the 2673A, while Claude Nichols, Brodie Keast, Brooke Winter and Bob Jones contributed various parts of the firmware to the 2671A/G. John Ignoffo, Joe Barbera and John Widder contributed design improvements for the print mechanism. Bill Huseby, Ken Williams, Dave Hansen, and Dave Lee provided technical problem-solving support in later stages of the project. Ba Nguyen did extensive software compatibility checkout. Jim Goecks and Scott Bonnet were the reliability and safety engineers.

The 9826A/9836A Language Systems

by Kathryn Y. Kwinn, Robert M. Hallissy, and Roger E. Ison

IN EVALUATING A COMPUTER'S software, customers are interested in how it can help them. Designers want to know not only what it does but how it does it. This article outlines the capabilities of the three language systems (BASIC, HPL,* and Pascal) available for the HP Model 9826A and 9836A Computer Systems, explains some of the technical challenges that arose in providing these capabilities, and describes how those challenges were met.

BASIC

The BASIC system used in the 9826A and 9836A is a rich and powerful language compared to most implementations of BASIC. It is patterned after the system used in HP's 9835 and 9845 Desktop Computers, but it uses a different enhancement strategy. In the 9835 and 9845, the mainframe language provides many general-purpose computing facilities and is enhanced through the later addition of ROMs (read-only memories) to provide selected capabilities such as device I/O (input/output), graphics, and structured programming. In the 9826A and 9836A the goal is to provide in the initial product all the capabilities required to solve a wide variety of instrument control and CAD (computer-aided design) problems, and to allow incorporation of new enhancements from time to time. Hence, there are substantial I/O, real-time clock, and graphics capabilities in the initial BASIC system.

This version also provides a number of features that have not appeared previously on HP desktop computers. The ON KNOB statement can be used to define actions to be taken when the knob (rotary pulse generator) is rotated. Labeled COMmon blocks, which give greater data-sharing flexibility, can be defined. The ALLOCATE and DEALLOCATE statements provide simple (stack discipline) run-time memory management. Subprograms can have optional parameters. Subprogram libraries can be kept in mass storage and loaded quickly when needed. Functional HP-IB** noncontroller capabilities are also given more emphasis.

Customers are interested in execution speed as well as language capabilities. They have indicated that for their instrument control and computer-aided design problems of the 1980s, they want twice the execution speed of the 9825. The 9826A/9836A BASIC system meets this need, and for some applications, exceeds it. For example, some I/O operations execute six times as fast as their 9825 counterparts.

This version of BASIC is not only powerful, it is also easy to use as exemplified by its human interface. The knob in the upper lefthand corner of the keyboard can be used in the program-edit mode to provide rapid scrolling through the program text, an almost addictive feature. As with earlier desktop computers, the 9826A and 9836A have softkeys. However, they have the additional capability to associate labels with ten of the softkeys and to display them on a reserved area of the CRT (cathode-ray tube) display while a program is executing. This feature can be used to provide an operator interface in a menu-driven program. Another addition to the human interface becomes apparent when the system detects an error, either in parsing or executing a program. The error message, instead of just the error number, is displayed on the CRT.

The friendliness and ease of use of 9826A/9836A BASIC is not limited to its human interface, but extends through the rest of the language as well. Various enhancements to 9835/9845 BASIC that have proved very useful, such as the IF...THEN...ELSE...END IF construct, have been incorporated in the first release of 9826A/9836A BASIC. Changes were made to 9835/9845 BASIC to make it easier to use. The most radical changes appear in the I/O sublanguage.

However, making changes to 9835/9845 BASIC was difficult because of the concern for the investment the owners of these systems have made in software development. To help them convert their current software to 9826A/9836A BASIC, a translation program is available which can do most of the job automatically. It prompts the user for help when needed, and also reports complicated situations that need to be reworked by hand.

*The language used in the HP 9825 Desktop Computer.

**Hewlett-Packard's implementation of IEEE Standard 488 (1978).

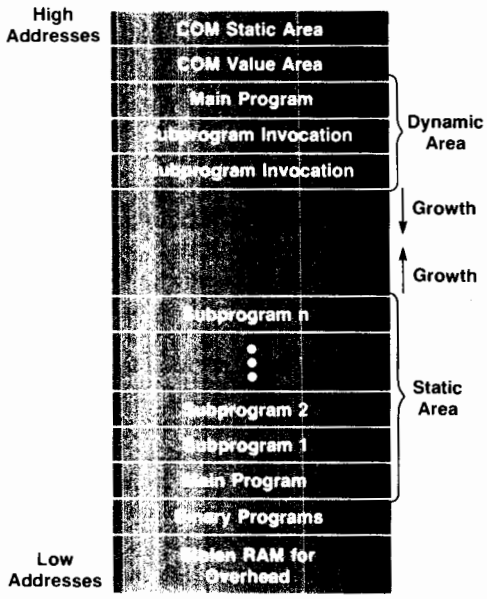


Fig. 1. *Map of user RAM for 9826A BASIC language system.*

BASIC: I/O Sublanguage

From our customers we learned that for instrument control and testing applications, they favor the 9825's direct approach to I/O over the implicitly buffered approach used by the 9835 and 9845 Desktop Computers, so we adapted 9825-style I/O to BASIC syntax. That is, all 9826A/9836A BASIC device I/O is under the direct control of the programmer instead of being buffered at the system's discretion. In addition, the I/O language is unified so that the same keywords are used to communicate to devices and files.

The "glue" that unifies the I/O sublanguage is the ASSIGN statement. An ASSIGN statement associates a name and a set of attributes with an I/O path. (An example of an attribute is FORMAT, whose value specifies whether the data transmitted is assumed to be encoded in the machine's internal representation or in ASCII.*) For greater flexibility, this association is formed at statement execution time, not at program prerun. Variations of the ASSIGN statement allow the programmer to change the attributes associated with a defined I/O path or to cancel the association.

The resulting I/O sublanguage is much simpler to use because of the ASSIGN statement. Only two keywords, OUTPUT and ENTER, are needed to send and receive data. This is the case regardless of whether the interaction is with a file or an internal or external device. In the 9835 and 9845, only select codes could vary their attributes. In the 9826A and 9836A, attributes can be associated with files as well. Another big advantage of the unified I/O is that it is an extremely simple matter to redirect I/O from one place to another. Only the ASSIGN statement must be changed to specify the new object and its attributes. Indeed, subprograms can be written to do I/O without knowing what kind of I/O paths they are using. Still another advantage to this system is its flexibility. New types of I/O paths can be supported very easily by extensions to the language. The programmer will only need to learn how to define I/O path names for the new types in ASSIGN statements. ENTER and OUTPUT will still be the keywords normally used to do the I/O.

*American Standard Code for Information Interchange

Development of 9826A/9836A BASIC

Many of the technical challenges that arose as we were implementing 9826A/9836A BASIC resulted from the goal of making it run twice as fast as HPL on the 9825. The 68000 used as the CPU (central processing unit) runs at 8 MHz, somewhat faster than the 6 MHz achieved by the 9825's CPU, but not sufficiently faster to give the desired speed improvement. To complicate matters, 9826A/9836A BASIC is a much more involved language than HPL. HPL can use a fixed-length symbol table because only single-character variable names are permitted. In 9826A/9836A BASIC, because variable names may be up to 15 characters long, a variable-length symbol table is required. This means that operand fetches are more complicated. In addition, BASIC subprograms have separate environments, whereas HPL subprograms share the main program's environment. The result is that BASIC programs incur significantly more overhead whenever a context switch is required. An additional slowdown results from the fact that approximately 80% of the BASIC system was coded in Modcal.** Assembly language was used only for speed-critical areas such as I/O, graphics, and arithmetic and for very low-level operations for which Modcal was inadequate. Modcal gave us many advantages in reducing the time and effort required to develop and debug the system, but as expected, the code produced by compiling Modcal is not optimal.

The most obvious way to obtain speed would have been to write a BASIC compiler rather than an interpreter. However, several factors favored an interpreter. Interpreters are generally easier to write and debug, and the code produced by an interpreter is usually much more compact than compiled code. Thus a customer can solve the same problems with less memory, saving the cost of additional memory. Furthermore, our customers have come to expect a friendly, easy-to-use program development environment which an interpreter can provide. We felt that an interpreter would better serve their needs, so we set about producing a fast, friendly interpreter.

The fact that the 9826A and 9836A have a large, linear memory was an enormous help to us in making the systems meet their speed goal. It meant that we could assume that there would be sufficient memory for program execution without resorting to block-switching or other memory overlay techniques. Fig. 1 shows a simplified map of user memory. A static area for COMmon occupies the high end and just below this area is the value area for the COM variables. Next comes the main program's dynamic area, followed by a dynamic area for each invocation of every active subprogram. These dynamic areas form a stack that grows toward the low end of memory. At the low end of memory is space reserved by the system software intermixed with binary programs. Just above this is the static area for the main program followed by a stack of similar static areas for each subprogram in the system. This stack grows toward the high end of memory.

**Modcal, developed by HP's Desktop Computer Division for internal use, is an extended form of Pascal.

Each static area for the main program or a subprogram consists of a header containing information about the associated program block and the layout of the rest of the static area. It also contains a symbol table, a token table, the internal code, and a dimension table. The static area for COM is similar, but contains no internal code. The static areas are built at program entry and program prerun time. When the **ENTER** key is pressed to signify the end of a statement, the parser parses the line, builds and partially initializes the symbol table, builds the token table, and emits internal code in reverse Polish notation (RPN).

The internal code was designed with both execution speed and language expandability in mind. It is sequential for simplicity and compactness. Since it contains only relative addresses, moving the internal code when it is necessary to expand the symbol table during program entry does not slow the system unduly. All references to program variables in the internal code are given as indexes into the appropriate symbol table. This helps execution speed by eliminating symbol table searches. The system does not retain the actual source line, but can recreate it by means of a backlister. When the **RUN** key is pressed or the RUN command is executed, the system performs a prerun on the entire program. During this prerun phase, each static area is completed. The header and symbol table are filled in, and a dimension table containing the declared dimensions of every array is built. The COM value area is also created, the structured programming statements are statically matched (e.g., each FOR statement is associated with exactly one NEXT statement), and the internal code is modified to include the branching addresses associated with these constructs. After the prerun, the system uses a table-driven interpreter to execute the internal code.

When execution begins, a dynamic area for the main program is created immediately below the COM value area, and the interpreter begins executing the main program's internal code. Another dynamic area is created on top of the dynamic stack each time a subprogram is invoked. Each dynamic area contains a complete current environment for either the main program or a subprogram invocation (9826A/9836A BASIC is recursive) and includes a call block, pointers to parameters, a copy of the dimension table, the execution header, storage for the values associated with FOR loops, and the value area for statically and implicitly declared local variables. The size of this portion of the dynamic area is known at subprogram invocation. It is pushed onto the dynamic stack and parts of it are initialized at this time. The dynamic area also contains storage for variables created in ALLOCATE statements, for the control blocks associated with event-initiated branching statements (ONs), and for a GOSUB return stack. It also has room for temporary structures used during statement execution and for expression evaluation. These areas are created and destroyed as necessary during execution, but always according to a strict stack management discipline.

Certainly some of the data structures and operations described above sound more complicated than necessary. The reason for introducing these complications is to obtain greater execution speed. The speed penalties associated with a variable-length symbol table are partially overcome by making each symbol table entry the same length. Thus each symbol table is an array of records and can be accessed efficiently by the interpreter. To make fixed-length symbol table entries possible neither the identifier names nor their values are kept in this table. The names are stored in a separate token table in the static area, and each symbol table entry contains an index into the token table. Token table entries are variable in length, and hence the token table must be searched whenever it is accessed. However, the token table is used only during parsing and when it is necessary to recreate the program source (e.g., for listing the program), so it is not important that access be fast. The values are stored in the dynamic area at the high end of memory, just below the COM static area. Values are represented in the symbol table by an offset from the base address of the dynamic area. These offsets are determined and stored in the symbol table during program prerun, along with the size of the value area. Hence, at subprogram invocation time, the value area can be pushed onto the dynamic stack very quickly.

Several language restrictions were necessary to implement the system as described. It is not possible to edit a program without stopping it. Furthermore, statements that are to be executed directly (i.e., are not part of a program) may not introduce new variables. SUBroutine and DEFine FunctioN statements can only be appended to the end of the program, not inserted randomly in the code. Similarly, to delete a SUB or DEF FN statement, the entire subprogram must be deleted in the same operation. Declaratory statements such as INTEGER are processed at program prerun and hence must use constants rather than numeric expressions to specify array bounds and string lengths. ALLOCATE, on the other hand, is processed during program execution and may contain expressions. Because structured programming statements are statically matched at prerun, programs that attempt to abuse these structures (e.g., by branching back and forth between FOR loops) will not produce the expected results.

Execution speed was also enhanced by doing binary arithmetic instead of binary-coded-decimal (BCD). The 9825 uses BCD representations of numeric data, but the 9826A and 9836A use two's-complement representation for integers and the proposed IEEE 64-bit floating-point standard format for real numbers. In this way, we are able to take advantage of the 68000 microprocessor's built-in instructions for integer arithmetic. Using binary arithmetic has several advantages in addition to speed. First, binary computations lend themselves to easier error analysis. In addition, binary representation is much more compact so that for the same amount of storage space—64 bits per real number—we are able to get 15 decimal digits of precision versus 12 digits in the 9825 and are also able to support a much larger range of exponents.

Another way in which the 9826A/9836A BASIC system met its speed goal was reducing the interpreter's end-of-line overhead. There are many conditions that must be tested after each line is executed (e.g., is trace output required, is there any event-initiated branch that should be serviced, and is there a statement waiting for direct execution?). In all, some twenty or more tests can be performed at the end of every line. Rather than perform all of these tests for each line, we introduced a master bit that is set

whenever any of these conditions becomes true. The master bit is always checked at the end of a line. If it is false (which is normally the case), no further tests need to be performed. Only if it is true does the system do exhaustive testing to determine the required actions.

The large memory in the 9826A and 9836A allows the interpreter to run faster because stack overflow is highly unlikely. A block of memory that is large enough to allow the evaluation of a worst-case normal numeric expression is reserved at the end of the static area. The dynamic stack is not allowed to grow into this area unless the system is evaluating an expression. Operations such as subprogram invocation that require a large block on the dynamic stack do an overflow check, but most operations do not need to make this test.

Because the 9826A and 9836A are designed to be HP's premier HP-IB instrument controllers, I/O speed is of paramount importance. For this reason, most of the I/O system was written in assembly language. The strategy used by the I/O system is much the same as the strategy used to expedite subprogram invocation, namely doing as many operations as possible before the actual data transfer begins. The speed optimization begins at system power-up. At this time, the system initializes a select code table that records such information as whether or not an interface is in use, the type of interface, and the location of the appropriate I/O drivers. Individual I/O statements consult this table rather than determine the necessary information on their own.

Each I/O operation is divided into three phases: initialization, data transfer, and termination. The initialization phase increases speed by minimizing the per-byte overhead associated with data transfer. During initialization, the interpreter determines the source or destination of the data transfer, what I/O driver is to be used, and whether the data is to be encoded in internal form or in ASCII. This information is stored away in a temporary table on the dynamic stack where the system can refer to it for the duration of the statement's execution. Certain procedure variables (variables whose values are procedure names) that will be used to accomplish the data transfer get values at this time. The I/O driver also checks validity during initialization. For device I/O, the driver determines that the specified interface does indeed exist and that the source or destination address is valid and then addresses the bus. For files, the driver locates the place on the storage medium where the operation is to begin.

BASIC Enhancements

Language enhancements can take on two forms: adding totally new statements or adding new syntax and semantics to existing keywords. Either kind of enhancement can be difficult to implement unless the system has been designed for expandability.

Part of the solution to the expandability problem involves both the parsing strategy and the design of the internal code generated by the parser. 9826A/9836A BASIC parses a program statement whenever the **ENTER** key is pressed. It also parses statements entered from the keyboard when the **EXECUTE** key is pressed. In either case, if there are syntax errors, they are reported and no further action is taken. Of course, a given line might be erroneous if some enhancement is absent, but perfectly correct if that enhancement is part of the system. To allow such lines to be parsed properly, we adopted the following strategy. The initial 9826A/9836A BASIC system has its own syntax module, and so does each of the enhancements. The parser for the initial model of the 9826A or 9836A always begins the parsing process. If it does not recognize a token, it does not report an error immediately, but first polls each enhancement's parser to see if any of them can handle the token in question. If any of the enhancements does recognize the mysterious token, it parses it, generates the appropriate escape sequence and internal code, and then returns control to the main parser. Only when none of the parsers can deal with the token is a syntax error generated.

To understand how the enhancements are processed at execution time, it is necessary to understand the design of the RPN internal code generated by the parser. At execution time, this internal code is processed by a table-driven interpreter. Each operand is represented by an index into the appropriate symbol table array. (Op codes 1 through 127 are reserved for operands. The first one hundred are simple indexes. The remainder are used only as the first byte of a two-byte encoding of a symbol table index. Using this scheme, each symbol table can accommodate approximately 7000 entries.) Each operator is represented by one to three bytes. To maximize both execution speed and code compactness, single-byte op codes are assigned to the more frequently used operators and two-byte op codes to the less frequently used operators. A single-byte op code is an index into the primary interpreter table. The first byte of a two-byte op code sequence identifies a secondary interpreter table, and the second byte is an index into that table. A three-byte op code is always associated with an operator supported by a language enhancement. The first byte identifies the op code as a three-byte sequence, the second byte identifies the enhancement that supports the operator, and the third byte is an index into the enhancement's interpreter table.

The scheme outlined above solves many, but not all, of the language enhancement problems. It works well for adding totally new statements to the language, but it often breaks down when an existing statement is enhanced. As an example, consider file I/O. The only mass memory device supported by BASIC in the initial model of the 9826A is the internal flexible disc drive, but the first set of enhancements allows use of the HP 9895 flexible disc drive as well. The process of writing data to a file on the 9895 is totally different from writing to a file on the 9826A's internal flexible disc unit, yet the programmer uses the same statement for either operation. In the past, problems like this were often solved by making the enhancement ROM support both the old and the new capability. This strategy is fraught with problems, not the least of which is that after several enhancements the system may contain multiple versions of code to do (supposedly) the same thing, and all but one version will be ignored. We were determined to avoid this situation whenever possible. To do so, we devised an elaborate system of "hooks." In many cases, hooks consist of procedure variables in the implementation code. The action taken at a given time depends on the procedure name that is the value of the variable when it is encountered. Such hooks

are incorporated in the system everywhere extensions can be reasonably anticipated. So far, they have proved very effective in eliminating code duplication.

Since the 9826A and 9836A put such a strong emphasis on I/O, this is one area in which many enhancements were anticipated. The hooks in the I/O implementation are thus extremely important. As mentioned in the discussion of I/O speed, each I/O operation is divided into three phases: initialization, data transfer, and termination. Each of these phases is further divided into an object-independent and an object-dependent (I/O driver) portion. The object-independent code performs all manipulations that are independent of the source or destination of the I/O. Among these are data conversion and formatting. The object-dependent code performs the rest of the operation and is tailored to a particular source or destination. There is a standard interface between the object-independent and object-dependent routines. By using procedure-variable hooks at this junction, it is relatively simple to support new hardware interfaces by adding only I/O drivers. The beauty of this approach is that such additions are transparent to the programmer.

HPL

The use of HPL began in February of 1972 with the introduction of HP's 9820 Programmable Calculator.[1] By the time the 9825A[2] was introduced in January 1976, HPL had evolved into a sophisticated programming and I/O control language. Today, there are over 28,000 9825 systems installed worldwide, proving that HPL has made a home for itself.

While the 9825 and its language have become popular with HP's customers, HPL is not a current-technology language: there is no real structured programming capability. Variables are single letters, not multicharacter. It is cryptic and often hard to read. Parameters and local variables for functions and subroutines are limited to simple numerics. So one might justifiably ask the question, "Why would Hewlett-Packard implement HPL on another machine when BASIC and Pascal are better choices?"

The answer lies in the recognition that the cost of software is continually increasing while the price/performance ratio for hardware has been plunging downward. Computer users are looking for ways to increase performance by moving to new hardware without having to discard their previous investment in software.

The overriding goal of the HPL project for the 9826A and 9836A was to provide a way for 9825 users to run their existing software on new-technology hardware. The key strategies for achieving this goal were to maintain a similar internal architecture by translating 9825 system code to 68000 code and to place 9825 compatibility as the highest priority of that translation process. As will be seen shortly, even some performance is sacrificed to maintain compatibility.

The second objective of the project was to add language enhancements that would either complement the new hardware available in the 9826A and 9836A or fill in known deficiencies in the language. The knob, alphanumeric and graphic CRT displays, large read/write memory, built-in flexible disc drive, programmable beeper, powerfail, and real-time clock are examples of hardware features for which HPL language extensions were desired.

Development of 9826A/9836A HPL

The design team had to address four major challenges to meet the objectives: human interface, mass storage, I/O, and mathematics.

In the area of human interface, much of the 9825's editing and operator I/O was designed around a 32-character, single-line display and a 16-character strip printer. One of the major enhancements of 9826A/9836A HPL is the implementation of multiline editing and a scrolling "soft printer" on the CRT. For the scrolling printer, the number of pages of information that can be maintained off-screen can be programmed from a minimum of two pages to as many as the available read/write memory can hold. Also, the 9825 definition of special-function keys is extended on the 9826A and 9836A to allow labels to be displayed on the CRT and to allow the keys to be individually definable from a running program.

Another important hardware feature available on the 9826A and 9836A is CRT graphics. The HPL system has been extended so that plotter commands that on the 9825 were always directed to an external plotter can now be directed to the graphics screen. Some new statements necessary for CRT graphics were also implemented.

The mass storage system is an area where creative solutions have allowed the 9826A and 9836A to be compatible with the 9825, yet still provide a real performance contribution. The 9825 has two distinct mass storage systems: tape and disc. The internal tape cartridge uses a nondirectory, file-by-number system and is accessed by a set of statements independent of the directory-based, file-by-name disc system. It was necessary to implement both systems, but how could it be done on only one device? The disc system was the most natural fit with the built-in flexible disc drive, but what could be done with the tape statements?

The solution to the dilemma was simultaneously mapping tape file numbers into disc file names and tape statements to disc statements. Consider, for example, a tape statement such as rcf 5 which would be used to record the current program onto tape file 5 of the current tape track, say track 1. In the 9826A and 9836A, such a statement will cause the program to be saved onto disc file T1F005. Thus all actions dictated by tape statements are emulated on the disc.

Another key feature of the 9826A/9836A HPL mass storage system is the unified approach to all devices. The same tape or disc statements can be used to access several devices (internal flexible disc, or external 9885, 9895, 9134, 82901, 82902). The 9134, 9885, and 9895 support two different formats: one compatible with the 9825, 9835, and 9845 computers and the other, HP LIF (logical interchange format), compatible with 9826A/9836A BASIC and Pascal.

One of the major uses of the 9825 is as an I/O controller, and a majority of HPL software uses direct I/O card register access. However, the I/O interfaces of the 9826A and 9836A are of completely new design and the card registers R4 through R7 no longer exist. To provide compatibility with existing HPL programs, the 9826A/9836A HPL system provides a software emulation of the 9825 I/O cards. Internally,

BCD Arithmetic on the 68000

The main objectives in writing the 9826A/9836A HPL arithmetic routines were 100% compatibility with the 9825 and performance that met or exceeded that of the 9825. Because of major architectural differences between the two processors, a switch to binary arithmetic was considered early in the project. The BPC (binary processor chip) used in the 9825 has two floating-point accumulators and ten instructions for manipulating 12-digit BCD numbers.[1] On the other hand, the 68000 CPU has only three BCD arithmetic instructions (add, subtract, and ten's complement), and these operate on only two BCD digits at a time.

Most of the HPL language system was translated from BPC assembly language to 68000 assembly language. However, this was impractical for the arithmetic routines because of the high use of custom BPC math instructions. As a result, the arithmetic routines were written from scratch using the 9825 only as a definition of what each operation should do. Speed was obtained by applying a number of programming techniques:

1. Heavy use was made of in-line code instead of subroutines and loops.
2. Conditional branches were organized to minimize the execution time of the most commonly taken path.
3. Instructions and addressing modes were carefully selected to minimize execution time.
4. The multiply and divide operations were implemented using base 10000. With this technique, BCD numbers are converted to base 10000, the base 10000 numbers are multiplied or divided using the binary multiply and divide instructions of the 68000, and the result is converted back to BCD.
5. Execution times for multiply and divide operations were further reduced by providing special routines for handling numbers with fewer than five significant digits.

Each of these techniques proved useful in reducing the execution time of arithmetic operations. However, the resulting 68000 code is several times larger than its BPC counterpart—a small price to pay for the added performance.

Of special interest is the use of base 10000 to speed up the multiply and divide operations. This technique works as follows. First, the 12-digit BCD mantissa is converted to a three-digit base-10000 mantissa (see Fig. 1). Each base-10000 digit represents four BCD digits, and is stored as a 16-bit binary number. Next, the binary multiply and divide instructions of the 68000 are used to perform a multiple-precision operation on the three base-10000 digits. The base-10000 result is converted back to BCD, after which rounding and normalization operations are performed. The conversion to and from base 10000 involves a combination of table lookup and arithmetic operations. Note that the base-10000 representation is only an intermediate step in the floating-point multiply and divide operations. Floating-point numbers on the 9826A/9836A HPL language system are stored in the same BCD format as in the 9825.

It may seem that the multiply and divide operations are fairly complex, and they are. However, the execution time for the entire operation (including conversion to and from base 10000) is one-half to one-third that of the 9825. Because a similar technique could not be found for the floating-point addition and subtraction routines, these operations execute at approximately the same speed as their 9825 counterparts.

Once the arithmetic routines were complete, their operation had to be verified against the 9825 for 100% compatibility. This was done by connecting a 9825 and a 9826A back-to-back via the HP-IB and running millions of operands through the routines and comparing the results. The final routines ran for several days without finding any discrepancies. On the 9826A, even the random number function produces exactly the same values as the 9825.

-Andy Goris

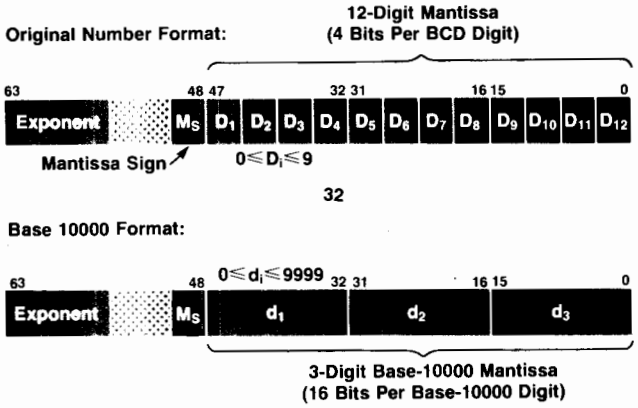


Fig. 1. *Conversion of BCD number format to base-10000 format.*

Reference

1. W.D. Eads and D.S. Maitland, "High-Performance NMOS LSI Processor," Hewlett-Packard Journal, Vol. 27, no. 10, June 1976.

each 9826A/9836A I/O card driver maintains sufficient state information to construct responses to card register accesses that will match what a 9825 system would have obtained.

The fourth major challenge area for the project was the mathematical calculations. The 9825 system uses BCD (binary coded decimal) representations of numbers along with special processor instructions to operate on BCD values. The 68000 CPU lends itself to binary representations better than to BCD, and has very little processor support for BCD operations at all. Since compatibility was the number one objective, BCD representation of numbers has been maintained in the 9826A/9836A HPL system. However, a special base-10000 mathematics package is implemented internally to take advantage of the multiply and divide instructions of the 68000 CPU (which operate on binary numbers). See the box on this page for more details on this conversion to base-10000 arithmetic.

If compatibility with the 9825 was the primary objective, it is reasonable to measure the success of the project by how easy it is to run 9825 programs on a 9826A or 9836A. The results of running many real applications indicate that many 9825 HPL programs run without modification. There are, however, certain 9825 machine characteristics that were impossible to duplicate on the 9826A and 9836A, and so there will be some programs that require minor changes. The incompatibility is confined to the following areas:

- I/O cards—some low-level control could not be emulated exactly because of new hardware design.
- Binary and memory files—because of the new processor and changes in internal data structures, files containing

binary programs and memory images are not compatible

- Memory use—some data structures now require more read/write memory. However, this is rarely a problem since the basic 9826A or 9836A Computer System has more memory than a 9825.
- Real-time clock (RTC)—the 9826A and 9836A support special mnemonics to access the internal RTC instead of the I/O statements used on the 9825 to access an external RTC.
- Keycodes—the hardware keycodes on the 9825 and the 9826A and 9836A are different, although a simple subroutine (included with the system) translates the new codes into the old codes for programs that expect the latter.

Unlike the incompatibility list, the enhancement list is quite long. Some of the enhancements are new uses of existing mnemonics while others are new mnemonics.

- CRT/Graphics: multiline editor, independent enable of the display of alphanumerics (text), key labels, and graphics, separate display areas for program and keyboard-executed display statements, full graphics support including binary plot, pointer, and graphics load/store, alphanumeric and graphics dump, and read/write alphanumeric text from any area of the display.
- Keyboard: full international keyboard support, program interrupt from knob, programmatic definition of softkeys and their labels, ability to push any key from a program or softkey, and ten-deep recall stack.
- Real-time clock: time-of-day clock and program interrupts based on time match, time delay, or cycle.
- I/O: read and write access to buffer pointers, ability to abort an active transfer, ability to set EOI (end or identify) on the last byte of an HP-IB transfer, access to two channels of DMA (direct memory access) and use of DMA with HP-IB, "printer is" command for setting the default printer device, and cancelling of on-interrupt branches.
- Mass storage: tape emulation on disc, "mass storage is" command for setting the default mass storage device, and unified mass storage commands.
- Powerfail (if installed; this is a hardware battery backup option to be available in the near future): a program interrupt can be enabled in the event that power is removed for more than a specified time, allowing for checkpointing or other corrective action.
- Miscellaneous: 32 user flags, string/numeric conversion to any base, read, data, and restore operations for program constants, and access to the programmable beeper.

Pascal

Pascal on the 9826A and 9836A combines the appeal of a modern, well structured and widely accepted programming language with the computing power of compiled code and a state-of-the-art microprocessor. The product is both an easy-to-use Pascal programming environment and a complete system development toolbox for the 9826A and 9836A. Unlike previous HP desktop computer products, this software system gives programmers direct and complete access to all the hardware resources built into the 9826A and 9836A mainframes.

Earlier in this article it was mentioned that the BASIC language system is largely written in Modcal, a programming language used internally at HP's Desktop Computer Division. Modcal consists of the HP standard Pascal language and some extensions for system programming. Many of the extensions were derived from the Modula language designed by N. Wirth.[3]

The software tools used to develop BASIC were also written in Modcal, and although they did not originally run on the 9826A, they were adapted to it for improved performance when prototype hardware became available. This process involved compiling the programs into native 68000 code, and designing and implementing an I/O subsystem. The 9826A became its own development environment, with a compiler producing native code, an assembler, a linker, and some other utilities.

These tools have been developed into production-quality software offering these features:

- The user-friendliness HP's products are known for
- A modern, highly structured programming language
- Assembly language for people who want it
- Complete access to all hardware resources with minimum system interference
- Very fast program turnaround, with immediate execution after compiling
- Execution speeds close to the limits of hardware performance
- Demand linking and loading (no explicit user action needed)
- Effective use and management of compiled code module libraries
- Debugging to the register level if necessary
- Smooth transition between normal execution and debugging modes
- Full use of the 16-megabyte address space of the 68000 CPU, including use of memory as a very-high-performance mass storage medium
- System easily tailored to have a new personality defined by the programmer.

Unlike BASIC and HPL, in which the entire language system is always resident in read/write or read-only memory, the 9826A/9836A Pascal environment is broken into subsystems which are loaded from mass storage when needed. The most important components are:

- Linking loader and file system
- Command interpreter
- Debugger
- HP standard Pascal compiler
- 68000 assembler
- File manager
- Screen-oriented editor
- Librarian

The system presents itself as a hierarchy, with the command interpreter at the top and the compiler, editor, file manager, assembler, and librarian as subsystems. At each level in the hierarchy the user is prompted with a menu of commands. A command is typically a single keystroke. For instance, at the highest level the system prompts:

Command: Cmplr Edit File Init Libr Run Xcut Ver

Typing C invokes the compiler, E or EDIT calls up the editor, F brings in the file manager, and so on. Each subsys-

tem in turn always prompts with its own list of capabilities.

The program development sequence is very simple. The editor is used to build or modify a source program, which is saved in a file in memory or on mass storage. The user presses the **RUN** key. If the compiler notices a syntax error, the editor can be invoked; it will place the cursor at the point where the error was detected. If there are no syntax errors, the program will immediately be loaded and begin execution. Subsequent RUN commands will not recompile the program unless the editor has been used to modify it.

The editor is an in-memory, screen-oriented design (no line numbers). It can edit as much text as the computer can store in its memory. It has complete search-and-replace capability, and can run in either program mode or documentation mode. Portions of text can be moved or copied. Full, smooth scrolling access to the entire edit file is provided by the knob on the keyboard.

The compiler normally accepts HP standard Pascal, conforming to the corporate standard. It can also restrict programs to ISO Pascal, reporting any violations of the proposed international language standard. It can also on request accept most of the language extensions of UCSD Pascal (a trademark of the Regents of the University of California), so programs written in that language will transport easily. Finally, it can be told to accept eight system programming extensions chosen from DCD's internal Modcal language. These extensions are necessary to provide error trapping and full access to the 9826A/9836A hardware set.

Compilation is a one-pass process which translates programs into an internal tree representation from which code is generated. The output is relocatable object code, ready to run. Compilation speeds exceed 4000 lines per minute when the source and object files are stored in memory.

The two-pass assembler accepts 68000 instruction mnemonics in the syntax defined by the manufacturer, and produces object code in the same format as the compiler. Either absolute or relocatable assemblies are possible. Assembly speeds are up to 4000 lines per minute.

The debugger allows stepping through an executing program one line at a time, or one machine instruction at a time. As the program executes, the line numbers can be displayed at the lower right corner of the screen. A history queue can be kept of the last 63 line numbers or instruction addresses executed, so it's possible to see how the program arrived at the point where an error happened. Errors can be trapped before the program is aborted, and the entire state of the CPU registers as well as all of memory is accessible at any time while a program runs. The debugger maintains its own screen and human interface, so the normal user screen content is not disturbed by debugging action.

Up to 50 logical devices can be on-line at once. Many logical devices and their directories can be on a single large mass storage device. The file manager can create and examine directories, copy, purge, or rename files, and back up entire logical devices.

9826A/9836A Pascal defines a method by which incomplete portions of a program, called modules, may be compiled without reference to any program that may ultimately use them. A module, even after being compiled, carries with it the precise specification of how it must be interfaced to any other program or module.

A library is simply a file containing programs or modules of code. The output of the compiler, assembler, and librarian is always a library. A program using precompiled modules simply specifies that they are to be imported from some library; thereafter, the compiler and linking mechanisms automatically bind the module into the program. Libraries may be resident in memory or found in mass storage.

The librarian provides three library management capabilities: linking, constructing libraries, and disassembling libraries. Ordinarily, linking occurs without intervention when a program is run. The librarian's explicit linking function is mainly for use in generating an entirely new system. Constructing a library is simply the process of collecting modules of code into a group so they can be conveniently managed together. To disassemble a library is to examine any or all of its modules in detail—the interface specifications, linkage information, and even machine code can be displayed on demand.

A standard library is supplied with the system. It contains modules with about 150 procedures to perform device I/O, interactive graphics, and disc file interchange with many other HP computer systems. These library routines can be used by any Pascal application program.

Direct I/O to peripheral devices is supported in four different transfer modes over HP-IB, RS-232-C, and 16-bit parallel interfaces. Fast handshake transfers are done in a very tight loop, with the 68000 CPU devoting full attention to processing each byte or word of data as quickly as possible. Interrupt mode allows the CPU to initiate a transfer, then go on to other work. Whenever the peripheral device is ready to transfer a unit of data, it interrupts the CPU with a request. Burst transfer is a combination mode, initiated by a peripheral interrupt and completed in a fast handshake loop. The fourth transfer mode uses a very high-speed DMA controller, instead of the CPU, to service data requests. DMA transfers are normally overlapped with other computation by the CPU.

The graphics routines, which will be familiar to some readers as a subset of the DGL graphics library available on HP 1000 Computers, produce two-dimensional drawings suitable for many design and engineering applications. Graphical output can be sent to the built-in display, to external color monitors, and to a variety of plotters. Line drawing is very fast—about 2500 6-mm vectors per second can be drawn on the built-in display. Input can be taken from the knob on the keyboard or a digitizing tablet.

There is no distinction between user and system programs. In fact, the command interpreter is a relatively simple program that allows the user to execute other programs or subsystems, and to catch and report errors. It's easy to replace the command interpreter with any other program, resulting in a system that can only do what the substitute program knows how to do.

The linking loader and file system form the kernel of the system, the only part that has to be present for a single Pascal program to run. The loader is able to bring both programs and modules into memory. It also remembers what is in memory. This means that modules can be loaded once, then used over and over by subsequent programs. This feature can be used to extend or override system

Kathryn Y. Kwinn

Kathy Kwinn earned the BS degree in mathematics and chemistry from Saint Mary's College, Indiana in 1970. She then attended Iowa State University, receiving an MS degree in mathematics (1973), and the MS (1975) and PhD (1979) degrees in computer science. Kathy joined HP in 1978 and is involved with the BASIC software development for the 9826A and 9836A. She is a member of the Association for Computing Machinery (ACM). Kathy has taught FORTRAN and pattern recognition as a temporary assistant professor at Colorado State University. She was born in Cleveland, Ohio and lives in Fort Collins, Colorado. She is married (her husband also works at HP) and has a one-year-old daughter. Kathy's interests include swimming, sewing, and choral and organ music.

capabilities. Programs can also be loaded and retained in memory, to be called whenever they're wanted. When this is done, they are immediately executable. No later accesses to mass storage are required.

The operating system, consisting of the file system, I/O drivers, linking loader, and command interpreter, is a set of memory-resident libraries, most of which are written in Pascal. This means that the services provided can be called on by programs or other modules. Moreover, a programmer can construct custom modules that extend the capabilities of the system. Such extension modules are bound into the system simply by loading them, and there is an easy way to cause them to be loaded automatically when the computer is turned on.

The system can also be tailored by removing capabilities. All of the components mentioned above, except the linking loader and file support package, can be removed or replaced by the user in a few moments.

References

1. R.L. James and F.J. Yockey, "Interactive Model 20 Speaks Algebraic Language," Hewlett-Packard Journal, Vol. 24, no. 4, December 1972.
2. D.E. Morris, et al, "Third Generation Programmable Calculator Has Computer-Like Capabilities," Hewlett-Packard Journal, Vol. 27, no. 10, June 1976.
3. N. Wirth, "Modula: A Programming Language for Modular Multiprogramming," Software—Practice and Experience, Vol. 7, no. 1, January 1977, pp. 3-35.

Robert M. Hallissy

Bob Hallissy is a graduate of Virginia Polytechnic Institute and State University, holding the BA (1974) and MS (1975) degrees in electrical engineering. With HP since 1973, he developed option ROMs for the 9815 and 9835 Computers and the HPL language system for the 9826A and 9836A. Bob is project manager for 9826A/9836A software. He is the author of a paper on HP-IB controllers. A native of Hampton, Virginia, Bob is married and lives in Loveland, Colorado. Outside of work he participates in church activities and is interested in Bible study.

Roger E. Ison

Roger Ison joined HP in 1977 and was the project manager for 9826A/9836A BASIC software tools as well as the Pascal language system. He received a BA (1970) degree in international relations and the MCS (1973) and PhD (1977) degrees in computer science from the University of Virginia. Although he was born in Nyack, New York, he grew up in Saudi Arabia and Lebanon. He and his wife, who is curator of the Loveland museum, live in Loveland, Colorado and have a daughter, two cats, and a dog. Roger's current hobby is mathematical modeling of option investments.

Data Communications for the 9826A and 9836A Computer Systems

by Carl M. Dierschow and Robert P. Uhlrich

AS COMPUTERS become more common throughout the world, the need to exchange data between them also increases. Data communications is no longer a luxury; it is a necessity. The HP Model 98628A Data Communications Card provides this communication path for the 9826A and 9836A Computer Systems.

The 98628A card is a newly developed serial data communications interface for the 9826A and 9836A. It handles many asynchronous protocols and the Hewlett-Packard Distributed Systems Network/Data Link (DSN/DL) protocol. This interface fits well into terminal emulation applications because it is able to handle many dialects of asynchronous communications including half duplex, full duplex, and enquire/acknowledge or start/stop handshakes, as well as automatic detection of end-of-line and prompt-character sequences. It is also able to drive a variety of RS-232-C peripherals.

The data link protocol serviced by this card is tailored for data gathering applications in which an HP 1000 or HP 3000 Computer controls the operation of a factory floor. The protocol is handled automatically for the user, including blocking and full data transparency.

The 98628A is supported by the BASIC and Pascal languages used in the 9826A and 9836A. It uses the same I/O statements as other interfaces for these systems.

The 98628A contains three independent data and address buses: the mainframe, internal Z80 and shared buses (see Fig. 1). The mainframe bus provides a direct connection to the 68000 microprocessor's memory bus. The internal Z80 bus connects the Z80 microprocessor, timer, serial I/O (input/output) chip and program ROM (read-only memory). The RAM (random-access memory) and hardware registers are connected to both the Z80 and the 68000 by the shared memory bus. All interaction between the 68000 and the card's Z80 is performed through the RAM and registers on this bus.

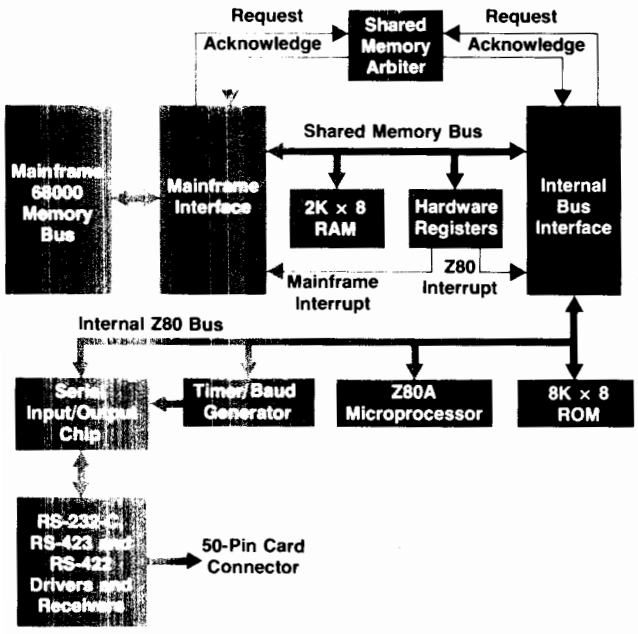


Fig. 1. *Block diagram of HP Model 98628A Data Communications Card.*

Mainframe Interface

The 98628A card contains a 2K-byte static RAM which is used to implement a powerful data communication interface to the 9826A and 9836A mainframes. It is shared between the processors in the mainframe and the 98628A card, meaning that both are allowed to read from and write to the memory. The arbiter ensures that both processors are not allowed to access the memory simultaneously. This shared memory fits well with the system processor which performs all of its I/O via a memory-mapping mechanism allocating 64K bytes to each I/O card.

The shared memory contains receive and transmit buffers. Both are implemented as dual circular buffers. One buffer of the pair is reserved for control information, while the other contains only data characters. A hardware semaphore protects access to critical pointers in this shared buffer scheme.

To satisfy some of the special mainframe I/O requirements, three special memory addresses are implemented in discrete hardware, and the first two locations of the shared RAM are supplemented with additional hardware. These special additions supply:

- A constant ID register, which uniquely identifies this type of card
- A mechanism whereby the card can interrupt the mainframe processor
- A mechanism whereby the mainframe can interrupt the card's processor
- A hardware semaphore that is used by the firmware to control access to critical resources in the shared memory for short periods of time.

The card-to-mainframe interrupt mechanism is able to communicate several different interrupts: received data available, transmit buffer space available, error, and an interrupt condition specified by the user language.

The mainframe-to-card interrupt mechanism communicates high-priority commands from the user to the card. Several other commands that are not used by the 98628A are sent via this mechanism by the system's drivers, but the card rejects these as unserviceable.

The data communications channel may be treated just

Protocols

Asynchronous Protocol

This term actually refers to two different levels of communications: the bit-level protocols, which are standardized, and the high-level handshaking protocols, which are not.

As shown in Fig. 1, each character consists of seven to twelve bits of binary data. The first is called the start bit, and is a logical zero, while the last 1, 1½ or 2 bits are called the stop bit(s) and are logical ones. Between these are five to eight data bits and an optional parity bit. The HP Model 98628A card is able to handle five forms of the parity bit: none, always zero, always one, even or odd. The last two options adjust the parity bit to produce a six-to-nine-bit word that contains an even or odd number of ones.

There are several asynchronous handshaking protocols. Two of these are half duplex and full duplex. These protocols deal with the type of modem connection used to communicate between computers. Half duplex allows only one computer to transmit at a time with many different conventions used to indicate when one computer is finished transmitting and the other may take its turn. Full duplex allows both computers to transmit and receive at any time. When using this kind of connection, there are several protocols by which a receiving computer may indicate to a transmitter that it is in danger of overrunning its buffers. Two of these are enquire/acknowledge (ENQ/ACK) and start/stop (XON/XOFF). The first is widely used by HP products, while the latter is popular among a number of other computers.

Data Link Protocol

The asynchronous multipoint protocol was first introduced as a means of connecting multiple HP terminals to larger HP computers. From this came the HP data link protocol (DSN/DL), which is essentially the same protocol communicated over a special two-wire connection. This cable allows communication distances of up to 4 km. Line control is determined through polling performed by the master computer. This link was used first as a means of connecting HP 3075A Desktop Data Capture Terminals and desktop computers to HP 1000 or HP 3000 Computers.

The characters of the DSN/DL protocol are asynchronous, with one start, one stop and eight data bits per character. The data is formed into blocks and a redundancy check is performed on each block to assure data integrity. If any data character is not communicated properly an automatic mechanism provides retransmission of the block of data. The controlling computer provides independent servicing of each terminal on the line to ensure that two terminals do not attempt to confuse each other's data.

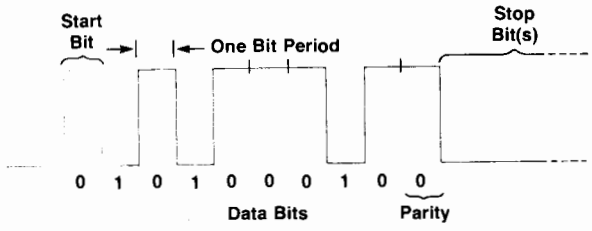


Fig. 1. *An asynchronous character binary data stream.*

like any other I/O channel. In the outbound direction, control information is placed in the buffer with BASIC statements such as CONTROL and OUTPUT END. Outbound characters are placed in the buffer by OUTPUT, PRINT, and other statements that produce output for a printing device.

In the inbound direction, control information is placed in the buffer according to decisions made by the card. The card allows individual selection of several conditions such as end-of-line received, prompt received, end-of-block, and parity errors. From this information, the BASIC program can precisely determine the location of these conditions relative to the data. The control information terminates an ENTER statement when it is reached, and the information associated with it may be read via two STATUS registers.

Because of this mixed control and data buffering, several blocks of data can be buffered in both directions, allowing the user's program more time to do useful work.

The linkage between the user program and the interface card is provided by some firmware, called the drivers, in the 9826A and 9836A operating systems. The drivers are designed not only to make the card independent of the mainframe, but also to provide a common linkage mechanism for the 98628A and future data communications cards. The drivers are very simple, acting mainly as a means of communicating the user's intentions to the card. The card itself makes many decisions about the meaning of the user's statements. The drivers are integrated into the I/O structure, which allows ENTER and OUTPUT formatting as with other I/O cards.

Data Communications Interface

The 98628A has a versatile electrical interface for connection to a wide variety of devices. Four cables and three electrical converters (pods) provide several interface options:

- RS-232-C DTE (male) cable
- RS-232-C DCE (female) cable
- RS-449/423 DTE (male) cable
- RS-449/422 DTE (male) cable
- 13264A Data Link Adapter
- 13265A 300-bit-per-second Modulator/Demodulator (modem)
- 13266A Current Loop Converter.

The card contains drivers and receivers compatible with the Electronic Industries Association (EIA) RS-232-C, RS-449/423 and RS-449/422 standards. One set of electrical drivers implements the RS-232-C and RS-449/423 standards. A separate set of drivers implements RS-449/422. To implement a given standard, the cable uses the appropriate pins on the 50-pin card connector.

The pods change the RS-232-C electrical signals from the card to the proper levels for connection to other types of devices. Each pod consists of a box containing active circuitry built into the cable. Power is supplied by the mainframe via the 50-pin connector. The 13264A Data Link Adapter converts the signals to the electrically isolated multidrop HP Data Link levels. The 13265A is a 300-bit/s modem for direct connection to North American telephone lines, supporting automatic or manual dialing to establish a connection to a remote computer. Finally, the 13266A Current Loop Converter is used to connect to older current-loop devices.

Firmware

The 98628A firmware services two different communications protocols: asynchronous and data link (DSN/DL). The first is used for communication with almost any type of computer, often over telephone lines. The second is an HP

Electrical Standards

For data communications, early electronic computers borrowed the current-loop signaling technique from telegraph and teletypewriter applications. This technique uses either the absence and presence of an electrical current or its direction to signal a logical zero or one. Because teletypewriters do not have stringent interfacing requirements, a current-loop standard has never been developed. The driving voltage, typically between 6 and 140 volts, determines the maximum cable length.

In the early 1960s, the Electronic Industries Association (EIA), responding to the need for standardized interfaces between computers, terminals and modems, developed the RS-232-C standard. RS-232-C defines the electrical and functional characteristics of the interface between the devices and uses voltages to signify logic levels. The data lines use negative-true logic—a positive voltage between 5V and 25V represents a logical zero and a negative voltage between −5V and −25V represents a logical one. The control lines use positive-true logic—a positive voltage signifies an on condition and a negative voltage represents off. A 25-pin connector has become the RS-232-C *de facto* mechanical standard. RS-232-C specifies a maximum cable length of 15.24 metres and data transfer rates up to 20,000 bits/s.

With the need for longer cable lengths and higher data rates, the EIA defined the RS-449, RS-423 and RS-422 standards in the mid-1970s. RS-449 defines the operation of 27 data, timing and control lines. The basic functional characteristics of RS-232-C are retained; however, ten new circuits are defined, three old circuits are deleted and some changes are made to the functional definition of some others. RS-449 specifies a 37-pin connector for main channel circuits and an optional 9-pin connector for secondary channel lines.

To provide an evolutionary growth path to the new standards, RS-423 was designed to be subset-compatible with RS-232-C. Voltages in the positive and negative 4V-to-6V regions indicate logic and data levels with the same polarity as RS-232-C. Note the overlap between RS-232-C and RS-423 in the 5V-to-6V region. RS-423 achieves data rates up to 100,000 bits/s and longer cable lengths using a combination of lower voltage swings and slower transition rates between levels to reduce crosstalk between lines.

RS-422 defines a balanced interface circuit using the polarity of the voltage between a pair of conductors to indicate a logic level. Balanced circuits provide greater noise immunity to permit long-distance communication at speeds up to ten million bits/s. RS-422 devices cannot interface directly to RS-232-C devices.

protocol used by HP computers, data capture terminals, regular terminals, and now desktop computers.

The 98628A card firmware is designed so that it is easy to use. For example, although there are two distinct data communications protocols that this card understands, it is possible to select between them either with a hardware switch or by software commands from the user's program. There are seven other protocol default switches that can be used to load initial option values into the card. However, all options can be overridden by user commands.

The eight protocol default switches on the card individually select the communications protocol, common data rate, and modem control protocol. For asynchronous operation the user also may select from four common combinations of bits per character and parity. DSN/DL operation allows selection between eight different device addresses.

The firmware on the 98628A card was written to handle the asynchronous and DSN/DL communications protocols completely. First, the card controls the modem lines that communicate with the attached hardware modem or pod. Second, the card handles the software, or character, protocol to prevent buffer overruns.

In the asynchronous mode this software protocol checks for handshake characters, end-of-line sequences, and prompts. All of these characters may be changed or disabled by the user to suit specific applications. It is useful for the card to detect these characters because the work required by the mainframe is reduced and buffer overruns are prevented. For example, the card can search for carriage-return/line-feed combinations and logically split the data at such points to facilitate data entry by the mainframe.

The DSN/DL protocol is considerably more complicated, with blocking, error checking, and timeouts. The firmware for this protocol is designed as a state machine. With this programming concept, the reception or transmission of each protocol character results in a change of the card firmware's state. This design promotes structured, modular code and a more logical servicing of the DSN/DL protocol.

The DSN/DL protocol features excellent error detection and recovery, mainly because a cyclic redundancy check (CRC) is transmitted and checked with every block of data. If the receiver's calculations do not agree with the transmitted CRC, the sender is asked to repeat the block of data. Other redundancies are also built into the protocol. For example, when the controlling computer asks for data from a slave computer it issues a polling sequence containing two addressing characters, which are both repeated to avoid confusion caused by spurious line errors.

Since the card attempts to handle both the hardware and software protocols completely, there are few cases in which

Carl M. Dierschow
Carl Dierschow has been with HP since 1978 and is a project engineer for 9835A, 9845, 9826A, and 9836A data communications. He holds a BSEE/CS degree awarded by the University of Colorado in 1978. Carl is a native of Denver, Colorado. He is married, has one daughter, and lives in Fort Collins, Colorado. His outside activities include skiing, swimming, and playing soccer.

Robert P. Uhlrich
Rob Uhlrich is a native of Bozeman, Montana and attended Montana State University, earning the BS and MS degrees in electrical engineering in 1976 and 1977. He then joined HP and worked on the HP-85 serial I/O and the 98046 and 98628 data communication interfaces. Rob became a project manager last year. He is married and lives in Fort Collins, Colorado. His interests include gardening, woodworking, and skiing.

action other than the normal movement of data is required of the user's program. However, for those exceptional cases, additional CONTROL and STATUS registers are provided, along with interrupt mechanisms, to allow a program to control the card's operation.

To facilitate friendliness and comply with legal requirements for switched public telephone circuits, several timeouts have been implemented to catch such conditions as:

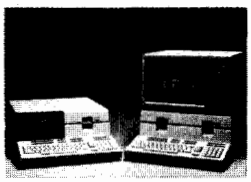
- Not establishing a connection (caused by a misdialed number or busy circuits)
- Losing a connection (the remote computer hangs up or there is a break in the circuit)
- Half-duplex transmission lockup (incompatible or confused half-duplex protocol)
- Lack of data activity on the line (loss of connection or computer down).

Each of these timeouts may be individually disabled or set to a new value by the user.

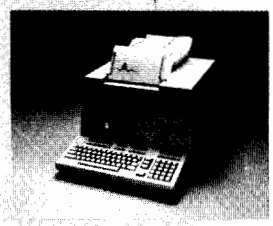
Self-Test

In a typical data communications system, a number of devices from various manufacturers are connected together. If the system does not operate properly, being able to test each of the individual blocks is important to isolating the problem. The 98628A has self-test programs to respond to this need. At power-up, the card's firmware runs a test and signals any failure to the user. A connector for a more thorough hardware and cable test is also available.

PRODUCT INFORMATION



HP Models 9826A and 9836A Computer Systems
MANUFACTURING DIVISION:
Desktop Computer Division
3400 E. Harmony Road
Fort Collins, Colorado 80525 U.S.A.
TECHNICAL DATA: HP Publications 5953-4576 (9826A), 5953-4589 (9836A), 5953-4581 (BASIC), 5953-4594 (HPL), and 5953-4588 (Pascal).
PRICE IN U.S.A.: 9826A Computer System: Option 011 (ROM BASIC), $8950; Option 014 (ROM HPL), $8950; Option 715 (RAM Pascal), $12,750.
9836A Computer System: Option 011 (ROM BASIC), $11,950; Option 014 (ROM HPL), $11,950; Option 715 (RAM Pascal), $15,750.
98261A Add-On Language for 9826A/9836A: Option 011 (ROM BASIC), $1200; Option 014 (ROM HPL), $1000; Option 715 (RAM Pascal), $2000.



HP Model 2670 Series Printers
MANUFACTURING DIVISION
Vancouver Division
2400 N.E. 65th Avenue
Vancouver, Washington 98661 U.S.A.
TECHNICAL DATA: HP Publication 5953-6260.
PRICE IN U.S.A.:
2671A Printer, $1195.
2671G Graphics Printer, $1495.
2673A Intelligent Graphics Printer, $2195.

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