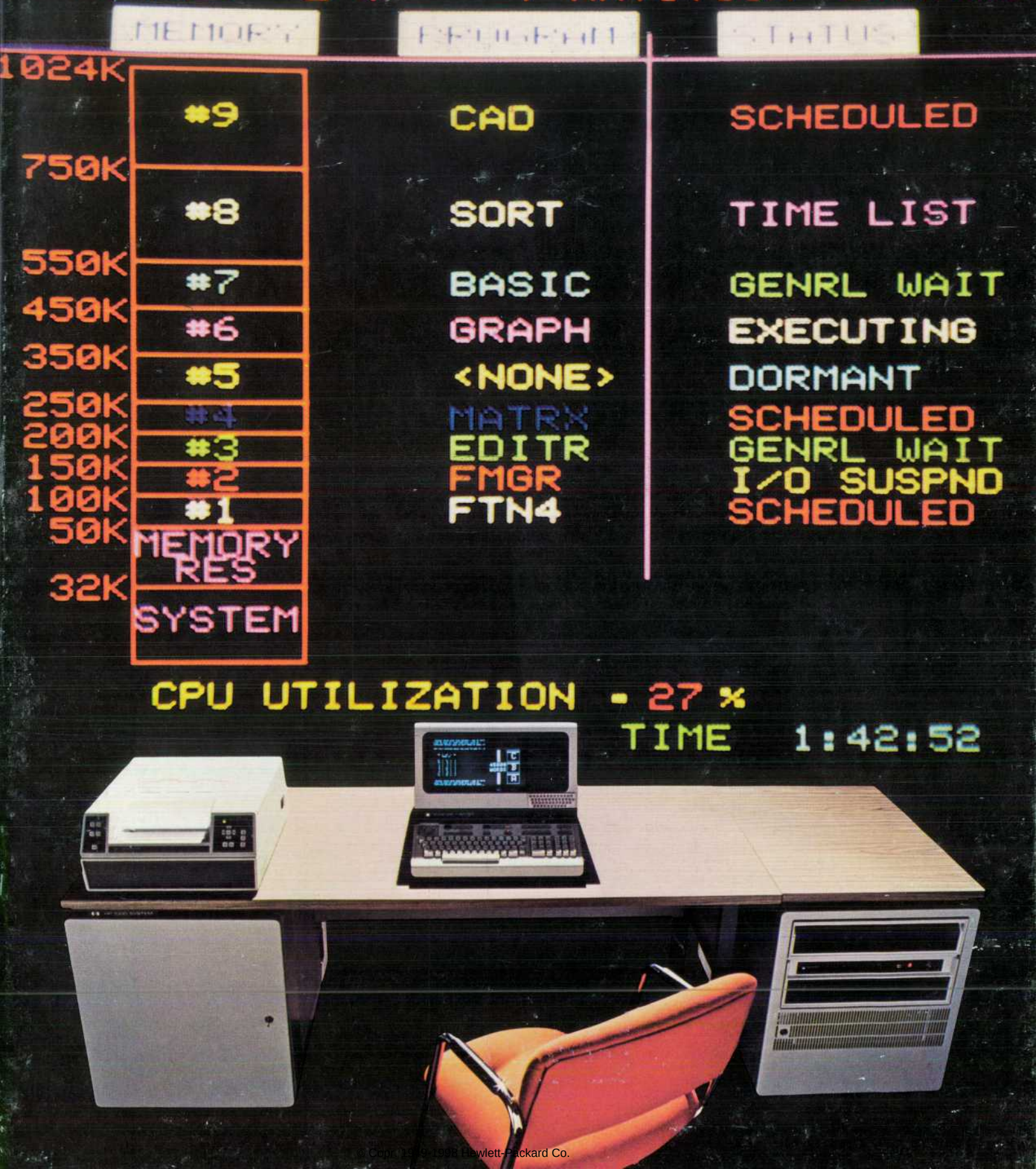
OCTOBER 1978

HEWLETT-PACKARD JOURNAL



Higher-Performance HP 1000 Computer Systems

The higher performance comes from new technologies, including new processors, faster 16K RAM semiconductor memories, and a new operating system.

by Rodney K. Juncker

HP 1000 COMPUTER SYSTEMS, first introduced in late 1976,[1] are designed to give the user a choice of preconfigured nucleus systems that are easy to use, easily adapted to user applications, accurately specified, and easily supported and maintained. Instead of having to build a system from a vast array of hardware assemblies and software modules, the user can choose a nucleus system that offers a tested and documented starting point for any application effort.

HP 1000 Systems are based on HP 1000 Computers (formerly 21MX Computers) and the real-time executive (RTE) operating system. Because the application areas for these systems are extremely varied and cannot be covered by a single system, a family plan was established, defining a range of systems suited to different applications.

The family starts with a low-cost system able to run applications programs under RTE control and provides an economically sound path for expansion to larger systems. The expansion path not only allows for the conversion of the starter system to a more powerful member of the family but also allows for the interconnection of a large number of family members to form a network.[2] With these capabilities, a user can solve almost any problem in almost any application area.

New HP 1000 Systems

Two new systems have now been added to the upper end of the HP 1000 family. These systems, Models 40 and 45, feature a new real-time executive operating system, RTE-IV, that manages up to 64 programs simultaneously and handles data arrays as large as two megabytes. The Model 45 System is based on a new version of the HP 1000 Computer, designated the F-Series. Its hardware floating point

Cover: *An HP 1000 Model 45 Computer System, newest and most powerful member of the HP 1000 family, is shown with a television display of its multilingual, multiprogramming computing activity. The new RTE-IV operating system has special facilities for managing large memories and large programs. (The TV monitor software was developed by author Mike Manley, the TV interface by author Denton Anderson).*

In this Issue:

Fig. 1. *The new Model 45, the most powerful HP 1000 System, features the new F-Series Computer with floating-point processor and the new RTE-IV operating system. It is designed for applications involving extensive computation, large programs, large data arrays, and graphics. It is shown here with 100M bytes of disc storage.*

processor, new scientific instruction set, and 350-ns 16K RAM memory give it significantly greater performance than the E-Series, which is the processor for the Model 40 System.

Another new HP 1000 System, Model 25, uses the F-Series Computer but has a memory-based operating system, RTE-M.

The HP 1000 Family

The HP 1000 family now consists of the following members, in decreasing order of capability.

The Model 45 System (Fig. 1) is the most powerful HP 1000 system, incorporating the high-performance F-Series Computer with built-in hardware floating point instructions and scientific instruction set firmware, 128K bytes of high-performance high-density memory, a high-performance graphics terminal with dual mini-cartridge units, a 19.6M-byte cartridge disc memory, the powerful RTE-IV operating system, and a versatile graphics applications software package. This system is oriented towards applications where high-speed computational power, graphics capabilities, large data arrays, and large program areas are required. It can be easily expanded to include up to 2M bytes of standard or high-performance memory, up to 1.8M bytes of fault controlled memory, up to 400M bytes of disc storage, distributed systems networking capability, data base management with IMAGE/1000 software, a BASIC language capability with the BASIC/1000 software, and a wide variety of peripherals and accessories.

The next most powerful member of the HP 1000 family is the Model 40 System. This system differs from the Model 45 in that it does not include the hardware floating point instructions and scientific instruction set firmware and that the graphics terminal and high-performance memory are optional. This system is oriented towards applications similar to Model 45's but where the high-speed computation capability is not essential. The Model 40 System can be expanded in the same way as the Model 45.

The Model 30 System is the original member of the HP 1000 family. This system incorporates the E-Series Computer, 64K bytes of standard performance memory, a 19.6M-byte cartridge disc memory, a fast, flexible display station with dual mini-cartridge units, and the RTE-II operating system. This system can also be expanded to include up to 400M bytes of disc storage, BASIC/1000, IMAGE/1000, and a wide range of peripherals and accessories.

The Model 25 System is a high-performance memory-based system. This system incorporates the high-performance F-series Computer with built-in hardware floating point instructions and scientific instruction set firmware, 64K bytes of high-performance memory, a fast, flexible display station with dual mini-cartridge units, and the RTE-M memory-based operating system. The Model 25 System is oriented towards applications that require low-cost, high-performance systems to be used as stand-alone systems or network satellite nodes. Model 25 can be easily expanded to include flexible

disc storage, graphics terminals, up to 2M bytes of memory, fault control memory, BASIC/1000, DS/1000 and many of the same peripherals and accessories that are available on the larger systems.

The Model 20 System is the smallest of the family members and differs from the Model 25 System in that it uses the E-series Computer instead of the F-series Computer. This system is a flexible, powerful low-cost system especially suited for applications such as instrumentation control, remote test and measurement stations in harsh environments, and laboratory test and measurement stations. This system can be expanded in the same way as the Model 25.

New HP 1000 Capabilities

In the design of the latest HP 1000 Systems, major contributions were made in computers, operating system software, and other system elements. These contributions include a new megaword-array operating system, new computers with hardware floating point and scientific instruction sets, a new power supply design, new memory subsystems, a graphics software package, and a multipoint terminal subsystem, many of which are discussed in detail in succeeding articles in this issue.

The major new system design contributions made to the HP 1000 family are focused in the RTE-IV operating system and the F-Series Computers. Besides providing the user with a high level of computing power and programming capability, the operating system design achieved some other very significant goals. By placing peripheral drivers in special memory partitions and bringing them into the user's space only when needed, additional addressing space was made available to user programs. By allowing peripheral device and memory reconfiguration to be done at system startup, the need to regenerate a system that must run in a computer with different memory size or peripheral device configuration was eliminated. These two contributions are of major importance to the system design goals, because they made possible the design of "primary" systems. The primary systems are a set of system software generations that include the most-often-used drivers and software subsystems and a set of computer and peripheral test programs that run under RTE. Because RTE-IV can be reconfigured at system startup, these preconfigured and tested primary systems can be easily set up for use on any of the Model 40 and 45 Systems. This eliminates the need for uniquely generating every system shipped to a customer, saving many hours of technician time per system, and allowing the customer to adapt this proven primary system to the application instead of being forced to perform a lengthy and difficult system generation when the system arrives.

The primary systems also include the on-line test programs that run under RTE, so computer and peripheral tests can be performed under the control of the same software operating system as the application programs. With the introduction of on-line system test and diagnosis, some major new capabilities can be realized. The user can perform some system tests while applications are running, increasing trouble-shooting speed and flexibility. Also, possibilities for remote test and remote fault diagnosis now exist, and will be essential for systems operating in distributed system networks and at remote, unattended sites.

Acknowledgments

The development of any large family of systems requires the devotion, ingenuity, and resources of many people. This is especially true for HP 1000 Computer Systems, and all who contributed at HP's Data Systems Division are to be complimented. The following people were uniquely responsible for the success of the family: Steve Boettner, Bob Daniel, John Frohlich, Chris Lehner, Hugo Schaerli, Bill Thomas, and Neal Walko.

References

1. L. Johnson "A New Series of Small Computer Systems," Hewlett-Packard Journal, March 1977.
2. R.R. Shatzer, "Distributed Systems/1000," Hewlett-Packard Journal, March 1977.

Rodney K. Juncker

Rod Juncker began work at HP in 1967 as a microwave network analyzer systems designer. In 1971, he was project manager at HP's Waltham Division for a hospital medical instrumentation system, and in 1976 he began work as project manager for HP 1000 system products. Born in Salinas, California, Rod spent seven years in the U.S. Navy as an electronics and computer science teacher and as an aviator in Vietnam. He received his BSEE degree in 1959 and his MSEE degree in 1967, both from Stanford University. In his leisure time, Rod enjoys swimming, tennis and camping in the Sierras or at Big Sur with his wife and two sons, ages eight and thirteen.

HP 1000 SYSTEM SUMMARY

System Type	MODEL 20		MODEL 25		MODEL 40		MODEL 45		MODEL 30		
Product Number	2174A	2174B	2175A	2175B	2176A	2176B	2177A	2177B	2170A	2171A	2172A
Base system computer type	E-Series		F-Series		E-Series		F-Series			E-Series	
Type of memory	Standard		High-performance		Standard		High-performance			Standard	
Memory cycle time	595 ns		350 ns		665 ns		420 ns			595 ns	
Operating system	RTE-M		RTE-M		RTE-IV		RTE-IV			RTE-II	
System console	2645A		2645A		2645A		2648A			2645A	
Memory: Base	64K	64K	64K	64K	128K	128K	128K	128K	64K	64K	64K
(bytes) Maximum†	2048K*	1280K	1280k	1280K	2048K*	1280K	1280K	1280K	64K	64K	64K
Standard system disc	None		None		7906 (19.6Mb)		7906 (19.6Mb)		7900 (4.9Mb)	7906 (19.6Mb)	
Optional alternate system discs	None		None		7920 (50Mb) 7900 (4.9Mb)	7920 (50Mb)	7920 (50Mb)	7920 (50Mb)	None		
Flexible disc available?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RJE/1000 available?	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DS/1000 available?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
IMAGE/1000 available?	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
91000A/2313A Analog-Digital Subsystem available?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2240A Measurement & Control Processor available?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
92840A GRAPHICS/1000 software available?	Yes	Yes	Yes	Yes	Yes	Yes	Incl.	Incl.	No	No	No
12790A Multipoint interface available	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
12979B Dual-Port I/O Extender available?	Yes	No	Yes	No	Yes	No	Yes	No	Yes	Yes	No
12990B Memory Extender available?	Yes	No	No	No	Yes	No	No	No	Not applicable		
Additional terminals, line printers, magnetic tape units, etc., available?	Additional peripheral devices are generally compatible with all HP 1000 Computer Systems, but check with the configuration guide when ordering to confirm availability for a particular system.										
Base system price (U.S.A.)	$22,000	$22,000	$27,500	$27,500	$38,500	$38,500	$45,000	$45,000	$31,500	$36,500	$36,500

†These figures are for non-fault-control memory; fault control reduces maximum capacity from 1280K bytes to 1024K bytes in computer mainframe, from 2048K bytes to 1792K bytes in computer mainframe plus memory extender.

*This memory size requires the additional memory module capacity provided by the 12990B Memory Extender.

MANUFACTURING DIVISION: DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, California 95014 U.S.A.

RTE-IV: The Megaword-Array Operating System

by Eugene J. Wong and C. Michael Manley

THE REAL-TIME EXECUTIVE is Hewlett-Packard's multi-user, multiprogramming operating system for HP 1000 Computer Systems. RTE comes in several versions, all upward compatible. These include the memory-based RTE-M, the disc-based RTE-II and RTE-III,[1,2,3] and the new RTE-IV, the most powerful HP real-time executive system to date.

RTE-IV's new operating system features include megaword data arrays, user code areas of up to 54K bytes, reporting and recovery from parity errors, memory and input/output reconfiguration, new multiterminal handling software, an improved user interface for languages, and expanded device driver areas.

These new features, especially megaword data array handling, have allowed RTE-IV to move into application areas formerly reserved for large mainframe systems. RTE-IV is already being used for large-scale linear programming, operations management, simulation, computer-aided design, and matrix manipulation problems. RTE-IV is available as a standard product (92067A) and as the operating system in two HP 1000 Systems, the F-Series-Computer-based Model 45 and the E-Series-Computer-based Model 40.

Like RTE-II and RTE-III, RTE-IV offers priority scheduling of concurrent programs, separation of real-time and background tasks into real-time and background partitions, and a powerful file management package. It provides program partition swapping, buffered output, "mailbox" input/output, online system generation, and a batch entry processor featuring both input and output spooling of jobs for maximum throughput.

Like RTE-III, RTE-IV manages up to two megabytes of main memory in the HP 1000 M-, E-, and F-Series Computers,* in up to 64 real-time and background partitions. However, with RTE-IV this entire area may also be used by just one program. Combined with the DS/1000 and DS/3000 Distributed System packages,[4,5] RTE-IV becomes a powerful network node capable of controlling distributed processors at satellite RTE nodes. Other software products that may be used to extend the power of RTE-IV include BASIC/1000,[6] data base management with IMAGE/1000, and the RTE microprogramming package.[7]

*Formerly 21MX Computers.

Memory Management

One of the major features of RTE-IV is its memory managing ability. The operating system is capable of managing up to 1024 pages of physical memory, each page consisting of 1024 sixteen-bit words. A hardware option to the E-Series and F-Series Computers, called the dynamic mapping system[2,8] provides four banks of 32 registers each. These are used as physical page registers. The 32 pages of physical memory described by the bank that is currently enabled are the 32 pages that make up the logical memory space.

The four 32-register banks are called "maps." The same term, "map," is also used to refer to the physical memory designated by the contents of the 32 registers. The two meanings of "map" are used interchangeably in this article.

The four maps include the system map, where the RTE-IV operating system resides, the user map, where the current user program resides, and two maps used for direct memory access by the dual-channel port controller (DCPC). Since the DCPC runs concurrently with program activity, up to three maps may be active at one time, either the system map or the user map and both DCPC maps.

One of the benefits of these maps is that, although the pages of memory they represent need not be physically contiguous, the system makes them appear logically contiguous. For example, the first three pages of a user map might be pages 0, 50, and 600. In this case the first page would be physically and logically the same page. However, the next page, physical page 50, would logically appear to be the second page. That is, to a program executing under the user map, the code on physical page 50 is used whenever accesses to the second page of the user map are desired. This ability to map physical memory into a logical address space is used extensively in the system and user maps under the guidance of RTE-IV.

The System Map

Every time an interrupt occurs in the HP 1000 Computer, the system map and thus the operating system is automatically enabled by the hardware. This allows the operating system to examine the source of the interrupt and determine the appropriate action. Interrupts are generally of two kinds, a user requesting executive services via a memory protect

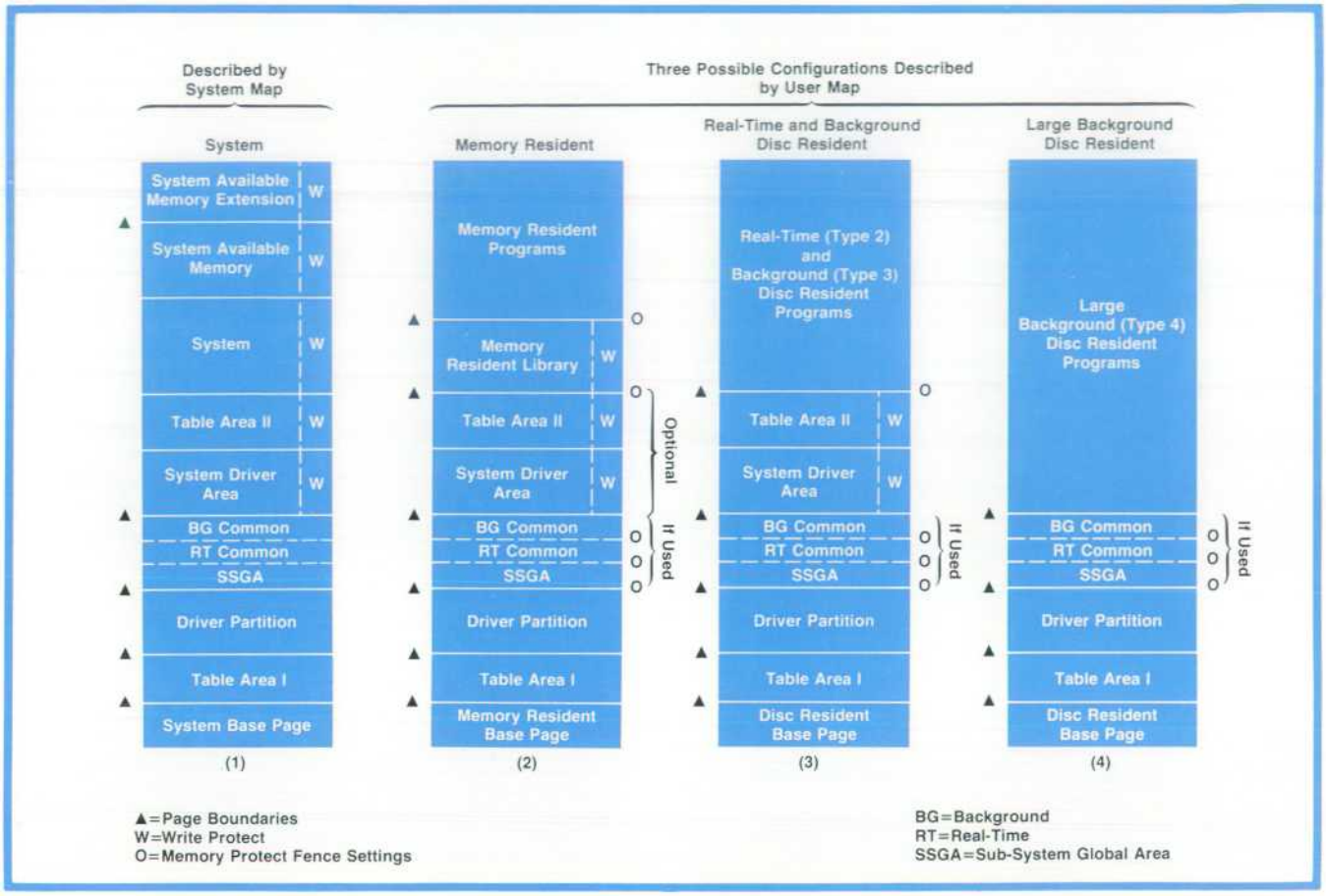


Fig. 1. *RTE-IV manages up to 2M bytes of physical memory using the dynamic mapping system, a hardware option to HP 1000 Computers. Segments of physical memory are mapped into 64K-byte logical memory spaces according to four maps: the system map, the user map, and two dual-channel port controller maps. At left is a picture of the system map as it would look when processing an interrupt. Also shown are three possible user map configurations. The maps are set up by the operating system.*

interrupt or a device interrupt that informs the system that a data transfer has finished and the device is now free. The left side of Fig. 1 shows a picture of the system map as it would look when processing an interrupt.

The system map in RTE-IV is static except for the driver partition area. The driver partition area, a new feature of RTE-IV, is a reserved area in each map that is usually two pages long and is used to address a device driver. A device driver is a software module that operates a device under the control of the operating system. This area is dynamic because drivers are included in a map only when required for handling input/output requests or device interrupts. The drivers are actually located in physical memory in a driver partition. When a device interrupt is acknowledged, RTE-IV determines which driver is needed, which partition the driver resides in, and which physical pages define that partition, and then maps these pages into the reserved area of the appropriate map. Thus, while these drivers are in different areas of physical memory, they all execute in the same area of logical memory. This concept of dynamically mapping a driver into logical memory is extremely efficient in conserving logical address space and improving real-time response. Other solutions to the driver addressing problem would require either having all of the drivers present at once in the active map, which would force the size of the user program or operating system to be smaller, or bringing drivers into memory from the disc, a slow, inefficient procedure, unworkable when real-time response is required. Now many drivers may be kept in physical memory and called into logical memory quickly whenever required. This allows a much greater variety of user problems to be solved with just one system.

The User Map

As mentioned above, the system map is static except for the driver partition. The user map, which is extremely dynamic, is controlled by the operating system. It is frequently being changed from one area

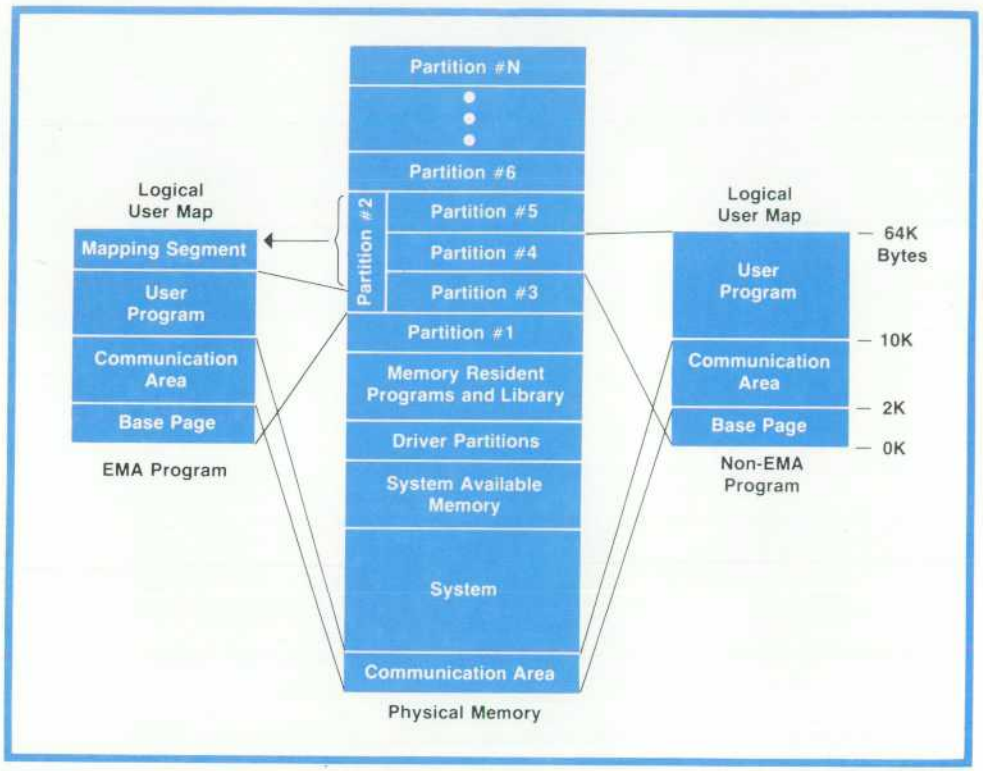


Fig. 2. *A physical memory map and two possible user map (logical memory) configurations. The map at left is for an extended memory area (EMA) program. EMA, a new feature of RTE-IV, makes it easier to manipulate large amounts of data. The EMA map is identical to the non-EMA map at right except for the mapping segment, a two-page window that is moved through physical memory, in this case partition #2, to access the requested data. (Note: partition #2 is a mother partition consisting of partitions #3, #4, and #5. Mother partitions are another new feature of RTE-IV.)*

of physical memory to another area of physical memory containing another program. This context switching is one of the tasks the operating system performs to suspend execution of one program and execute another.

The biggest change to the user map between RTE-III and RTE-IV was the removal of most of the operating system code from the user map. Fig. 1 shows three possible user map configurations. Fig. 2 shows a simplified version of the same map. Formerly, operating system code resided in both the user map and the system map, thus reducing the area available for user programs. With RTE-IV, the operating system code resident in the user map was replaced with a communications area to the executive. Since this area is much smaller than the system code itself, the area left for the user was greatly increased. The user code area was expanded to 54K bytes, which in most cases is twice as much as previously available.

Partition Types

As can be seen in Fig. 2, the user map is set up to point to a partition containing a program and to the communication area to the system. Partitions are divided into two types, real-time and background. In general there is not much difference between types. The distinction was made to increase program dispatching speed. For RTE-IV, a new type of partition, the mother partition, was created. A mother partition is a collection of real-time or background partitions united to form one very large program area. This scheme of collecting partitions allows RTE-IV to form a number of very large partitions, up to nearly two megawords, when required for large programs. However, when a large partition is not required, this memory reverts to a number of smaller partitions that may be used for normal-size programs. These mother partitions are typically used for programs that must manipulate a great deal of data.

Megaword Data Arrays

A goal during the development of RTE-IV was to give users the ability to execute very large programs, larger than the 32 pages of memory available under the user map. In analyzing these programs it was discovered that, for the most part, the programs are large not because of the program code, but because of the data declarations. That is, most programs are large because of the data to be manipulated, not because of the processing that the data undergoes. Thus the problem of large programs is mainly one of handling vast amounts of data. This problem was addressed in RTE-IV by dividing partitions into two parts, one for the program and the other for data. This special data area is called the extended memory area (EMA).

Since a program may still access only the 32 pages of memory currently enabled under the user map, a method was needed to bring into the user's logical address space any data required. This was accomplished by creating a window (a minimum of two pages) called a mapping segment (MSEG) at the top of the user map (see Fig. 2). When the user wants to

access a particular data element, a call is made to an extremely fast microcoded subroutine. The routine finds out which page the element is on, maps that page into the user's space, and returns the logical address to the program.

Fig. 3 shows the ease with which even a simple FORTRAN program can manipulate very large data arrays. The program adds two 30,000-word arrays, placing the result in a third array. In Fig. 3a an element of array B is accessed by moving the MSEG into the EMA area containing array B. In Fig. 3b the MSEG is moved to the C array area. The MSEG is moved a third time to address the element of the A array where the result will be stored. To the user program, it always appears that the appropriate section of memory is accessible.

Comparing EMA speed to other data segmentation schemes that require disc accesses, EMA can easily be a thousand times faster when the accesses are random in nature. As mentioned in the introduction to this article, this speed has allowed RTE-IV to move into application areas formerly reserved for large mainframe systems. Moreover, the large data areas have reduced critical path disc use by allowing many operations, such as sorting and searching, to be done completely in memory instead of on the disc.

Dispatching Programs

Before a disc-resident program can be executed, a partition must be found and a user map built for it. This process of determining which program to execute and which partition to load it into is called dispatching. Programs may be assigned to partitions to optimize dispatching speed. Otherwise, the system will search for the smallest empty partition that is large enough to hold the program. To reduce competition for partitions, RTE-IV provides different types of partitions—real-time, background, and mother—for different types of programs.

Once a free partition is found, the user map is set up for loading the program into memory. The user map's base page is the first physical page of the partition. The next few pages are set up in accordance with the program's needs; for example, the system common area is mapped only for those programs that need it. The remaining pages of the user map are the rest of the program's pages. Once this map is set up, a copy of it is kept in a special area of the user's physical base page that does not get mapped and is therefore not accessible by the user program. This copy of the map speeds the mapping of I/O drivers to handle interrupts, reduces interrupt latency, and speeds future dispatching of the program.

The dispatching of EMA (extended memory area) programs to mother partitions is much more involved than the dispatching of normal programs. If a program is assigned to a mother partition, or if an EMA program is vectored to a mother partition by the system, the status of each subpartition must be checked. If all of the subpartitions are free or if the programs resident in them are swappable, then those subpartitions are made unavailable to all programs not assigned to them and the necessary swaps are performed. If any program in a subpartition is in an unswappable state, the next mother partition of the appropriate size is checked. When all the subpartitions in a mother partition are empty, the mother partition is then available for a program.

Program contention for partitions is resolved, in general, by swapping lower-priority programs to the disc to make room for higher-priority programs. These swapped programs will be redispatched when a partition of the correct type becomes available. A program may resume execution in a different partition from its original one if no specific partition was requested.

EMA programs also contend for mother partitions on a priority basis. But how does one swap an EMA program of up to two million bytes to the disc? It cannot be swapped all at once, because 32 pages is the maximum DCPC transfer length. Therefore the swap is performed in two parts. The program code area is swapped first, and then the EMA data area is written to the disc in large chunks, up to 54K bytes long. This method provides better disc use and less delay in the operating system. A similar procedure is followed for the subsequent redispatching of swapped EMA programs.

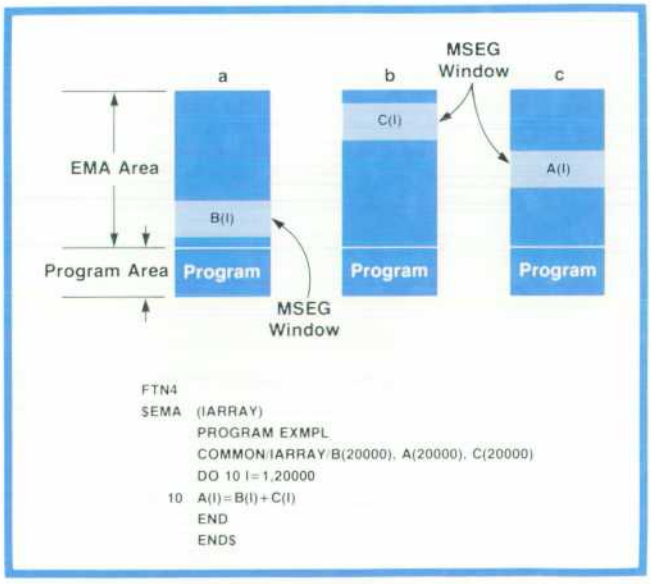


Fig. 3. *Use of the two-page mapping segment (MSEG) to access data in various locations in physical memory. The operating system moves the MSEG to the location of each desired data element. To the user program it appears that the needed section of memory is always accessible.*

Parity Error Fail-Soft

Bad memory is a problem faced by all computer manufacturers and users. Even the most rigorous testing efforts will not prevent some memory from degenerating and failing over a period of time. This problem is particularly acute in real-time systems used for process control or event and sensor monitoring applications, which require continuous operation. In megaword systems the probability of a memory failure is much higher than in smaller memory systems. To combat this problem, new parity-error-handling software was added to the RTE-IV operating system to maintain orderly execution of application programs.

Parity errors are automatically detected by the computer and cause an interrupt to the operating system. The logical address where the failure occurred can be fetched from the computer's violation register, but the true physical address must be found by looking at that logical address in all four maps.

RTE-IV looks first to see if the error is in the system map. If the parity error is in the operating system, further execution with a bad value or a bad instruction may cause unpredictable errors or catastrophic results, so in this case the computer is halted. The page number and the logical address of the bad memory location are displayed in front-panel registers so that the bad memory board can be identified and repaired.

If the parity error is not in the system map, the DCPC maps are checked next. If the error is still not found, the user map is checked. By now the bad location should have been identified. If the error was in a program partition, the program is aborted and all necessary information to locate the bad word of memory is printed on the system console. In addition, the partition is removed from the system so that future programs will not encounter the same error. If the error is in a subpartition of a mother partition, that mother partition is also removed from the system.

Parity errors not found by this verification process are "soft" parity errors. These rare errors are generally due to intermittent part failures or sometimes to improper use of user microcode. In any case, the location and other information is printed on the system console to aid the user in detecting the source of the failure. The key point is that the system continues to operate.

Memory and I/O Reconfiguration

Over a period of time a system's memory requirements may change as bad memory is removed for repair or more memory is added to support large programs. Input/output (I/O) demands on a system may also change because of new programs or changes in existing programs. To meet these needs for system flexibility, an optional memory and I/O reconfiguration phase was added to the system start-up procedure.

Memory reconfiguration allows the user to add or delete memory and to declare pages of memory to be bad. Thus if delays are unavoidable in repairing bad memory, the user can inform the operating system of the bad areas so these can be avoided. The user may also increase or decrease the size and number of program partitions.

I/O reconfiguration allows the user to move peripheral I/O cards into any position in the computer card cage. This permits device I/O priorities to be changed to improve system throughput. Another benefit of I/O reconfiguration is the ability to use an RTE-IV configured system disc from one system on another system of a slightly different configuration, provided the two systems have similar equipment.

New Multi-User Features

In addition to the above features, RTE-IV has many new "friendly" aspects. To aid program development, the FORTRAN compiler, the assembler, and the relocating loader were improved to provide a better user interface and to use files for input and output. This allows multiple users to execute many copies of FORTRAN, the assembler, and the loader at the same time. Several individuals may be doing simultaneous program development, but each appears to be the sole user of the system.

The file management and terminal handling software were also improved so that when commands to execute a program are given, an individual copy of the program is created for the terminal. This allows many users to use what appears to be the same program, but each user is automatically using his or her own copy.

User program debugging was greatly improved with DBUGR, a new symbolic debug utility. When a program is relocated, the user may indicate the automatic addition of the debug capability to the program. When the program is scheduled, it will execute under control of DBUGR. DBUGR allows program modification, tracing, and breakpointing. In addition, registers and memory may be examined or modified in symbolic mode, ASCII mode, octal mode, or in any other numeric base. With the new multiterminal monitor, any number of users may be debugging with DBUGR independently and simultaneously.

Acknowledgments

Any project as large as RTE-IV requires the talent and effort of many people. Our thanks to Linda Averett, the initial project manager and designer, and to David Snow, who continued the project management to completion. Special thanks to Kathy Hahn, Shaila

Kapoor, Dean Drake, Jim Nissen, and Ralph Deadwyler for their design and development contributions to RTE-IV; to Gary McAnally, Bryon Look, and George Anzinger for their file management and languages contributions. Thanks also to Tom Engleman's quality assurance team, whose tenacious efforts resulted in a better product.

References

1. G.A. Anzinger and A.M. Gadol, "A Real-Time Operating System with Multi-Terminal and Batch/Spool Capabilities," Hewlett-Packard Journal, December 1975.
2. L.W. Averett, "Real-Time Executive System Manages Large Memories," Hewlett-Packard Journal, December 1975.
3. D.L. Snow and K.F. Hahn, "HP 1000 Operating System Is Enhanced Real-Time Executive," Hewlett-Packard Journal, March 1977.
4. R.R. Shatzer, "Distributed Systems/1000," Hewlett-Packard Journal, March 1978.
5. P.M. Sakakihara, "Distributed Systems/3000," Hewlett-Packard Journal, March 1978.
6. J.T. Schultz, "Real-Time Multi-User BASIC," Hewlett-Packard Journal, January 1976.
7. H.D. Drake, "Development and Application of Microprograms in a Real-Time Environment," Hewlett-Packard Journal, March 1977.
8. J.S. Elward, "The Million-Word Minicomputer Main Memory," Hewlett-Packard Journal, October 1974.

C. Michael Manley

Mike Manley, a three-year HP employee, designed and implemented the executive request processor, scheduling processor and dispatcher for the HP RTE-IV operating system. A 1973 BSEE graduate of Kansas State University, Mike received his MBA degree from Wharton School of Commerce and Finance at the University of Pennsylvania in 1975. While studying toward his master's degree, he worked part time in software and hardware design and testing. Born in Kansas City, Missouri, Mike served in the U.S. Army for three years, teaching electronics and working as a technician in Vietnam. Mike is married, lives in San Jose and spends his leisure time building television sets, repairing stereos and golfing.

Eugene J. Wong

Eugene Wong designed and implemented the dispatcher, input/output and parity error handling modules for the RTE-IV operating system. He received his BA degree in mathematics from San Francisco State College in 1970 and joined HP shortly after graduation. Eugene also designed and implemented the TODS system and several other memory-based and disc-based RTE systems. Born and raised in Stockton, California, Eugene is a member of the Association for Computing Machinery. In his spare time, he enjoys black and white photography, racquetball, playing the banjo and playing GO, a Chinese game with strategy somewhat like chess. Eugene lives in San Jose, California, is married and has two daughters, ages two and four.

HP 92067A Real Time Executive IV Operating System (RTE-IV)

FEATURES:

- Management of up to 64 disc-resident program partitions in up to 2.048 megabytes of memory
- Non-swappable memory resident programs
- Up to 56K bytes for user's program code, data, and base page linkages independent of physical memory used by the operating system and drivers
- User addressing of extended memory areas for data limited only by available memory (nearly 2 megabytes in a 2.048 megabyte system)
- Time, event, and program-to-program scheduling for real-time measurement, control, and/or automatic test applications.
- Support for choice of 4.9, 14.7, 19.6, or 50M-byte system disc, the latter expandable to 400M-byte capacity with additional 50M-byte disc drives
- Batch-Spool Monitor for concurrent disc file management and batch processing
- Concurrent execution and development of BASIC (optional), FORTRAN IV, and assembly language programs
- FORTRAN IV compiler support of user-transparent program access to large data arrays
- Interactive debug package and interactive editor to aid program development
- Optional RTE microprogramming package for on-line development and debugging of user-microprogrammed subroutines for faster data processing by the computer
- True multi-terminal program development in all program languages using input and output files
- Memory, partition, and I/O reconfigurability at boot-up
- Input/output spooling to disc to speed throughput with minimal use of main memory for buffering
- RTE drivers and device subroutines for supported peripherals included with the system
- Support of optional IMAGE/1000 Data Base Management System for more efficient use of data files, easier access to data
- Support of multiple instrument clusters connected via the Hewlett-Packard Interface Bus (HP-IB). The Hewlett-Packard Interface Bus (HP-IB) is Hewlett-Packard's implementation of IEEE Standard 488-1975, "Digital Interface for programmable instrumentation" and identical ANSI Standard MC1.1
- Support of optional DS/1000 software-firmware for communication with other HP 1000 Computer Systems and/or with HP 3000 Systems.

ORDERING INFORMATION

92067A-030 RTE-IV distributed on 7900 (2.5M-byte) disc cartridge.
92067A-031 RTE-IV distributed on 7906 (10M-byte) disc cartridge.
92067A-032 RTE-IV distributed on 7920 (50M-byte) disc pack.
92067A-050 RTE-IV 2.5M-byte disc cartridge image distributed on 800 bpi, 9-track mag tape.
92067A-051 RTE-IV 10 or 50M-byte disc cartridge image distributed on 800 bpi, 9-track mag tape.
92067A-052 RTE-IV 2.5M-byte disc cartridge image distributed on 1600 bpi, 9-track mag tape.
92067A-053 RTE-IV 10 or 50M-byte disc cartridge image distributed on 1600 bpi, 9-track mag tape.

The 92067A RTE-IV system is included in the 2176A/B and 2177A/B Computer System building blocks, which form the basis of the HP 1000 Model 40 and 45 Computer Systems.

PRICES IN U.S.A.:

92067A-030, $5100.
92067A-031, $5100.
92067A-032, $5500.
92067A-050, 051, 052, 053, $5000.

MANUFACTURING DIVISION: DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, California 95014 U.S.A.

F-Series Extends Computing Power of HP 1000 Computer Family

by Julia A. Cates

HP 1000 COMPUTERS are a modular family of powerful general-purpose 16-bit computers that feature user-microprogrammable central processing units (CPUs), fast and reliable semiconductor memory systems, and HP's broad range of real-time executive (RTE) operating systems.

The HP 1000 family began in 1974 with the introduction of the 21MX Computer,[1] later renamed the HP 1000 M-Series. Two years later, in 1976, HP announced the E-Series Computer.[2] This implementation of the HP 1000 architecture couples 30% faster technology with a technique of dynamically varying the basic machine cycle to provide twice the computing power of the M-Series.

In April 1978 a new higher-performance series, the F-Series (Fig. 1) was added to the HP 1000 family. F-Series Computers combine the basic processing unit of the E-Series with dedicated hardware for executing floating point instructions and instruction set extensions for extremely fast execution of transcendental functions and commonly-used FORTRAN operations. Thus the F-Series offers the most computational power of the HP 1000 Computer family and opens up many new applications. Floating-point-intensive programs that were compute bound on the E-Series Computer will run at least twice as fast on the F-Series. The scientific instruction set (see article, page 18), which accelerates any programs using transcendental functions, makes the F-Series the best choice for curve fitting, graphics, and circuit modeling programs. Using the F-Series' microprogrammable floating point processor, the F-Series handles many applications that otherwise would require a 32-bit computer.

Floating Point Processor

The foremost contribution to the F-Series' computational capability is the hardware floating point processor (FPP). This processor is a hardware implementation of existing HP 1000 floating point arithmetic instructions. The processor performs these floating point operations on 32-bit single-precision or 48-bit extended-precision operands represented in standard HP 1000 floating point number formats. Single-precision floating point instructions execute 2½ to 6 times faster than the E-Series firmware instructions. For example, 32-bit addition, which provides almost seven decimal digits of precision, executes in three to five microseconds. Multiplication, which makes up 40% of all floating point instructions in a typical mix, is accelerated by special FPP circuits that make it as fast as addition or subtraction (Fig. 2). The extended-precision 48-bit floating point instructions furnish more than eleven decimal digits of precision and run three to six times faster than the equivalent optional microcoded routines run on the E-Series Computer.

Besides executing the standard HP 1000 floating point instructions, the floating point processor executes user-microprogrammed floating point operations. Since the FPP communicates with the central processor via the microprogrammable processor port (Fig. 3), any microprogram can control the FPP. Because the microprogrammable processor port provides a direct link to the CPU data bus, data can be transferred to the FPP at burst rates up to 5.7 megawords per second.

Very efficient use can be made of the FPP through microprogramming. Floating point instructions executed from software incur memory overhead amounting to as much as 70% of the total instruction time, both in fetching the operand from memory and in storing the result after the operation has completed. Most memory overhead can be eliminated under microprogram control by overlapping memory accesses with floating point operations. The floating point processor also has an accumulator, accessible only at the microcode level, in which intermediate results may be stored for chained calculations. The accumulator can eliminate the processor overhead required to store a result in memory before retrieving it as an operand for the next FPP operation. Since the FPP is peripheral to the central processor, the user may access memory and perform any I/O or central processor operations while the FPP completes an operation. Overlapped CPU/FPP processing and the FPP's accumulator mode can speed up any microcoded floating point calculation.

The scientific instruction set (see article, page 18) dramatizes the increase in computational performance that the microprogrammable features of the floating point processor make possible. The scientific instruction set (SIS) consists of the nine most frequently executed scientific functions (sine, cosine, tangent, arc tangent, hyperbolic tangent, base ten logarithm, natural logarithm, exponent, and square

Fig. 1. *HP 1000 F-Series Computers come in two versions, one with floating point processor built in, and the other with more memory and I/O slots and the FPP in a separate unit. Both have high-performance memory, firmware scientific instruction set, and firmware fast FORTRAN processor.*

root). Microcoded to perform all floating point operations in the FPP, these instructions are 6 to 24 times faster than the E-Series software library functions. Also, the algorithms of the SIS have been refined so that the SIS provides substantially more accurate results than the previous functions in software.

Benchmark Performance

Just how powerful is the F-Series' combination of hardware and firmware enhancements? One way to measure the performance of computers is to run benchmark programs and compare execution times. Benchmarks are standardized programs whose execution times indicate processing capability. Fig.4 displays the results of running benchmarks that involve single and extended-precision floating point instructions and single and extended-precision transcendental functions. Created by the British National Physical Laboratory, these benchmarks are compute-bound FORTRAN programs that are designed to measure central processor performance rather than software compiler performance.

The program execution times listed in Fig. 4 indicate that the HP 1000 E-Series Computer has twice the processing capability of the M-Series in all four program types. Since the hardware of the E-Series CPU is only 30% faster than the M-Series hardware, the doubled performance demonstrates the effectiveness of the E-Series microinstruction enhancements.[3]

Next, consider the benchmark times of the E-Series and F-Series computers. The F-Series executes the

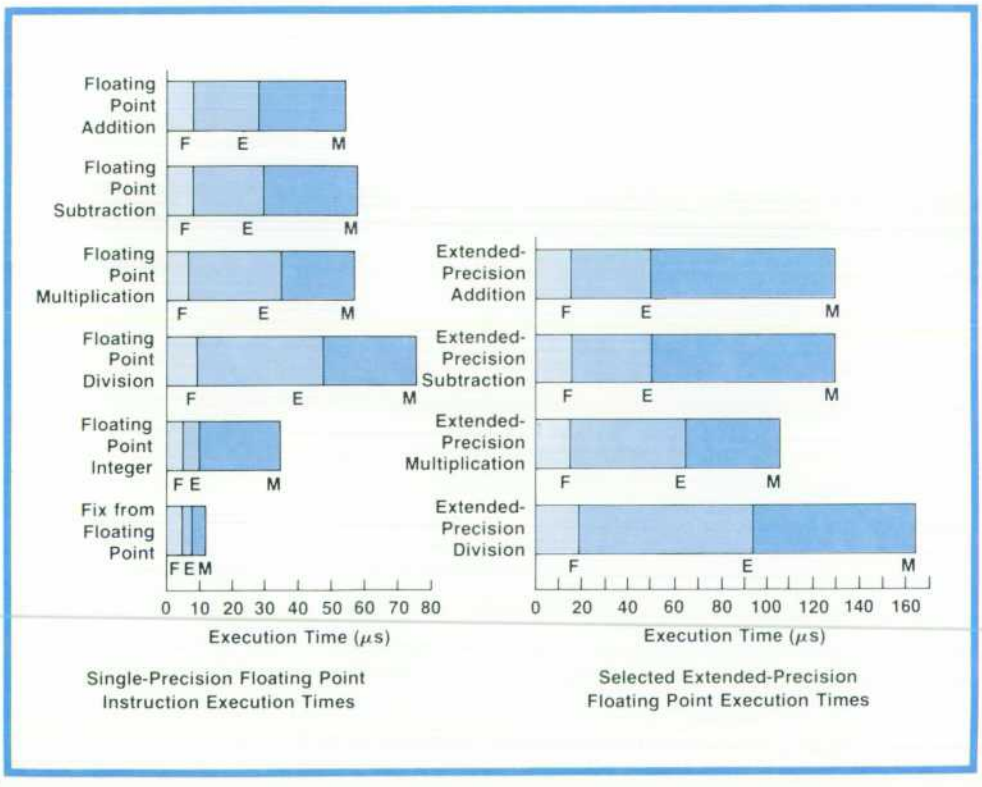


Fig. 2. *Execution times of floating point instructions on the M-Series, the E-Series, and the new F-Series HP 1000 Computers. The E-Series has a faster CPU than the M-Series. The F-Series has a hardware floating point processor and the same CPU as the E-Series.*

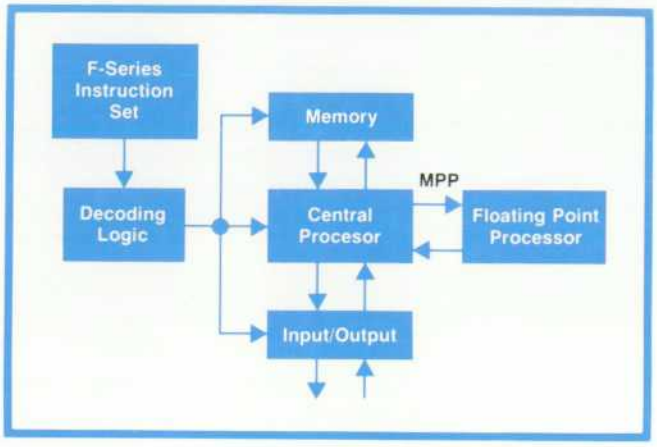


Fig. 3. *In the F-Series Computer, the floating point processor (FPP) is connected to the main data bus via the microprogrammable processor port (MPP). The FPP contains data and control logic dedicated to floating point operations; it relieves the central processor of the burden of the software or firmware floating point routines formerly used. The F-Series instruction set includes new firmware scientific instructions that make use of the FPP.*

single and extended-precision floating point programs 2½ times faster than the E-Series. Since the E-Series and F-Series Computers have the same central processor, this speed improvement illustrates the power of processing floating point instructions in hardware instead of firmware. Most impressive are the execution times for the 32-bit transcendental functions benchmark program. Here the F-Series is almost eight times faster than the E-Series Computer. Again, any application can attain a similar increase in performance by microcoding floating-point-intensive routines to use the FPP. The execution times for the 48-bit transcendental functions points out that any E-Series program involving the standard floating point instructions can execute twice as fast on the F-Series. However, when floating-point-intensive routines are microcoded, these programs run seven times as fast on the F-Series.

Floating Point Numbers

Before discussing floating point arithmetic, a quick review of floating point numbers is in order. Floating point numbers consist of a mantissa or fraction multiplied by two raised to an exponent or power. HP 1000 single-precision floating point numbers have a 23-bit-plus-sign mantissa that is multiplied by two raised to a seven-bit-plus-sign exponent. Extended-precision numbers have a 39-bit signed mantissa and the same seven-bit signed exponent. All mantissas are normalized, which means they are in the ranges $[½, 1)$ and $[-1, -½)$. In addition or subtraction, the arithmetic operation cannot take place until the exponents of the operands are equal. The exponent equalization process increments the smaller exponent while shifting right, or halving, the corresponding mantissa until the two exponents are equal. After any operation, if the result is not normalized, it is shifted left, or doubled, while its exponent is decremented until it is in the proper range.

This use of the term "floating point" is actually improper, since the radix point (decimal point, binary point, etc.) is fixed at the left side of the mantissa. An unambiguous term for this type of number representation is "scientific notation." The term "floating point" is often used for a free-field number representation in which the radix point can appear anywhere in the field; hence, it "floats." This latter use of "floating point" is common in the handheld calculator literature. However, the former use, as described above, is common in the computer field, and so it is used in the articles in this issue despite the unfortunate potential for confusion.

FPP Algorithms

One of the primary design goals of the floating point processor was to implement the operations add, subtract, multiply, divide, fix to integer, and float from integer with a minimum register and data path configuration that would fit on a single printed circuit board.

Floating point addition and subtraction algorithms

Systems	M-Series Standard Memory Fast FORTRAN Processor RTE-IV	E-Series High-Speed Memory Fast FORTRAN Processor RTE-IV	F-Series High-Speed Memory RTE-IV
Benchmarks			
WHETSP	2.02	1.03	.40
WHETDP	3.13	1.81	.86
TRANS SP	5.75	3.0	.40
TRANS DP	12.12	6.86	2.64
FLOAT SP		1.63	.58
FLOAT DP		2.91	.93

Fig. 4. *Comparative performance of the three series of HP 1000 Computers is shown by relative execution times of benchmark programs. All of these benchmarks are compute-bound FORTRAN programs that require only a small amount of memory. The WHETSTONE benchmark was designed to represent a typical FORTRAN program with an average floating point mix. It was coded in FORTRAN using the WHETSTONE algorithm created by the National Physical Laboratory in England. The algorithm represents an instruction mix derived from analysis of about one thousand ALGOL 60 programs. WHETSP and WHETDP are single-precision and extended-precision versions of this benchmark. The TRANSSP and TRANSDP benchmarks perform transcendental calculations; TRANSSP is the single-precision version and TRANSDP is the extended-precision version. They make extensive use of the square root, sine, cosine, arc tangent, and exponential functions in the F-Series scientific instruction set. FLOATSP and FLOATDP perform FORTRAN floating point calculations; FLOATDP is the extended-precision program.*

involve shifting mantissas right to equalize exponents, adding or subtracting mantissas, and then shifting the result left to normalize it. Thus the minimum hardware configuration had to include bidirectional shift registers and arithmetic logic units (ALUs). In light of this requirement, the subsequent algorithm investigation focused on multiplication and division algorithms that consist of sequences of shift cycles and arithmetic cycles.

The multiplication algorithm shifts over strings of zeros and ones while detecting and correcting for isolated zeros or ones. To understand what this means, first consider the simplest type of multiplication. The two operands, the numbers to be multiplied, are called the multiplier and the multiplicand. The simplest multiplication algorithm scans the multiplier and adds a copy of the multiplicand to the partial product at each "1" bit position of the multiplier. Observe that a bit pattern in the multiplier of ...100001... is equivalent to multiplying by $2^{n+5} + 2^n$. Also, ...0111110... is equivalent to $2^{n+4} + 2^{n+3} + 2^{n+2} + 2^{n+1} + 2^n$ or, more importantly, $2^{n+5} - 2^n$. Note that one addition and one subtraction can replace four additions. Since any multiplier can be reduced to strings of ones and zeros, multiplication can be a process of add or subtract cycles and shift cycles. FPP shift cycles take only 50 nanoseconds while arithmetic cycles take 125 nanoseconds, so the goal in designing the algorithms was to perform as few ALU cycles as possible.

With this in mind, what happens in the sequence ...0001000...? If the lone one is treated as a string, the above method dictates $2^{n+1} - 2^n$, or one addition and one subtraction. Obviously, one addition should suffice. However, if a history bit[4] is used to indicate the type of string that is being shifted over, the isolated bit can be detected, and the single addition will be performed.

Since the multiplication algorithm calls for an arithmetic cycle only at the start and the end of strings and once at isolated bits, the processor will never perform two consecutive arithmetic cycles. Thus, arithmetic cycles are always followed by shift cycles. Also, each arithmetic cycle includes a shift operation. This means that each time a partial product passes through the arithmetic circuits, it is shifted twice (see Fig. 5). The FPP accomplishes this double shift instantly, by means of multiplexers at the output of the arithmetic circuitry. With the multiplexers, every arithmetic cycle eliminates two shift cycles. This is a key factor in the speed of the new F-Series floating point processor.

Since the bit sequence of the multiplier dictates the sequence of arithmetic and shift cycles, every operation of the multiplication process can be predicted at the start of the process. Therefore the FPP initiates the next operation while the current cycle is completing. This look-ahead technique and the double-shifting multiplexers make typical multiplication execution times almost as fast as addition times.

The division algorithm, as in multiplication, shifts over strings of ones or zeros and adds or subtracts the divisor to or from the dividend. However, unlike multiplication, there are no look-ahead techniques for reducing the number of cycles required to form the quotient. Also, since division algorithms form a one's complement version of the quotient, negative quotients have to be incremented to the two's complement form after the division process. Most hardware-implemented two's complement algorithms do not round quotients properly, since the remainder is thrown away. However, the FPP develops one extra quotient bit, which is the bit to the right of the least significant bit (LSB). Using this bit and the negative quotient correction cycle, the FPP always rounds the quotient correctly.

Not only are the results of division rounded properly, but also the results of addition, subtraction, multiplication and fix-to-integer are checked for rounding. The round circuitry (see box, page 16) uses three guard bits, which represent the three bits just to the right of the LSB, and a sticky bit, which indicates if there are any ones to the right of the guard bits. The sticky bit is set by any ones that are right-shifted out of the guard bits. Since the least significant bit position depends on the precision of the operation, multiplexers are used to shift the appropriate LSB into the guard bit register on shift operations. Other multi-

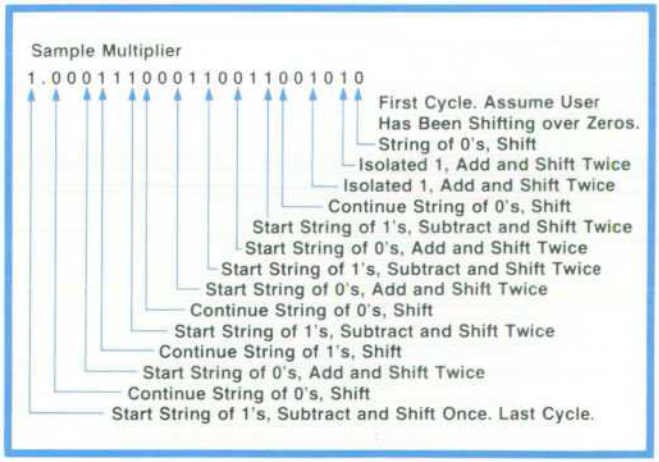


Fig. 5. *Floating point multiplication in the F-Series floating point processor executes almost as fast as addition. The multiplication algorithm calls for an arithmetic operation (addition or subtraction) only at the start and end of a string of zeros or ones or at isolated ones in the multiplier. As shown here for this sample multiplier, this means that each arithmetic cycle is followed by two shift cycles. In the FPP this double shift is performed by hardware multiplexers, thereby eliminating the shift cycles and saving a great deal of time.*

F-Series Rounding Techniques

To minimize error propagation in a floating point calculation, each floating point operation must produce results that are as accurate as possible. Some operations generate results that have more than 23 or 39 bits. For instance, multiplication of two 23-bit mantissas generates a 46-bit product. The excess bits are used to decide whether to truncate or round the result to form a proper-length mantissa. Rather than use expensive double-length registers, the HP 1000 F-Series floating point processor (FPP) holds information about the extra bits in a single four-bit register (Fig. 1).

This rounding information register holds three guard bits, which represent the three bits to the right of the resulting mantissa's least significant bit (LSB). The round decision also uses a sticky bit that indicates whether there are any 1's in the bits to the right of the guard bits.

The round decisions for positive and negative operands differ. Positive operands are rounded if the first guard bit is a one. Negative operands are rounded if the first guard bit is a one and there is one more one in the bits to the right of the first guard bit.

Although the FPP maintains a single round register, rounding information is routed to the register in four ways. First of all, as operands are shifted right, the bit from the LSB position is shifted into the round register. Since the LSB position depends on the precision of the floating point operation, multiplexers are used to shift either the twenty-third or thirty-ninth bit into the first guard bit.

A refinement is made on the shift-right-multiplex-LSB process in the subtraction case, where the subtrahend is undergoing exponent equalization. In subtraction, since the subtrahend is complemented and then added to the minuend, the subtrahend bits entering the round register have to be complemented. One method of forming the two's complement of a binary number is to start at the right end of the number and move left until a 1 is encountered, leaving all zeros as they are, then taking the one's complement of all the bits to the left of the first 1. When the first 1 is encountered, the FPP sets a latch that causes all succeeding bits to be complemented. In this way, the round register effectively maintains a complemented subtrahend.

Multiplication sets up the round register in a third way. Since the partial product is shifted twice to the right during ALU cycles, the two LSBs must be loaded into the round register. Again, because the LSB position depends on the precision of the operation, a second set of multiplexers is used to sort out the proper LSBs for the round register.

In contrast to the above operations, division uses different information in its rounding decision. Its decision is based on the sign of the quotient and the first guard bit so the division process develops one extra quotient bit, which is loaded into the first guard bit. After the quotient and round register are adjusted for mantissa overflow or normalization, the quotient is rounded if the first guard bit is a 1.

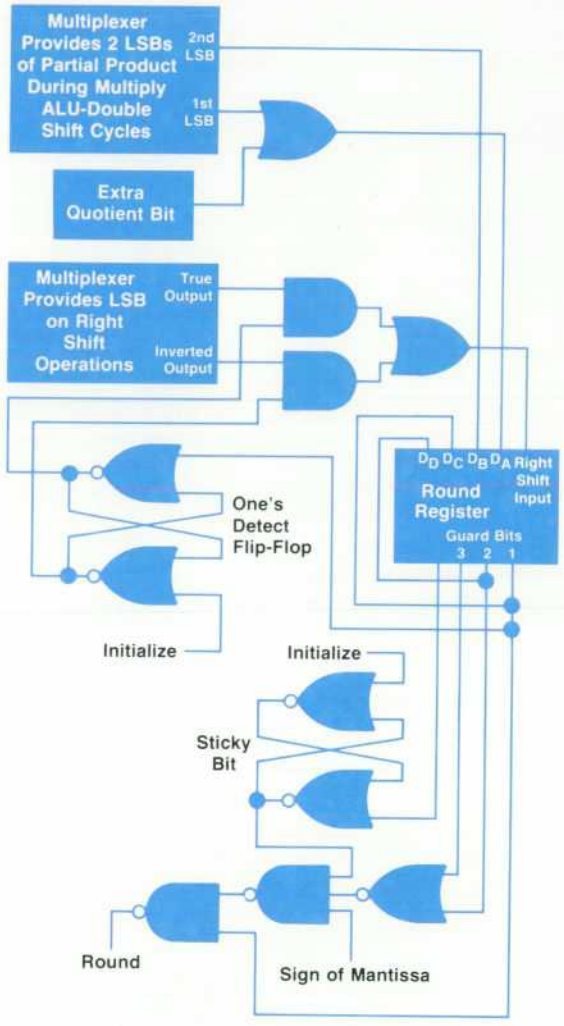


Fig. 1. *Floating point processor rounding circuits. Depending on the operation, rounding information is set up in the round register in four ways: the least significant bit may be shifted into the round register through a multiplexer, subtrahend bits may be complemented as they are shifted into the round register, two partial product bits may be loaded into the round register, or a quotient bit may be loaded into the first guard bit of the round register.*

plexers route appropriate least significant bits of the partial product to the round register during multiplication. Thus in all operations, the round circuits maintain sufficient information to properly round floating point results.

Control of floating point operations is directed from a state machine of 60 states. Since all of the FPP operations are sequential, a state machine, in which control flows from one state to the next, is the best implementation. To make ALU cycles and shift cycles as short as the hardware circuits permit, the state machine consists of shift registers clocked at 40 MHz. Thus, a particular state is active for only 25 ns. It turns out that a state machine implementation, in which control flows only in a distinct pathway, makes most FPP component failures easy to troubleshoot despite the complicated processes some operations demand.

Impact on System Environment

A major design goal for the floating point processor was compatibility with existing software. To help achieve this goal, the FPP preserves the floating point

number representations and instruction formats of the HP 1000 Computer. To minimize control store requirements, the F-Series microcode required for all of the standard FPP instructions resides in the control store module that holds the firmware 32-bit floating point instructions on the E-Series Computers. Thus, since the F-Series is able to maintain the 32-bit instruction codes, programs using these instructions run on the F-Series without being recompiled or even reloaded into the user's software system.

However, since the F-Series provides new instruction codes for extended-precision instructions, programs written for the M- and E-Series Computers that use extended precision must be reloaded in the user's software system environment before these programs can be executed (the loader generates the machine code that is actually executed). However, these programs need not be recompiled, because they use the same calling sequences as the E-Series instructions. Similarly, programs using the scientific instruction set must be reloaded into the system.

A major problem in adding hardware to a system, especially from the product support viewpoint, is the incremental complexity in tracking down system failures to a failing component. The project team paid particular attention to this area in designing the floating point processor. The software diagnostic exercises every logic circuit pathway and calculates each result using a software simulator. The diagnostic checks each type of bit pattern that can be input to the processor using direct and indirect memory references for operands. The firmware module for the standard floating point instructions contains special-purpose microcode that the diagnostic uses to direct accumulator or expanded exponent operation tests. The scientific instruction set resides in two firmware modules, and the diagnostic is able to isolate SIS firmware errors to the particular failing module. As a measure of the complexity of operations possible with the FPP and of the effectiveness of the diagnostic, one pass of the diagnostic involves 1,000,000 different floating point operations.

Besides providing a software diagnostic to pinpoint FPP failures, the F-Series has firmware tests written in microcode that verify FPP and SIS installation. These tests, which are run from the computer's front panel, verify that the FPP firmware modules are installed, that the FPP has power, and that the FPP-CPU interface cable is operational.

What happens when failures occur in the operating system environment? The firmware for the standard floating point instructions is able to detect major FPP failures, such as loss of power or interface cable, and report these failures to the RTE operating system. RTE handles these failures in the same way it responds to memory parity errors. When FPP errors are detected, RTE aborts the program using the FPP and reports to the system console the name of the program that was aborted, the instruction of the failed FPP operation, and its memory address. This protects the user from accepting invalid results from a failing FPP.

Acknowledgments

The successful introduction of the F-Series Computer was accomplished only through the special efforts of several people. As project manager, Cle Riggins coordinated the efforts of quality assurance, technical support, production engineering, and project management. Dave Kuykendall worked on the initial FPP design and designed the FPP power supply. Craig Chatterton made key contributions to the F-Series in the following areas: powerful, all-inclusive diagnostic firmware for the standard floating point instructions, special-purpose microcoded diagnostic aids, and particular attention to the project in the production prototype phase. Many thanks go to section manager Bob Frankenberg for his support of the F-Series and to Steve Sheafor and Greg Hansen who performed the algorithm investigation. In sum, the enthusiasm and special efforts of the entire project team made the F-Series a significant contribution to the HP 1000 product line.

References

1. J.M. Stedman, "A User-Oriented Family of Minicomputers," Hewlett-Packard Journal, October 1974. Also other articles in same issue.
2. C.C. Riggins, "E-Series Doubles 21MX Computer Performance," Hewlett-Packard Journal, March 1977. Also other articles in same issue.
3. S.J. Stallard, "How the E-Series Performance Was Attained," Hewlett-Packard Journal, March 1977.
4. I. Flores, "The Logic of Computer Arithmetic," Prentice-Hall, Inc., 1963, p. 196.

Julia A. Cates

Julie Cates is a BSEE and MSEE graduate of Stanford University. An HP employee since graduation in 1974, Julie was project leader for the HP 1000 F-Series computer and worked on the F-Series floating point processor. She also wrote the diagnostic for the 6940 multiprogrammer and designed the signal conditioning circuits for the 9600 measurement and control systems. Born in Pasadena, California, Julie now is in computer product management. She is married, lives in Los Altos, California, and enjoys all sports, particularly volleyball, running and tap dancing. Julie is also a novice airplane pilot and enjoys traveling to various parts of the world.

Microcoded Scientific Instruction Set Enhances Speed and Accuracy of F-Series Computers

by Charles R. Geber

THE SCIENTIFIC INSTRUCTION SET of HP 1000 Computers consists of nine transcendental functions that find extensive use in scientific and engineering applications. These functions are sine, cosine, tangent, arc tangent, exponential, square root, natural and common logarithms, and hyperbolic tangent (SIN, COS, TAN, ATAN, EXP, SQRT, ALOG, ALOGT, and TANH). In earlier HP 1000 Computers, these functions have been offered as software library routines. In the new F-Series, they have been microcoded, making them part of the computer's standard instruction repertoire and therefore directly accessible by FORTRAN, BASIC, and assembly language programs. Microcode implementation, taking advantage of the computational capability of the F-Series' floating point processor (see article, page 12), provides dramatic increases in speed and accuracy over the corresponding software routines.

The increase in execution speed provided by microcoded algorithms can be attributed to two factors. First, the access time of the microcode control store is far less than that of the main memory, where macro (software) instructions are stored. Second, the execution of microinstructions takes place at a much higher rate than that of macroinstructions. The new scientific instruction set provides a 6-to-24-fold improvement in execution speed over the equivalent software routines, enabling the F-Series Computer to equal the performance of many large mainframe computers in the execution of these functions.

In the evaluation of transcendental functions, finite algorithms are used to approximate the function values. Such algorithms usually trade off execution speed for accuracy. Because microprogramming makes this trade-off less apparent, the new scientific instruction set achieves significant performance in both of these categories.

Algorithm Speed Optimization

Various algorithms for the SIS (scientific instruction set) functions were first implemented in software. After the assembly-language versions had run successfully, the flowchart was implemented in microcode. In this conversion, algorithm steps were sometimes modified or reordered to take advantage of hardware capabilities available only to the microprogram. For example, a combination of logical and shift functions in one microinstruction could often replace two or three software steps.

The software simulation made it easier to debug the algorithms and provided an early indication of accuracy and relative speed for competing strategies. Evaluation of the software trials indicated several techniques for algorithm speed optimization. One important technique is computation step minimization. A simple reorganization of a formula can often lead to a significant decrease in execution time. For example, the definition of the TANH function is usually stated as:

$$\mathrm{TANH}(X) = \frac{e^{X} + e^{-X}}{e^{X} - e^{-X}} \qquad (1)$$

Since the EXP function is included in the SIS, the TANH routine could be implemented by a literal application of equation 1. However, this would involve two calls to the EXP function, each requiring about 45 microseconds.

An equivalent form of equation 1 can be found by multiplying the numerator and denominator of the right-hand side by e^{X}, yielding:

$$\mathrm{TANH}(X) = \frac{e^{2X} + 1}{e^{2X} - 1} \qquad (2)$$

With equation 2, only one call to EXP is required. One addition, one subtraction, and one division will then yield TANH(X).

Function domain segmentation is another technique for optimizing the speed of an algorithm. Often the algorithm can be greatly simplified over certain segments of the function's domain. In the case of TANH (see equation 1), for large positive values of X the leading terms of both numerator and denominator will dominate, and TANH(X) will approach +1. Similarly, for large negative X the function value will approach −1. For a given large X, the error in approximating TANH(X) by 1.0 is simply 1−TANH(X). When this error becomes less than the achievable precision in computing equation 1, the approximation is acceptable. For the single-precision number

representation used in the SIS, a lower bound of ABS(X)>8 was determined, where ABS(X) is the absolute value of X. Therefore, the microcode first segments the domain of TANH into the areas ABS(X)≤8 and ABS(X)>8, and applies the extremely simple algorithm TANH(X) = 1*SIGN(X) to the latter case. Function domain segmentation is also applied to error reduction, as described later.

Processor Overlap

Microprogramming techniques were also applied in the design of the SIS to achieve maximum computation speed. The most significant contribution comes from efficient use of the processing overlap capability of the central processor (CPU) and the floating point processor (FPP). The sequence of microcode operations necessary to perform a floating point calculation with the FPP is:

1. Send opcode (add, multiply, etc.) to FPP
2. Send FPP start signal
3. Send operands to FPP
4. Wait for FPP completion signal
5. Retrieve answer from FPP.

The time spent in the fourth step is determined by the FPP hardware and is a function of the desired operation and precision. The actual time of FPP execution varies between 0.6 and 5 microseconds for functions required by the SIS. FPP completion is determined by repeatedly testing a flag from the FPP.

While the FPP is computing, the CPU of the F-Series Computer is free to accomplish other computation tasks for the SIS. It is in this waiting period that the SIS microcode gains its major speed advantage over its software counterpart. Operations performed in the SIS during FPP execution include coefficient generation, subroutine linkage, and algorithmic decision making.

Coefficient generation. Polynomial coefficients are generated with immediate micro-operations. In general, six microinstructions (1.05 microseconds) are required per coefficient. Since SIS algorithms typically contain four to six coefficients per function, processor overlap contributes a 10-15% speed enhancement for coefficient generation alone. It should be noted that these coefficients could be stored in main memory and fetched when needed, significantly reducing the length of the SIS microcode. The disadvantage of this technique is that the SIS functions would no longer be directly callable from FORTRAN, but would have to link via assembly language subroutines. The desire for maximum performance (and the availability of microcode space) prompted the decision to microcode the generation of coefficients.

Subroutine linkage. The F-Series processor allows a nesting of three levels of subroutines. Microsubroutines, like their software counterparts, reduce the overall code requirements by consolidating repeatedly performed functions. With both software and firmware subroutines, however, the call and return linkage to the routines adds an execution overhead that usually results in longer overall execution times compared to in-line coding. By handling this linkage while the FPP is processing, the SIS is able to conserve code space with virtually no performance degradation.

Algorithmic decision making. The SIS often performs branching within the algorithms, based on the initial operand or intermediate results. Such decisions, which may require several microseconds of computation, are performed in the FPP execution interval and a flag is then set to indicate the result of the decision. After the current FPP operation has completed, the flag can be quickly tested and the algorithm continued via the appropriate path.

Accumulator Operations

One of the extremely useful features of the F-Series floating point processor is its accumulator-mode operation. In this mode, the FPP can use the result of a previous operation as one (or both) of the operands for a successive calculation. Thus, to evaluate the expression:

$$X = A(B+C) \tag{3}$$

the microcode would first instruct the FPP to add B to C, leaving the result in the FPP accumulator. This value would then be multiplied by A, the answer retrieved by microcode, and stored in X. The key point is that the intermediate result (B+C) need not be retrieved; thus the multiplication is accomplished by sending only one operand to the FPP. For single-precision operands, it takes 350 nanoseconds to transfer a value between the CPU and the FPP. Thus in equation 3, accumulator operation saves 700 nanoseconds. In a chain of several calculations the execution-time savings can be quite significant.

To use this calculation mode effectively, SIS algorithms had to be put into a suitable algebraic form. Consider the general polynomial:

$$P = C_0 + C_1X + C_2X^2 + C_3X^3 \tag{4}$$

Equation 4 can be rearranged to form a new but equivalent value:

$$P = C_0 + X(C_1 + X(C_3X + C_2)) \tag{5}$$

Equation 5 is ideal for accumulator chained calculation. After the initial two-operand calculation of C_3X, each successive computation uses the result of the preceding one as an operand.

Execution Times

	SIN	COS	TAN	ATAN	SQRT	EXP	ALOG	ALOGT	TANH
t_E E-Series	289.9	314.4	978.8	1044.8	197.2	362.2	301.2	334.5	471.5
t_F F-Series	47.6	47.9	48.4	42.4	30.9	44.7	43.3	49.4	57.2
$\frac{t_E}{t_F}$	6.1	6.6	20.2	24.6	6.4	8.1	7.0	6.8	8.2

(All Times in Microseconds)

Fig. 1. *Execution times for scientific functions for the HP 1000 E-Series and F-Series processors. Both computers have the microcoded fast FORTRAN processor and high-performance memory. The F-Series has the new microcoded scientific instruction set. Times for each function are averaged over a range of typical input values. The exceptional performance increases for* TAN *and* ATAN *are due primarily to an algorithm change from Chebychev polynomials to rational forms.*

The combination of these speed enhancement techniques results in extremely fast execution times for the SIS functions. Fig. 1 compares the speed of the HP 1000 F-Series SIS to E-Series software execution of transcendental functions. The exceptional performance increases of 20 and 24 times in TAN and ATAN are results of the speed enhancement techniques just described and of algorithm changes. The earlier software routines approximated these functions using Chebychev polynomials, while the new SIS approximates all of the transcendental functions using ratios of polynomials of the type shown in equation 5.

Accuracy Enhancement

In addition to the design objective of fast execution speed, the SIS is required to produce results with the high level of accuracy needed in engineering and scientific applications. One key contribution to this goal came from the field of numerical analysis, in the area of coefficient optimization.

As stated earlier, the SIS evaluates transcendental functions using rational forms (ratios of two polynomials). A FORTRAN program was used to calculate the coefficients that would yield the best approximation. Using Remes second algorithm,[1] the program first inputs the desired polynomial degrees and the domain of the approximation. Coefficients are then calculated to produce the least maximum relative error over the approximation interval. Relative error is defined as:

$$RE = \frac{(\text{Actual value}) - (\text{Approximated value})}{\text{Actual Value}} \qquad (6)$$

Increasing the polynomial degree will give a better curve fit and hence more accurate results, but the additional terms will obviously increase the execution time of the final algorithm. Relative error can also be thought of in terms of the number of correct bits in the final SIS answer. The proper trade-off occurs when the relative error of the approximation is about equivalent to the precision of the floating-point mantissa representation, which for the SIS is 23 bits, or about seven decimal digits.

The polynomial coefficients calculated by the FORTRAN program were used in the software algorithm simulations, and finally microcoded into the SIS.

Reduction Routine

The rational polynomial approximations just described are optimized over a given interval. For function domain values outside this region, the approximation's relative error will typically be excessive. Fig. 2 shows the relative error for the polynomial approximation of a trigonometric function such as SIN(X).

From Fig. 2, it's obvious that the polynomial P(X) has been optimized over the interval $(-\pi/4, \pi/4)$. Therefore, before SIN(X) can be calculated, the value X must first be mapped into this region. After the polynomial is evaluated using this new value, a correction factor is applied to reflect the mapping.

The process of mapping X into this interval is called

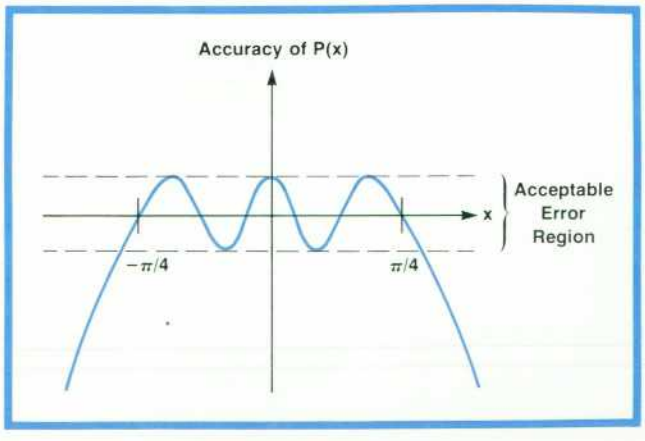


Fig. 2. *The new scientific instruction set approximates scientific functions using ratios of polynomials. The approximations are valid over a given interval, and outside this interval the relative error is excessive. This is a typical relative error curve for an approximation valid over the interval $-\pi/4, \pi/4$. Before a function can be computed, it must be mapped into the valid region, a process called reduction.*

RMS Relative Error

	SIN	COS	TAN	ATAN	SQRT	EXP	ALOG	ALOGT	TANH
SIS	8.8E−8	8.8E−8	2.0E−7	1.3E−7	6.7E−8	1.4E−7	1.3E−7	1.4E−7	1.3E−7
Software Library	1.7E−4	5.5E−6	6.6E−4	1.0E−7	6.7E−8	2.8E−7	2.6E−5	1.7E−5	2.3E−7
Improvement Factor	1932	62	3300	.77	1	2	200	121	1.7

Fig. 3. *Rms relative errors for the new scientific instruction set and the equivalent software library routines. Rms errors were computed for several thousand points distributed over a typical range of input values. Note that the precision of an individual single-precision number is about 1×10^{-7}. The marked accuracy increase for SIN and TAN is the result of an enhanced reduction routine. Although the high-speed SIS ATAN algorithm results in slightly poorer error performance, the resultant rms relative error is close to the floating point precision, and thus is considered acceptable.*

reduction. A reduction subroutine in SIS performs this function for SIN, COS, TAN, and EXP. The general formula for reducing X is given by:

$$R(X) = X - KC \qquad (7)$$

where $C = \pi/2$ for SIN, COS, TAN, and $C = 2/\ln(2)$ for EXP. K is chosen to place R(X) within the valid approximation region. The reduced value R(X) is then applied to the approximation polynomials.

There are two situations where the reduction process can drastically alter the accuracy of the final result. The first occurs when the input value falls near a multiple of C. For example, suppose it is desired to compute SIN(53.4). In this and the following example, we will assume that the single-precision floating point format used by the SIS contains seven significant decimal digits.

Equation 7, with $C = \pi/2$ and K = 34, yields:

$$R1(53.4) = (53.40000) - (53.40708) = -.0070800 \qquad (8)$$

The exact reduced value using ten decimal digits would have been:

$$R2(53.4) = (53.40000) - (53.40707512) = -.00707512 \qquad (9)$$

Applying Equation 6 to find the relative error in R1(X):

$$RE = \frac{-.00707512-(-.00708)}{-.00707512} = -.00069 \qquad (10)$$

The number of accurate bits left in R1(X) is equal to $-\log_2(RE) = 10$. Thus, even before the approximating polynomial is applied, the final answer is doomed to contain no more than three accurate digits.

A similar problem occurs when large values are reduced. If we wish to compute SIN(19000), the reduction with K = 12096 and $C = \pi/2$ becomes:

$$R3(19000) = (19000.00) - (19000.35) = -.3500000 \qquad (11)$$

The exact value using ten decimal digits would have been:

$$R4(19000) = (19000.0) - (19000.35237) = -.35237 \qquad (12)$$

The relative error in R3 is found to be:

$$RE = \frac{-.35237 - (-.35)}{-.35237} = .0067 \qquad (13)$$

The relative error in equation 13 corresponds to seven remaining bits of accuracy. Thus SIN(19000) would contain at best two accurate decimal digits.

To eliminate these large relative errors from the reduction routine, the SIS firmware simply extends the precision of the reduction operation. The extended precision format of the FPP provides an additional 16 bits in the mantissa, yielding a precision of about 11 decimal digits. The accuracy loss of 4-5 decimal digits in the previous examples would still leave the reduced R(X) with as many significant digits as the original X.

The use of increased precision in the reduction routine adds about two microseconds to the function execution time, but the exceptional increase in overall accuracy is well worth the cost. It should be noted that floating point values in the extended precision format require three 16-bit quantities instead of two. Thus, in software, the relative increase in execution time resulting from this technique would be higher because of the memory access overhead. The microcoded SIS, however, stores the intermediate result in high-speed CPU scratchpad registers, and therefore suffers only a minor speed penalty to achieve the greatly improved accuracy.

Function Domain Segmentation for Accuracy

Referring back to the definition of TANH(X) in equation 2, it can be seen that for small values of X the quantity e^X will be close to 1.0, so the subtraction in the denominator will result in the cancellation of

many significant bits of accuracy.

To correct this situation, the domain of TANH(X) is once again segmented to eliminate an interval of X values from the algorithm expressed in equation 2. Relative error analysis indicates that for ABS(X) <0.5 the cancellation in the TANH algorithm becomes excessive. Therefore, for values of X in the range (−0.5, 0.5) the SIS TANH routine uses a separate rational polynomial that is optimized over this interval. Thus the SIS microcode segments the domain of TANH twice: once to maximize the computation speed, and again to maintain a high level of accuracy in the result.

Fig. 3 indicates the accuracy enhancement of the SIS over the corresponding software library routines.

Acknowledgments

Several people made major contributions to the development of the scientific instruction set. Bill Gibbons, our resident numerical wizard, supplied the algorithms and optimized coefficients used in the SIS. Bill Baumann, a summer employee from Lehigh University, did an initial pass at the microcode, supplying many of the routines that communicate with the floating point processor. Dan Zuras performed an exhaustive quality assurance test of SIS accuracy. And FPP designer Julie Cates often did an incredibly tactful job of pointing out microcode bugs to a programmer who was sure that "the hardware must be getting the wrong answer."

Reference

1. J.F. Hart et al., "Computer Approximations," Wiley, 1968.

Charles R. Geber

Chuck Geber wrote the microcode for the scientific instruction set of the HP 1000 F-Series computer. Before joining HP in 1976, he worked as a development engineer in digital data communications. Chuck is a 1973 BSEE graduate of Cornell University and 1974 MSEE graduate of University of California at Berkeley. He was born in Hagerstown, Maryland, is single and lives in Santa Clara, California. Scuba diving, restaurant exploring and wine tasting are some of Chuck's interests.

Chuck, who bears some resemblance to film director/actor Woody Allen, also enjoys acting, and played Allen Felix (Woody Allen) in a local theater production of Allen's "Play It Again, Sam".

SPECIFICATIONS
HP 1000 F-Series Computers

FEATURES:

- High performance for computation intensive applications is provided by a combination of a high-speed central processor, a high-speed floating point processor, and a new set of instructions that speed up processing in scientific and industrial computer applications.
- The high-performance floating point processor is dedicated to floating point operations. The processor works with both single-precision (32-bit) and extended-precision (48-bit) numbers, and performs a single-precision multiply in 6.2 microseconds.
- The scientific instruction set is a set of nine instructions for extremely fast computation of trigonometric and logarithmic functions. SIN(X), for example, requires less than 52 microseconds.
- The fast FORTRAN processor is a set of instructions that greatly accelerates FORTRAN operations by performing such jobs as array address calculations at hardware speed.
- Powerful HP 1000 architecture and base instruction set feature variable microcycle timing for optimum price/performance.
- High-performance main memory, featuring a cycle time of 350 nanoseconds, is standard; 64K bytes in the 2111F, 128K bytes in 2117F. Fault control capability is optional.
- Dynamic mapping system, optional in 2111F, standard in 2117F, provides for accessing up to 2 megabytes of memory (1.8 million bytes with fault control) in 2117F computer plus extender.
- High-speed direct memory access is available via the dual-channel port controller, with transfer rates up to 2.3 million bytes per second.
- Fully user microprogrammable; complete microprogramming support software is available. Floating point processor is available as a computing resource to the microprogrammer.
- Two models to choose from:
  2111F, with space for up to 640K bytes of memory and nine I/O channels in 12¼ inch mainframe.
  2117F, with space for up to 1280K bytes of memory and fourteen I/O channels, in 17½ inches of panel space.
- Auto bootup and remote program load capability.
- Self test for CPU and memory.
- Disc loader program, contained in non-volatile read-only memory, is standard.

CENTRAL PROCESSOR: The central processor is microprogram controlled and is also microprogrammable. Microprogrammability fully software supported.
ADDRESS SPACE: 65,536 bytes; 2,097,152 bytes with dynamic mapping subsystem (DMS).
WORD SIZE: 16 bits.

SYSTEM CYCLE TIMES (All cycle times in ns):

	Cycle	Minimum	Typical	Maximum
High-Performance Memory	Read w/o DMS	350	350	385
	Read w/DMS	385	420	455
	Write	350	350	385
	Refresh	350	350	385
High-Performance Fault Control Memory	Read w/o DMS	386	421	456
	Read w/DMS	456	491	526
	Write	386	421	456
	Refresh	386	421	456

BASE SET INSTRUCTIONS: 156 standard instructions including index register instructions, bit, byte and word manipulation instructions, extended arithmetic instructions, and floating point instructions, scientific instructions and fast FORTRAN instructions, plus 38 dynamic mapping instructions (HP 2117F).
DATA REGISTERS: 2 accumulators, 2 index registers.
INITIAL BINARY LOADERS: ROM resident; capacity of four 64-word programs callable from operator panel. Computer can be configured for forced cold loading from a remote site.
SELF-TEST: Automatic tests of CPU and memory operating condition. Executed on cold power-up and whenever operator panel IBL/TEST switch is pressed.

INPUT/OUTPUT:
INTERRUPT STRUCTURE: Multilevel vectored priority interrupt; priority determined by interrupt location.

I/O SYSTEM SIZE:

	HP 2111F	HP 2117F
Standard I/O Channels	9	14
With one extender	25	30
With two extenders	41	46

MEMORY SYSTEMS
TYPE: 4K and 16K N-channel MOS semiconductor RAM.
WORD SIZE: 16 bits plus parity bit.
CONFIGURATION: Controller plus multiple plug-in memory modules. Available in 32K and 128K-byte modules.
PAGE SIZE: 2,048 bytes.
ADDRESS SPACE: 65,536 bytes without DMS; 2,097,152 bytes with DMS (2111F and 2117F).
DIRECT MEMORY ACCESS (DCPC ACCESSORY): Assignable to any two I/O channels.
MAXIMUM TRANSFER BLOCK SIZE: 32,768 words.

DCPC TRANSFER RATE (all rates in Mbytes/s):

		Minimum	Typical	Maximum
High-Performance Memory	**HP 2102E**			
	Input w/DMS	2.282	2.284	2.284
	w/o DMS	2.282	2.284	2.284
	Output w/DMS	2.036	2.114	2.196
	w/o DMS	2.196	2.284	2.284
High-Performance Fault Control Memory	**HP 2102H**			
	Input w/DMS	2.28	2.28	2.28
	w/o DMS	2.28	2.28	2.28
	Output w/DMS	1.902	1.968	2.036
	w/o DMS	2.036	2.114	2.196

PHYSICAL CHARACTERISTICS
WIDTH: 42.6 cm (16¾ in) behind rack mount; 48.3 cm (19 in) front panel width on sides.
DEPTH: 62.2 cm (24½ in); 58.4 cm (23 in) behind rack mounting ears.

	HP 2111F	HP 2117F
HEIGHT:	31.1 cm (12¼ in)	44.5 cm (17½ in)
WEIGHT:	30 kg (66 lb)	50 kg (110 lb)

ELECTRICAL CHARACTERISTICS
LINE VOLTAGE: 2111F: 88-132V; 176-264V with option 015.
2117F: Computer mainframe same as 2111F. Floating Point Processor voltage selector offers choice of 90-110V, 108-132V, 196-242V, and 216-264V ranges.
LINE FREQUENCY: 47.5 to 66 Hz.
POWER DISSIPATION: 2111F: 625 watts (maximum).
2117F: 825 watts (maximum).

ENVIRONMENTAL LIMITATIONS
OPERATING TEMPERATURE: 0° to 55°C (+32° to 131°F).
STORAGE TEMPERATURE: −40° to 75°C (−40° to 167°F).
RELATIVE HUMIDITY: 20% to 95% at 40°C (104°F), non-condensating.
VIBRATION AND SHOCK: Vibration: 0.30 mm (0.012 in) p-p, 10-55 Hz, 3 axis.
Shock: 30 g, 11 ms, ½ sine, 3 axis.
Contact factory for review of any application requiring operation under continuous vibration.

PRICES IN U.S.A.:
2111F, $12,250 (includes 64K bytes high-performance memory).
2117F, $16,000 (includes 128K bytes high-performance memory and dynamic mapping subsystem).
MANUFACTURING DIVISION: DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, California 94015 U.S.A.

New Memory Systems For HP 1000 Computers

by Alan H. Christensen and David C. Salomaki

TAKING ADVANTAGE of the latest innovations in memory component technology has been an HP strategy since the company became one of the first users of 4K-RAM semiconductor memory in computers in 1974. As denser, faster, and more reliable memory components have become available, HP has created new products to use them, thereby passing on to its customers the impressive performance and cost benefits of the latest technology (Fig. 1).

The latest advance in memory components is the 16K dynamic RAM (random-access memory). Based upon this recent development are two new 128K-byte memory array boards for HP 1000 Computers. One is a standard-performance memory array, and the other is a high-performance version using high-speed 16K RAMs.

With these new boards, users can now have one megabyte of main memory for the first time in a small computer like the HP 1000. Combined with new software tools, such as RTE-IV, that are designed to use large memories effectively, this increased main memory capacity enables HP 1000 Systems to solve many problems formerly addressable only by large mainframe computer systems. Two examples are simulation and computer-aided design.

Development of the New Memory Array Boards

The introduction of the 16K-RAM-based memory boards was the culmination of a memory program begun at HP's Data Systems Division almost three years ago. At that time a new high-speed processor, later known as the HP 1000 E-Series Computer, was under development.

It was planned that the new processor would use the same memory systems as were used in the M-Series. Early in the project, however, it was realized that the architecture of the new computer would allow substantial performance benefits with a high-speed memory system.

The first (and key) question to be answered was what 4K RAM type to use. There were three possibilities: 22-pin RAMs (4030s), 18-pin RAMs (4050s), and 16-pin RAMs (4027s, 4096s). Each had its advantages and weaknesses. The twenty-two-pin part was attractive because of the length of time it had been in production, but was hampered by poor board packing density. Eighteen-pin RAMs were desirable because of their use in current production, but suffered from high-voltage-level clock requirements. Sixteen-pin RAMs offered good board density and TTL compatibility, but required extra circuitry for

Year Introduced	Product	Technology	HP 1000 Computers Used in	Memory Access Time	System Cycle Time	Bytes per Array Board	System Power Consumption (64K Bytes)	Typical Cost per Byte (Cents)**		Typical Memory System Size (Bytes)
								System	Add-On	
1971	Core Memory System* (12885)	Core	2100	400 ns	980 ns	16K	215 Watts	51		64K
1974	13187A 2102A	4K RAM Std. Speed	M-Series	250 ns	595 ns	32K	28 Watts	15	14	64K
1977 (Spring)	12741A 2102E	4K RAM High Speed	E-Series	150 ns	350 ns	32K	36 Watts	16.4	6.4	128K
1977 (Fall)	12747A 2102B	16K RAM Std. Speed	M-, E-Series	250 ns	595 ns	128K	30 Watts	5.8	4.8	256K
1978 (Spring)	12747H 2102E	16K RAM High Speed	E-, F-Series	150 ns	350 ns	128K	32 Watts	4.8	3.8	256K
1978 (Fall)	12747A 2102B	16K RAM Std. Speed	M-, E-Series	250 ns	595 ns	128K	30 Watts	4.0	3.1	256K

*64K-Byte Core Memory System Consists of Four 16K-Byte Core Stack Boards, Four X-Y Driver Cards, Two Inhibit Driver Cards, One Inhibit Driver Load Card, and One Memory Controller.

**All Semiconductor Memory Systems Consist of One Controller and the Proper Number of Memory Array Cards for a Typical System (DMS Card Is Included for Memory System Sizes Greater than 64K Bytes).

Fig. 1. *Chronology of memory system development at HP shows the benefits gained by using the latest, best available memory components.*

address multiplexing. After careful consideration of the alternatives it was decided to use the 16-pin 4027 RAM as the basis for the new high-speed memory system.

With the RAM decision made, project effort turned to developing the circuit design for the new memory controller and memory array board. The key question was where in the system to perform the address multiplexing required by the 16-pin RAMs. One possibility was to multiplex the addresses on the memory controller. This had the advantage of yielding the least expensive memory system because only one set of multiplexing circuitry was required. The other alternative was to multiplex the address on each memory array module. This increased the cost of large memory systems but offered very significant improvements in memory performance. By doing address multiplexing on the same board as the memory components, the timing skews introduced by the buffers and interconnecting cable between the controller and memory array boards could be eliminated. This permitted using the RAMs at the limit of their specifications, thus allowing main memory access times that rivaled cache memory speeds in other computers. This overriding performance advantage led to the decision to perform the address multiplexing on the individual array boards (Fig. 2).

This multiplexing solution had an important secondary benefit. It permitted a controller/memory board interface compatible with that of the existing M-Series. This created an interesting engineering situation: if a memory interface could be designed that would work with both the M-Series and the E-Series Computers, it would be possible to develop a universal memory array board that could be loaded with high-speed 4K RAMs to create high-speed memory modules or loaded with standard-speed 4K RAMs to create standard performance memory modules. Thus, previously developed standard-speed controllers, as well as future controllers (high-speed or low-speed), could use the same array board. Also, by going to a single printed circuit board based on 16-pin RAMs, earlier boards based on 22-pin and 18-pin RAMs could be phased out of production, leading to significant efficiencies in component ordering, scheduling, RAM device level testing, production, training, and repair.

Before many months had elapsed, the idea of a universal array board was taken one step further. At that time, 16K RAMs were just beginning to be proposed by the memory vendors. Before long it became clear that the industry standard on this future part would be a 16-pin design with timing and interface specifications similar to the 4027 4K RAM. Therefore the universal RAM array board was modified to be able to use 16K RAMs as well as 4K RAMs. Minimal additional circuitry was needed to provide this compatibility.

From this critical decision to the present, the development of memory systems for HP 1000 Computers has been reasonably straighforward. The first product to use the universal 16-pin-RAM array board was the high-speed memory subsystem for the E-Series Computer. This consisted of a new high-speed controller board and a high-speed 16K-word memory module (the universal array board loaded with high-speed 4K RAMs). Shortly thereafter, a standard-speed memory board loaded with slower 4K RAMs was made available for both E-Series and M-Series Computers.

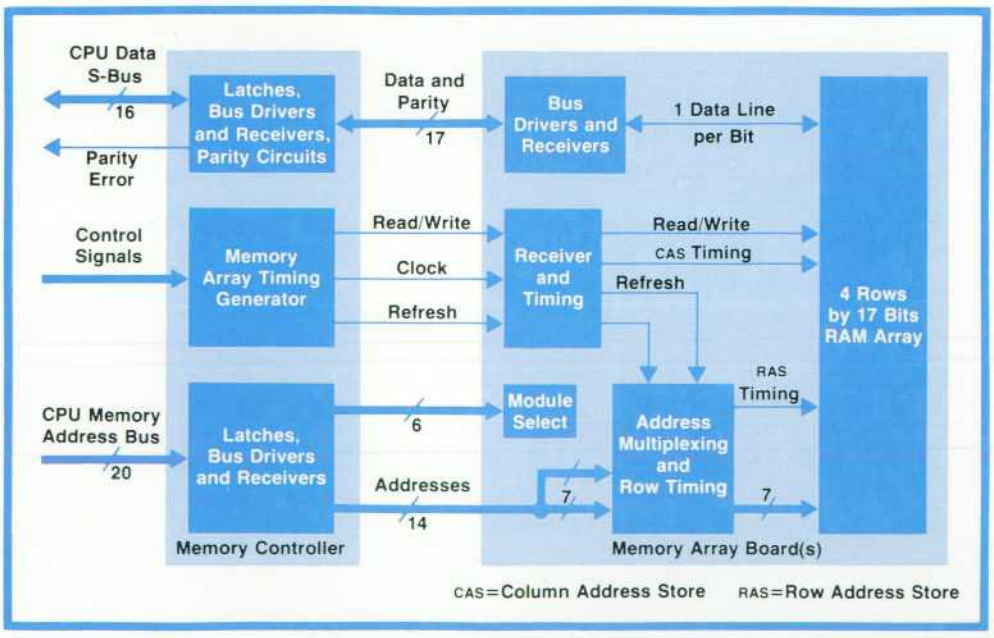


Fig. 2. *HP 1000 parity memory system architecture. Doing address multiplexing on each memory array board minimizes logic skews and allows higher performance. Putting this function in the controller would have simplified the memory array boards, but a slower memory system cycle time would have been necessary to allow for timing uncertainties introduced by the memory system cabling.*

As soon as 16K RAMs became available in sample quantities, they were tested in the universal array board. When production quantities became available, the 64K-word array board was released to production. This fast progression from sample parts to finished product occurred for both standard-speed and high-speed 16K RAMs.

Fault Control Memory

The 16-pin RAM array modules and the high-speed controller are not the only new memory products recently added to the HP 1000 product line. Also introduced have been standard-performance and high-performance fault control memory systems. By correcting single-bit errors in memory, these new products offer significant improvements in memory system reliability, thereby opening the door to many new reliability-oriented applications.

HP's experience with MOS RAMs has shown that once infant RAM mortalities have been weeded out (normally after the first 500 to 1000 hours of operation), RAM devices have a fairly stable life period characterized by random hard failures at rates of 0.01% to 0.1% per 1000 hours of operation.

RAM failures are typically classified as hard or soft. Hard failures are those that occur every time the malfunctioning bit or chip is accessed. Soft failures, on the other hand, are intermittent, or non-repeatable. These failures manifest themselves mainly by the loss of one bit of data in a RAM, and are usually attributable to RAM sensitivities to external conditions such as noise spikes on power supplies or clocks, temperature extremes, and/or timing variations, including refresh. Soft failures may also be caused by address/data sensitivities within the RAM.

Experience has shown that soft failures are the predominant ones once this infant mortality period has passed and that they usually affect only a single bit within a RAM and a single RAM within a memory word. Therefore, in most cases single-bit error correction is sufficient to allow continued system operation.

Error correction works by coding the data word with additional check (code) bits. Simple parity, used in regular memory systems, is an example of a single-bit error *detecting* code; information is encoded in just one bit more than the data bits. The number of check bits required for single-bit error correction can be determined from the equation $2^k \geq m+k+1$ where k is the number of check bits required and m is the number of data bits. Thus in a 16-bit computer, five additional bits will provide single-bit error correction.

In the HP 1000 fault-control memory, six check bits are used. The addition of one more check bit provides double-bit error detection. This capability insures that a double-bit error will not be mistakenly interpreted as a single-bit error and corrected to a wrong value.

Many possible codes can be generated for 16 data bits. A code is defined by parity equations for each check bit. Valid codes have two characteristics: no two check bits can be derived from exactly the same set of data bits, and no two data bits can contribute in the same way to all check bits. The particular code used in the HP 1000 fault control memory option was chosen to minimize the number of parity generator inputs so as to provide the fastest possible code-generating and checking times and thus reduce the memory cycle overhead incurred for error correction. The fault control (error correcting) circuitry for the

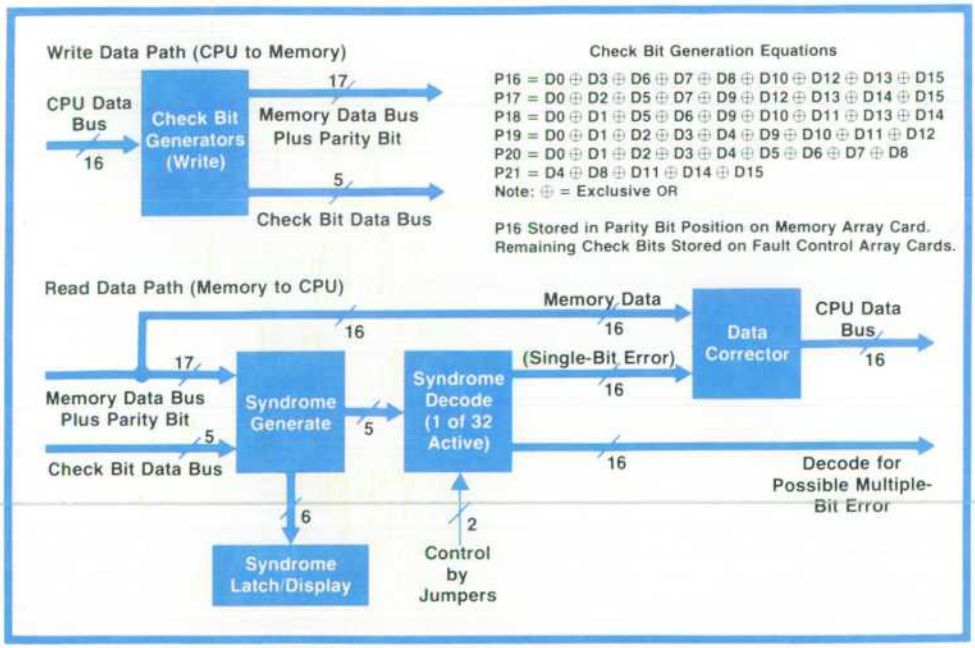


Fig. 3. *Fault control memory generates and stores six check bits (P16-P21) on each write to memory. On each read from memory, the six check bits are regenerated from the data bits (D0-D15) and compared with the stored check bits to form a two-octal-digit syndrome that indicates the type of error, if any. Single-digit errors are automatically corrected by inverting the erroneous bit.*

Achieving Reliability in Semiconductor Memory Systems

The memory systems based on 16K RAMs are the third generation of semiconductor memory systems for HP 1000 Computers, and are expected to be the most reliable to date. As a result of the lessons learned from previous memory products, we have come to understand that good memory reliability can only be achieved through a combination of three things: a good electrical design, use of very reliable RAMs, and a carefully planned and executed manufacturing process.

Design

Four important design steps were taken to assure memory system reliability. The first of these was the use of worst-case design rules for the circuit design. The proper operation of every circuit path was computed and verified using both minimum and maximum specifications for all devices. All devices were properly derated for temperature, loading, power consumption, etc.

Second, a thorough operating condition checkout of the memory boards was performed. Ringing on signal lines, power supply ripple and noise, and rise and fall times were measured to insure that the printed circuit board version of the theoretical circuits behaved as expected.

Third, a thermal study was performed. While the memory system integrated circuits were in actual operation in a computer, the case temperatures of all ICs on the board were measured and used to compute device junction temperatures. These junction temperatures were then compared with empirically derived values to extrapolate device MTBFs (mean time between failures). The individual device MTBFs were then used to calculate the expected MTBF of the board to make sure that the design would meet the reliability goals.

The last step in the design process was an exhaustive round of environmental testing. This testing involved running diagnostics and operating systems software in heavily loaded computer systems under various conditions of operating temperature extremes, humidity, vibration, power variations, and static discharge. The first round of these tests was performed on the first printed circuit versions of the boards. The results of these tests and of temperature profile tests led to layout changes designed to increase board reliability. The environmental tests were repeated on the first production run of boards to guarantee their reliability under normal production variances.

RAM Reliability

In even the most complex memory system, each RAM contains more transistors than all the non-RAM components combined. Given that there can be anywhere from 34 to 1400 RAMs in a memory system, the dominant importance of RAM reliability becomes apparent.

For this reason, one of the principal efforts in the development of the 16-pin RAM family products was an extensive reliability evaluation of the 16-pin RAMs on the market. This evaluation proceeded in two steps. The first was a series of characterization tests performed on all available 16-pin RAMs to determine device margins for our applications. On the basis of these initial tests, a number of vendors were selected to undergo a qualification process. The parts chosen for this second step underwent an elaborate series of tests that included package testing, 125°C static and dynamic burn-ins, life testing in operating computers (3 million device hours equivalent), and system compatibility testing.

Manufacturing Process

Developing a reliability-oriented manufacturing process is the third and final step in achieving good memory system reliability. Experience has shown that it is not enough to develop a reliable design and select RAM vendors with reliable parts. MOS LSI manufacturing is subject to day-to-day fluctuations that may go undetected by the memory vendor but dramatically affect reliability. Consequently, the user of dynamic MOS memories must gear the manufacturing process to detect changes in incoming RAMs and eliminate parts that are bad or marginal.

The HP 1000 Computer manufacturing line is organized around this idea. All incoming RAMs are dynamically burned in at 125°C for 72 hours and then tested to the limits of their specifications. Lot failure rates are carefully monitored, and lots that have unusually high failure rates are returned to the vendor. Fully tested RAMs from good lots are soldered into boards and the boards are tested for approximately 100 hours. Board-level tests include vibration, temperature extremes, and system compatibility.

The manufacturing flow discussed here has evolved over a number of years, and the process continues to evolve as new failure modes emerge and old failure modes cease to be important. Two examples of this evolution are cold testing at the device level and vibration testing. At one time all devices were individually tested at 0°C as well as at 70°C. As memory vendors have learned to test accurately for cold sensitivities, the need for 100% cold testing has disappeared. Now, sample cold testing along with board-level temperature extreme testing is adequate to screen for this failure mode. Vibration testing, on the other hand, is a recent addition to the manufacturing flow. Intermittent failures at the end of the production line and in the field spotlighted the need for a screen of this type. Since this was begun, intermittent failures have dropped to a very low level.

The evolution of the manufacturing process and the RAM diagnostics are critical to ensuring continuing memory reliability. Typical failure rates of incoming RAMs range from 0.5% to 3% (lots with higher failure rates are rejected). The manufacturing process weeds out bad RAMs to the extent that warranty failure rates are expected to be no greater than 0.03% per 1000 hours of operation.

standard and high-performance memory systems is identical. The only differences are in memory timing.

Operation of the fault control system is as follows (see Fig. 3). On a write to memory, the check bits are formed and written to a separate fault control array board that operates in parallel with the memory array boards. When a memory location is read, new check bits are generated from the 16 data bits and compared with the check bits from memory to form a syndrome for the data word. This syndrome is a six-bit code word that can be decoded to indicate whether there are no errors, an error in one of the 22 bits, or a multiple-bit error. If there are no errors, the 16-bit data word is passed to the CPU unchanged. When a single-bit error is detected, the respective bit is inverted (to correct it) before the data word goes to the

Memory System Size (bytes)	RAM Failure Rate λ (% per 1000 hr)	Parity System MTBF (hr)	Error Correction System Effective MTBF (hr)	Reliability Improvement Factor
128K	0.08	19,293	27,283	1.4
	0.12	14,772	27,186	1.8
512K	0.08	5696	20,011	3.5
	0.12	4184	19,727	4.7
1280K	0.08	2364	10,699	4.5
	0.12	1719	10,486	6.1

Fig. 4. *Mean times between failures for parity and error correcting memories. These rates are predicted for a system in the 1500-to-5500-hour operating range. System MTBFs are approximately 60% lower during the first 500 hours of operation.*

CPU. In the case of multiple-bit errors, the correction feature is disabled, uncorrected data passes to the CPU, and a parity error signal is sent to the CPU. Action from that point is a function of software control.

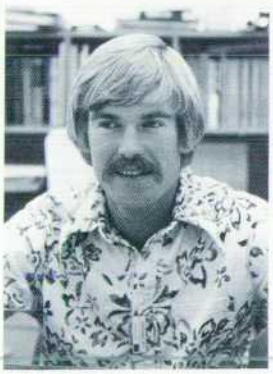
A number of features are included on the memory controller board to assist in fault location. Six LEDs display the syndrome and are updated whenever the parity error signal is sent to the CPU. The error correction ability can be disabled for diagnostic purposes to activate the parity error signal for single-bit errors; in this case, the syndrome LEDs can be decoded to find the erroneous bit. Another LED indicates whether a single-bit error has occurred since the CPU was last reset. Finally, as with the universal memory array board, jumpering and loading options have been included on the memory controller and fault control array boards to allow a single printed circuit board to be used for both standard-speed and high-speed applications.

The benefits of fault control are dependent on the RAM failure rate (both hard and soft failures), the memory size, and the interval between preventive maintenance periods (when all hard-failure RAMs should be removed). Although the reliability improvement factor of fault control over parity can be from 20 to 100 if only the RAMs are considered, the improvement gained by using error correction in a computer is more realistically a factor of about 1.5 to 20. This is because peripheral IC reliability becomes an important factor when RAM failure rates approach their burned-in levels.

In general, the contributions of fault control are more noticeable for large memory systems and for RAMs with high failure rates. Fig. 4 gives an example of these considerations (calculated for the HP 1000 fault control memory system using 16K RAMs with a 1000-hour preventive maintenance interval).

Acknowledgments

Many people were crucial in the development of the current offering of HP 1000 memory products. Bob Frankenberg provided the initial (and sustaining) impetus to use semiconductor memory and to remain at the leading edge of memory system technology. Jim McClure and Jan Hofland were responsible for most of the design and planning of the memory family; their ideas and innovations paved the way for easily integrating RAM improvements into our memory systems. Gordon Goodrich was important for his RAM evaluations and design expertise. Also contributing were Brian Fisher, Carl Ubis, Dave Langley, Scott Stallard, and Cle Riggins.



Alan H. Christensen

An HP employee since 1972, Alan Christensen was project manager for the HP 1000 high performance/fault control and high performance/high density memory products. Alan received his BSEE degree and MSEE computer engineering degrees from Stanford University in 1972. Born in Dayton, Ohio, and raised in El Segundo, California, Alan is single and lives in Sunnyvale, California. He enjoys all types of athletics, but is particularly fond of volleyball. An all-American volleyball player while in college, Alan is now a member of a local AA men's volleyball team. These days, he spends much of the remainder of his time installing a fiberglass Jacuzzi whirlpool in his backyard.

David C. Salomaki

Dave Salomaki is a 1975 BSEE graduate of Worcester Polytechnic Institute and a 1977 MSEE graduate of Stanford University. An HP employee since 1977, Dave was a designer of the HP 1000 E-Series and F-Series memory systems. He also worked as a design engineer in microprocessor applications at Stanford Research Institute. A resident of Palo Alto, California, Dave is fond of all sports—especially football, racquetball, tennis and swimming—and plays softball on the HP/Cupertino city league team. He enjoys reading fiction, "monkeying around" at home with microprocessors, and going out to eat. He is also involved in a local church young adult group.

Multipoint Terminals for HP 1000 Systems

by Denton B. Anderson, Mitchell B. Bain, and Gary Johnson

MULTIPOINT IS A TECHNIQUE that allows many computer terminals to share one communications line. This means only one computer interface and one pair of modems are needed, thus lowering the communications cost to the user.

No longer a capability only of large mainframe systems, multipoint is now available for HP 1000 Computers. The HP 1000 multipoint protocol is based on IBM's binary synchronous communications procedure (Bisync). This protocol resolves line contention by addressed poll and select sequences, and allows for extensive error detection and correction (by retransmission).

A new microprocessor-based interface card and accompanying software were developed to implement this protocol and reduce the CPU overhead involved. The multipoint software consists of a driver and some utilities. The firmware for the microprocessor and the driver software were designed together, resulting in a simple and logical interface between the computer and the multipoint card. Tasks are partitioned so that the driver software never concerns itself with communications protocol, modem control, error control, time-outs, or message content. The RTE-IV operating system is interrupted only at the conclusion of a transaction. Packed data strings, as many as 1000 characters at a time, are rapidly transferred between the interface buffer and the HP 1000 main memory via DMA (direct memory access). Data integrity is assured by means of a 16-bit cyclic redundancy check (CRC-16) after each message.

Terminal operators can be developing programs using any of the RTE facilities, or may be responding to customer-developed application programs. The terminals can be HP 2645A or 2648A CRT Terminals in any combination. All of the HP 2645A/2648A features such as user-definable soft keys, cartridge tape drives, printers, and graphics (2648A) are available.

Assured Terminal Access

The multipoint driver's terminal servicing algorithm assures each terminal access to the line by examining the status of each terminal's equipment table entry sequentially and checking for active write, read, or control requests from a system or user program. Write or control requests are serviced immediately. Read requests are honored after the status of all the other terminals has been queried once. Terminals without active requests pending may be routinely polled so the operator can get RTE system attention or a user-written program can be activated.

Terminals are assigned conventional RTE equipment table (EQT) and logical unit (LU) numbers so multipoint terminals are treated the same as any other RTE peripherals. Thus user programs may use standard FORTRAN read and write statements to communicate with multipoint terminals. Specifically,

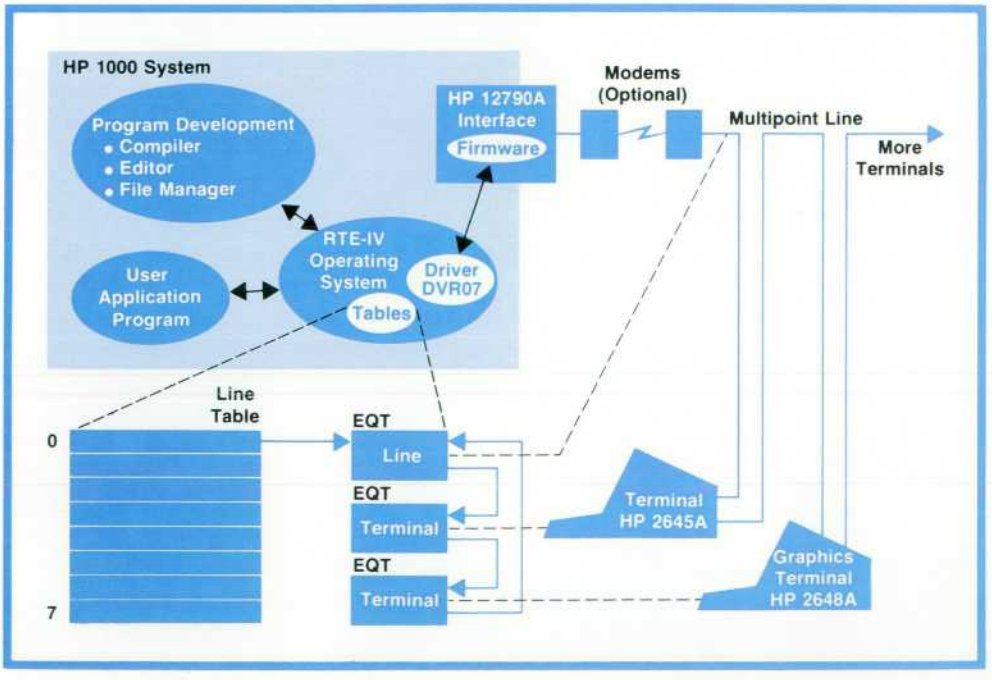


Fig. 1. *Multipoint allows many terminals to share one communications line. In HP 1000 multipoint, terminals are assigned conventional RTE equipment table (EQT) and logical unit (LU) numbers, and are treated like other peripherals. The communications line also has EQT and LU status. All of the EQT numbers of a particular line are stored in a linked list that describes that line.*

programs already written for point-to-point terminals will function *unchanged* with multipoint. Standard RTE EXEC calls are used, and programs may be written to take advantage of large block transfers.

All of the EQT numbers of a particular communications line are stored in a linked list that describes that line (Fig. 1). The head of the linked list is the line EQT number and is pointed to by an entry in an eight-word line table. Since the line and interface have EQT and LU status it becomes possible for the user to broadcast messages to all terminals on a line and to collect status information using a "Who-Are-You" request.

There are four basic requests that the driver makes to the interface:

- Poll/Select (for Read or Write)
- Unload Text (activate DMA after a Poll)
- Load Text (activate DMA after a Select)
- Transmit (terminate Load then send text).

The Poll/Select requests specify which terminals are to be queried, and the interface firmware builds the communication messages in conformance with the protocol. The firmware adds the protocol control characters to the text blocks and computes and verifies the error control characters (CRC-16) at the end of each block (Fig. 2).

Low Overhead

With only routine polling taking place, the RTE-IV overhead at 9600 bits per second is a constant 6% and 10% for synchronous and asynchronous lines respectively. These figures are essentially independent of the number of terminals on a line, but increase proportionately with the number of lines. This overhead actually decreases while text information is being exchanged.

In addition to the four basic driver requests a number of control requests can be made. These override certain default conditions (number of retries when an error is encountered, message blocking factors, etc.) and supplement the basic modem control exercised by the firmware. These control requests include the following:

- Time Delay
- Read Status
- Change Retry and Blocking Factors
- Read Modem Status
- Set Modem Controls
- Change Watchdog Time
- Reset (Self Test).

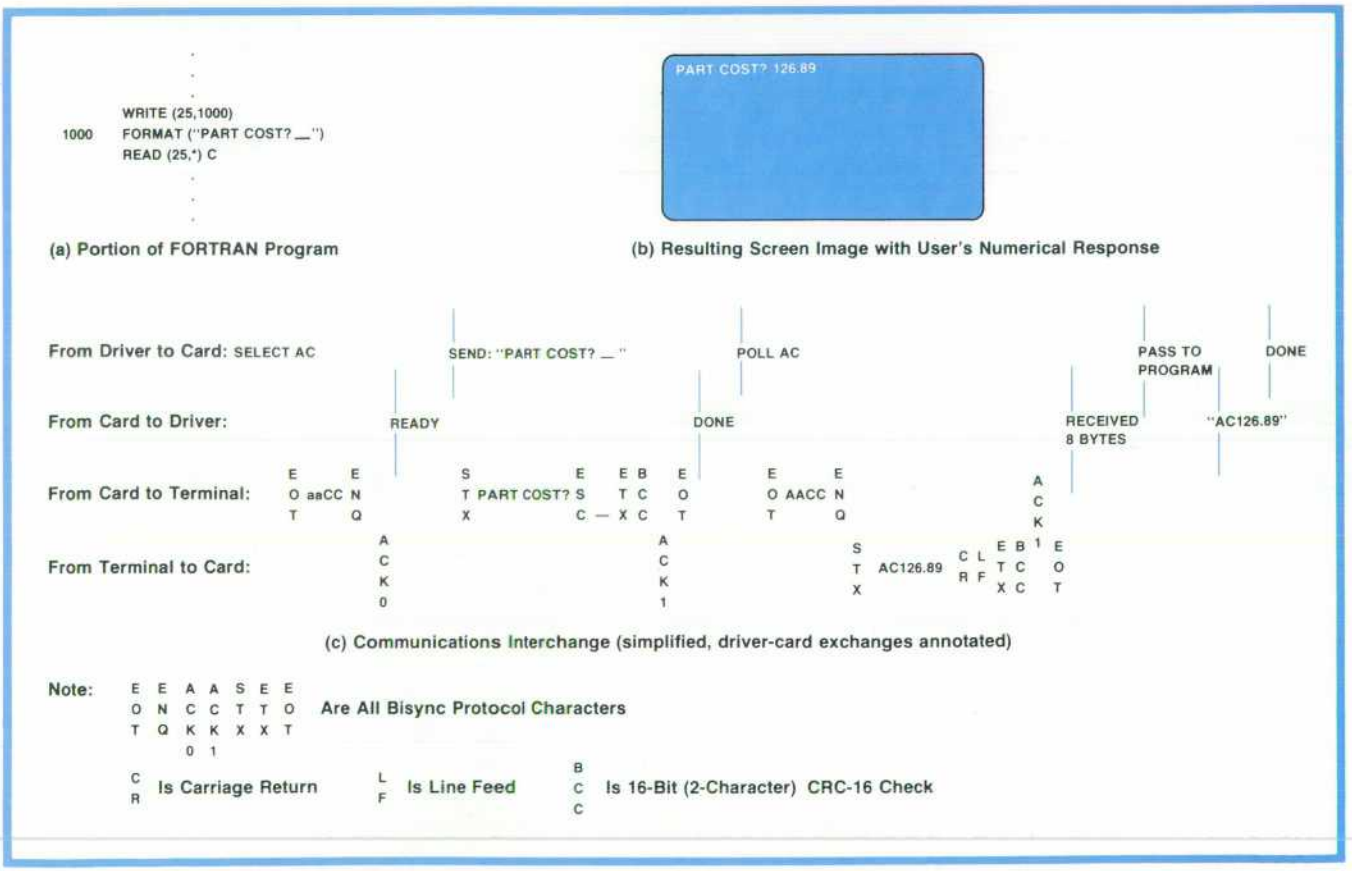


Fig. 2. *HP 1000 multipoint protocol example, showing typical communications interchanges between the multipoint software (driver), the multipoint hardware (card), and a terminal. The multipoint interface card, which is microprocessor-based, handles the protocol requirements, relieving the computer of this burden.*

The multipoint interface card is equipped with more modem control/status leads than are needed by the standard Bell System modems. These extras are passed to the user's program (through the software driver) so that non-standard modems can be manipulated by user software.

An eight-bit configuration switch on the interface card is set to match the multipoint network and terminal requirements of a given installation. The switch is examined by the interface firmware at configuration time (power-up) and such details as synchronous/asynchronous timing mode, communication bit rate, and modem control states are initialized.

After initialization the data communications function is handled by the firmware in three general areas:

- The USART. Universal synchronous/asynchronous receiver/transmitter serializes and deserializes the data bits at the RS-232 interface.
- The Protocol. Examine each character. If a control character, advance the protocol state. If a display character, store in the buffer.
- Error Control. Check the protocol and compute the CRC-16 check. If an error, advance to the error control state.

The HP 2645A/2648A protocol is based on IBM's binary synchronous communication procedure (Bisync), which can best be described as having many states with relatively simple state transition rules. The many states require that a microprocessor-based controller have a relatively large control ROM, while the simple transition rules assure that the microprocessor will be idling most of the time. At 9600 bits per second, characters arrive every millisecond.

The multipoint interface card exploits this characteristic by doing the USART function in firmware, not in LSI hardware. The USART routines are entered by the interrupts from the modem clock or the on-board baud rate generator (see block diagram, Fig 3). In effect, the microprocessor is asked to "earn its keep" and not idle so much.

Communicating in a half-duplex mode (sending and receiving but not simultaneously), the interface is capable of sustaining a rate of 19,200 bits per second, the upper limit of RS-232. It can do this either synchronously or asychronously. In the latter case the interrupts occur at eight times the bit rate.

Fast Microcontroller

For this scheme to work, a fast microprocessor with good interrupt facilities is necessary. A fast microcontroller developed by Hewlett-Packard's Loveland Instrument Division and used in other HP products fills the requirements for data communications admirably.

At 9600 bits per second (the fastest terminal rate) the microprocessor is capable of executing 1536 instructions per eight-bit synchronous character and 1920 instructions per ten-bit asynchronous character (start, eight-bit, and stop). The overall timing load is illustrated in Fig. 4. Advancing the protocol state requires some ninety instructions. The USART routines can take as much as 20% of the available time and the CRC calculation another 15%. The idle time is spent in a "watchdog" state so the host computer can expect a response from the interface no matter what data communication or procedural failure might occur.

Although the foregoing timing analysis is an important consideration in assessing the performance of the microprocessor-based interface, the real value of the front-end processor goes far beyond its speed. In addition to the three basic communications functions (USART, protocol, and error control), the microprocessor relieves the computer of the following tasks:

- Power-up/self-test
- Initialize or override error retries
- Buffer management between the computer and the

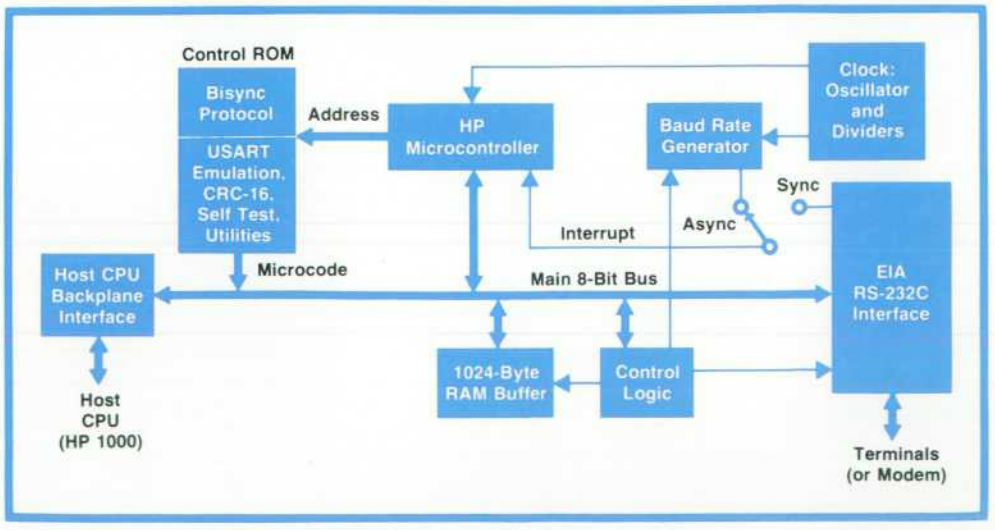


Fig. 3. *Multipoint interface card performs three main functions: USART (universal synchronous/asynchronous receiver/transmitter), multipoint protocol, and error control.*

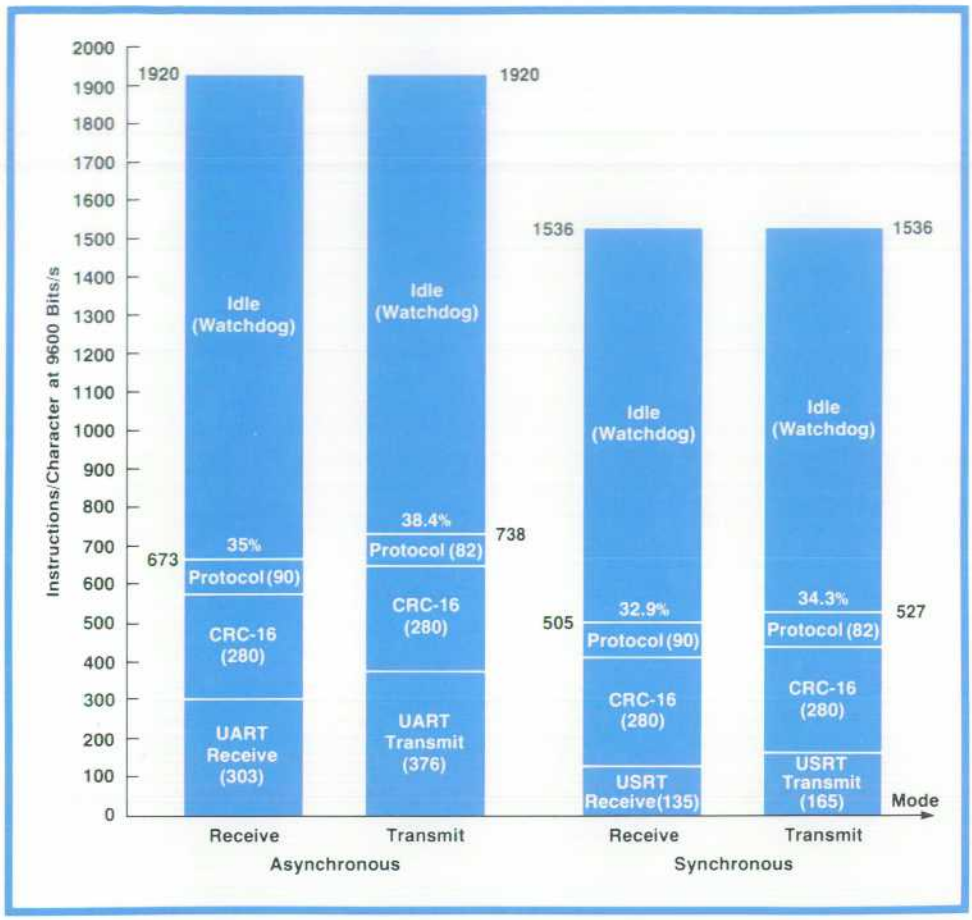


Fig. 4. *Multipoint card microprocessor timing load at 9600 bits per second, the fastest terminal rate. The microprocessor can execute 1920 instructions in the time it takes to receive or transmit each ten-bit asynchronous character at this rate. Eight-bit synchronous characters take less time, enough for 1536 instructions. In either case the microprocessor idles much of the time.*

terminals
- Some editing (e.g., strip the carriage return and line feed)
- Handshake modems, turn lines around
- Time delays and watchdog time-outs
- Simplify host computer interface, return error state information

Denton B. Anderson

Denton Anderson, an HP employee since 1968, designed the firmware for the HP 1000 multipoint terminal system. His earlier projects have included the 91200B television interface and the system pacer for the 2313 data acquisition system. Born in Richmond, California, Denton received his BSEL degree from California State Polytechnic Institute at San Luis Obispo in 1965 and his MSEE degree from Stanford University in 1967. Prior to college, he spent three-and-a-half years in the U.S. Navy as an electronics technician and basic electronics instructor. Denton's outside interests include fly fishing, tennis and attending sports car races. He also holds a brown belt in Kung Fu and is an avid Stanford football fan who "never misses a home game." Denton is married, has a four-year-old daughter and lives in San Jose, California.

Mitchell B. Bain

Mitch Bain designed the hardware for the HP 1000 multipoint terminal system interface. An HP employee since 1966, Mitch has also designed several of the communications interfaces for the 2100 series computer. Before coming to HP, he worked as a design engineer at F.L. Moseley Co. (now HP's San Diego Division). Born in Los Angeles, California, Mitch received his BS degree in physics from California Institute of Technology and has taken several graduate courses in computer science at University of California at Los Angeles and University of California at Berkeley. The author of several published papers on instrumentation and computers, Mitch is an alumni of the U.S. Navy Treasure Island Radio Materiel School. Mitch spends much of his spare time cruising and racing his sailboat and is a member of the U.S. Coast Guard Auxiliary. He also enjoys taking color photographs and developing them in his own darkroom. Mitch lives in Mountain View, California, is married and has four daughters, ages 20 through 27.

Gary W. Johnson

Gary Johnson designed and implemented the software for the HP 1000 multipoint terminal system. With HP since 1973, Gary was also developer of the 7905 disc drive version of the DOS operating system. He attended the University of Michigan in electrical engineering for one year before becoming a full-time staff member at the university. During his seven years as a U of M staff member he worked as an engineer in the psychology department and then as lab supervisor for the Simulation Research Center (a division of the aerospace engineering department). Gary was also an engineer for Motown Records and specialized in aircraft inertial and radar navigation systems during his one-year U.S. Air Force enlistment. Born in Pontiac, Michigan, Gary is single, lives in San Jose, California, and enjoys yard work and working with his home videotape system, but he says his real hobby is his job.

ABRIDGED SPECIFICATIONS

HP 1000 Multipoint

OPERATING SYSTEM: 92064A RTE-M system (RTE-MIII configuration) for application program execution and 92067A RTE-IV system for both application program execution and program preparation.

HARDWARE: 2645A and 2648A CRT terminals with option 030 and 13260C/D terminal interface accessory.

NUMBER OF TERMINALS PER MULTIPOINT LINE: Nominally, up to 32 terminals can be connected to the 12790A interface via a single multipoint line.

NUMBER OF LINES PER SYSTEM: A maximum of eight multipoint lines (eight 12790A interfaces) can be supported per system.

NUMBER OF TERMINALS PER SYSTEM: Limited by the system EQT number allocation (63 maximum including all other system peripherals).

SYSTEM USE: The approximate requirement for otherwise user-available processing time at 9600 bps in an HP 1000 E-Series Computer with standard performance memory operating under RTE-IV is:
SYNCHRONOUS: 6%
ASYNCHRONOUS: 10%

INTERFACE TO NEAREST TERMINAL OR MODEM: 15.2 metres (50 ft), maximum.

BETWEEN ANY TWO TERMINALS: 609 metres (2000 ft), maximum.

TOTAL LINE LENGTH: 4876 metres (16000 ft.), not including distance between modems.

POWER: Taken as required from HP 1000 Computer.

PRICE IN U.S.A.: 12790A Interface, $1500; 91730A Driver, $250.

MANUFACTURING DIVISION: DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, California 95014 U.S.A.

- Build canned messages (Poll/Select)
- Get terminal configuration information (Who-Are-You request).

Thus the HP 1000 multipoint subsystem is a good demonstration of the benefits of microprocessor-based interface cards. This approach allows more complex protocols to be used without further taxing the RTE system. This preserves more computational power for the user.

The HP 12790A hardware has two front-edge connectors, one with an RS-232C interface to connect to the multipoint network, and the other with a microprocessor bus interface. The second connector is invaluable in testing the interface. The bus connector allows an easy connection to logic analyzers for troubleshooting, and is connected to diagnostic ROMs during production checkouts.

Acknowledgments

The authors want to acknowledge the invaluable assistance of Bob Shatzer, project manager, and Bill Stevens, product manager. Major contributors to the success of the product were made by Dave Handbury and Dave Hancock, who brought it into production, and by Carol Gilstrom and Bill Hamlin, who saw to the extensive training and documentation package. Ray Spear and Bill Thormahlen performed thorough hardware and system QA cycles. Jean-Pierre Baudouin has provided testing for numerous European modems.

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

HEWLETT-PACKARD JOURNAL

OCTOBER 1978 Volume 29 • Number 14

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 1501 Page Mill Road
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
Amstelveen 1134 The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku
Tokyo 168 Japan

Editorial Director • Howard L. Roberts
Managing Editor • Richard P. Dolan
Art Director, Photographer • Arvid A. Danielson
Illustrator • Susan E. Wright
Administrative Services, Typography • Anne S. LoPresti
European Production Manager • Dick Leeksma