HEWLETT-PACKARD JOURNAL

SEPTEMBER 1985

HEWLETT-PACKARD JOURNAL

September 1985 Volume 36 • Number 9

Articles

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

In this Issue

Our cover subject this month is the HP 3000 Series 37 Computer. This newest and smallest member of Hewlett-Packard's business data processing computer family is an affordable, user-installable system that supports up to 28 terminals and is suitable for small companies (like the horse breeder suggested by our cover photo) or for departments or workgroups of larger companies. At about half the price of the previous entry-level HP 3000, the Series 37 makes this computer family accessible to many more users. A major advantage is that all HP 3000 Computers run the same software. HP offers standard upgrades to larger systems from any family member, so a user starting with the Series 37 has an easy growth path all the way to a system that supports as many as 400 terminals and can handle the data processing needs of a fairly large company, and no reprogramming or software conversion will be required at any step. Any HP 3000 can also be part of a network that includes other HP 3000s, mainframe computers, personal computers, and engineering workstations. For example, HP's own worldwide electronic mail system runs on a network of HP 3000 Computers.

The Series 37's designers report on its design on pages 4 to 22. Among the engineering challenges was the integration of the central processing unit (CPU) on a single semicustom gate array chip (page 7). Simulation of the CPU chip and another gate array (page 13) refined the two chip designs to the point where the first chips produced worked as designed, a major accomplishment. To keep the cost low and ensure reliability and family compatibility, the project was carefully managed (page 4), and the hardware and software debugging tools received special attention (page 17).

In the articles on pages 25 to 36, you'll find the design story of the new HP 4760 PageWriter Cardiograph family. While a cardiograph is very different from a business computer, the major engineering contribution in the HP 4760 family is much like the Series 37's—very large-scale integration puts more computing power into a smaller package. Two of the new cardiographs have parts of the HP ECG analysis program, formerly available only in a separate computer system, built right in, along with a dedicated 68000 microprocessor. The HP 4760AM, which has the ECG measurements portion of the program, can make more than 4000 measurements on the ECG waveform and print the complete results or a summary. The HP 4760AI has the full ECG analysis program and provides an interpretation of the ECG waveform. Adult analysis is standard; pediatric analysis, based on age-dependent criteria, is an option. Some feel that such automated interpretation can be helpful in eliminating normals in high-volume screening, or in emergencies when no cardiologist is available. The ECG measurements capability helps the cardiologist reduce interpretation time and is useful in research and teaching.

R. P. Dolan

What's Ahead

Next month's issue will be devoted to the design of the HP Integral Personal Computer. The HP-UX operating system of this 25-pound transportable computer is HP's version of AT&T Bell Laboratories' UNIX™ operating system.

VLSI Delivers Low-Cost, Compact HP 3000 Computer System

This entry-level, user-installable computer system runs the same software as the largest HP 3000, but fits under a table and is much quieter than a typewriter.

by James H. Holl and Frank E. La Fetra, Jr.

THE HP 3000 COMPUTER product line is HP's current-generation business computer family. At the top of the line is the HP 3000 Series 68, which is capable of supporting hundreds of terminals and handling the data processing needs of a fairly large company. The newest and smallest HP 3000 is the Series 37, Fig. 1, a compact, quiet office computer capable of supporting up to 28 users. Like all HP 3000s, the Series 37 runs the same software as the Series 68. Although slower than the Series 68, of course, the Series 37 has about the same processing power as the Series III, the top-of-the-line HP 3000 when it was introduced seven years ago. VLSI (very large-scale integration) is the key to the exceptional price/performance of the new computer.

The HP 3000 Series 37 was conceived as the answer to the need to add a low-cost computer to the HP 3000 product line while maintaining reasonable performance. Although the need was obvious (many people are willing to take credit for discovering it), the solution remained elusive until a key project manager proposed a product that evolved into the Series 37.

Design Objectives

The original design objectives stated at the time the project was proposed were:

- Low system list price
- Four to eight terminal ports
- Mean time between failures (MTBF) greater than 2 years, including peripherals required to run the operating system
- Mid-1983 manufacturing release
- Series III performance
- Easier to use—you turn it on and it works
- Networking capability.

Most of the original objectives were met. The system list price is in the originally targeted range. Seven terminal ports are standard; 28 is the maximum number. The esti-

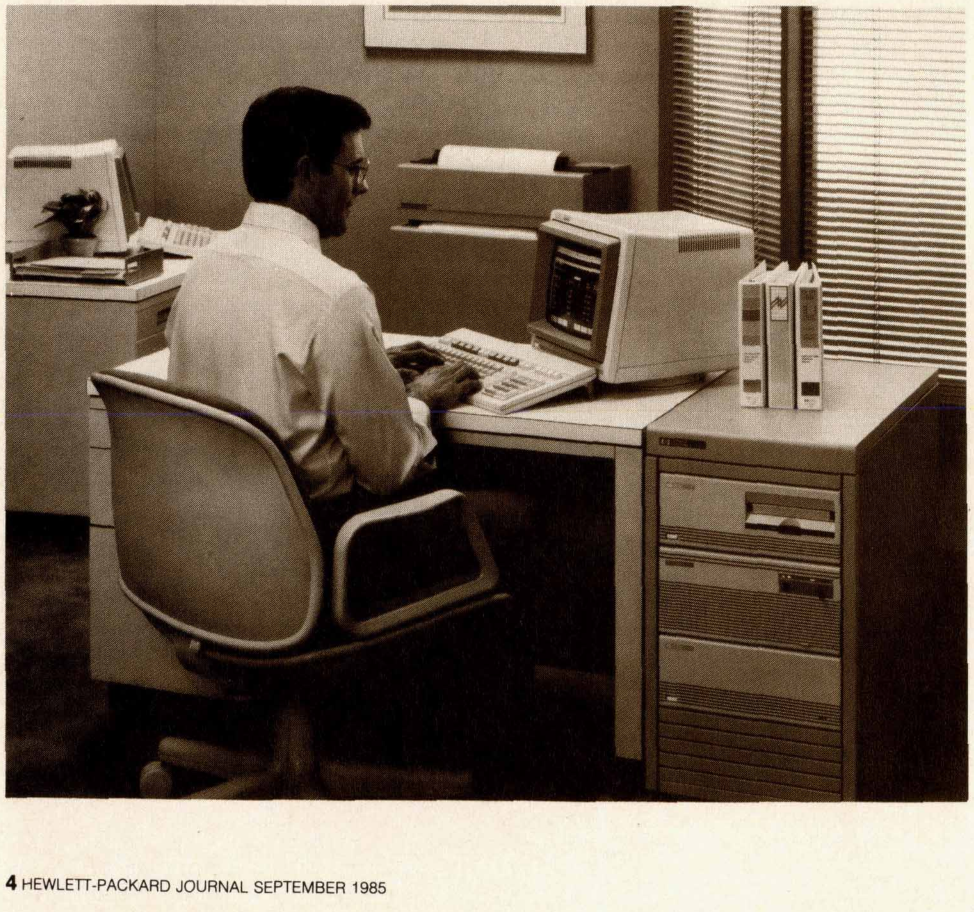


Fig. 1. *The HP 3000 Series 37 Computer runs the same software as other HP 3000s, supports up to 28 terminals, and fits easily into an office environment. The Series 37 system processing unit is the second unit from the top in the cabinet at right.*

mated MTBF of the system processing unit is 1.17 years, which did not meet the goal. However, the two-year goal was very aggressive and involved many separate parts of our corporation. The time to market goal was based on the time it would take to complete similar projects, on the average. This is called our 50% schedule because it is expected that the schedule can be met half of the time. The time it would take to complete nine out of ten similar projects was also projected, and this 90% schedule estimate turned out to be exactly the time taken. The actual release date was February 1985.

The Series 37 has Series III performance with up to 20 terminals. It can be installed by the customer without expert assistance and is ready to run applications when it is first turned on. The Intelligent Network Processor is part of the initial release, so the networking goal was met.

Our lab teams operate in an environment that allows them to set their own, often aggressive goals, which they strive toward but do not always achieve. This encourages our teams to take appropriate risks in areas where there is a range of acceptable results. Performance is an area where risk is appropriate. Product quality is an area without much opportunity for risk taking. Although this project did not meet all of its goals, it is considered to have been successful, since the product met the goals considered important by both management and the project team and has sold well.

Initial Design Approach

The originator's first designs emphasized maximum hardware integration. The team considered a single printed circuit board with everything on it, including the controllers for the peripherals. This design would contain no connectors and assembly would consist of snapping things together. This approach would minimize cost at the expense of configuration flexibility. The team chose to add the ability to expand memory and thus avoided being trapped with a small memory while memory use continued to expand.

The single printed circuit board with everything on it might have been fully explored had not organizational considerations within HP brought about its early termination. HP entities have traditionally produced both hardware and the software necessary to make it function. They tend to use revenues from hardware sales to support the software. Although this trend was changing when the Series 37 project began, there was enough concern to justify separating the hardware along divisional boundaries. The team decided not to produce a new version of the peripheral controllers and made the I/O channels separate printed circuit boards.

Another early decision led to the inclusion of operating instructions in the product packaging. This evolved into a pull-out card with instructions on it. The card became unpopular when we faced the issues involved in localizing the card for the other languages we support. The card was eliminated late in the project when it was noted that its slot was contributing significantly to electrical noise radiating from the system.

Active Investigation

The team made attempts to eliminate costly features that had become standard on members of the HP 3000 family. These efforts had some success. A battery to sustain the memory during powerfail is an example of a feature that was eventually retained. A separate service processor is an example of a feature that was deleted. The Series 37 achieves its maintenance functions by putting itself into a different mode of operation (see article, page 17). The system acts as its own service processor. Although there isn't a separate service processor in the Series 37, many features needed to debug system software problems have been made available by a microcoded debug package.

"Keep it simple" became a motto for the team. This led to the decision not to support peripherals that could not share their channels. Since we were channel limited, we couldn't afford to dedicate a channel and didn't want the coordination problems involved in working with other HP divisions to redesign their products. Another decision was to refuse the offer of another division to take control of our terminal connections. Thus, we avoided the management complexity that would have resulted had we increased our dependence on HP entities outside the project's control.

Although we wanted to use the newest subsystems being developed within HP, we didn't want to introduce any part availability problems. We discovered that unless new products were produced with the HP 3000 family in mind, they invariably lacked features required of an HP 3000 system component. Powerfail/auto-restart functionality is a good example of a frequently missing capability.

We are finding that we can develop new hardware faster than the software required to make it function. Because our software resources were committed to other projects, many early decisions were made to minimize the impact on the software development teams. In many cases, emerging products looked very attractive until we realized that the software for them couldn't be developed in time. This situation, together with the other problems we found trying to use emerging products, led us to decide to leverage the huge investment in existing I/O software for HP-IB (IEEE 488) and ATP (Advanced Terminal Processor) peripherals instead of using peripherals that required software development.

Development of new system microcode has been a historic bottleneck. To reuse an existing (and working) microcode set, we attempted to copy an existing hardware design, but couldn't find any that were suitable for VLSI. We eventually built a microcode development team and wrote totally new code.

Development Phase

Our VLSI processor was put into a single gate array to avoid the performance and connection limitations of a partitioned design. We selected the gate array by looking for the largest gate array that was already in production and would require little power (see article, page 7). We also put part of the terminal interface controller into VLSI.

Full simulation was correctly seen as necessary to get the VLSI designs right before they were fabricated (see article, page 13). Much of this work was performed on the most powerful systems available to the design team. Simulation was also used to debug microcode before the hardware was available. We ran the initial part of the system boot software on a simulator. The limiting factor became the effort it took to give the simulator the I/O information that the software

High-Volume Test Strategy

At the beginning of the design phase for the HP 3000 Series 37 Computer, it was clear to the manufacturing team that the production methods used on the existing HP 3000 production line would require a complete revamping if manufacturing were to be successful in meeting the high volume demands projected. The key was clearly in developing a test strategy that allowed high throughput, high confidence of shipping 100% operational systems, and efficient diagnostic tools, none of which could be at the expense of lengthy test times. This meant more than just a new set of diagnostics and test tools.

The R&D team felt confident that the design tools used, coupled with the dramatic parts count reduction, would make this HP 3000 one of the most reliable. But they agreed that the need for manufacturing to be able to build 100% quality systems in high volume had to be addressed. Thus the HP 3000 Series 37 became a springboard to launch the Computer Systems Division into a completely new methodology for manufacturing complex computer systems. Relative to the test strategy, manufacturing engineering felt the major goals were to:

- Establish a new, more effective real-time method of feedback to the lab on the product and product testing.
- Establish a test flow that no longer requires highly technical operators.
- Develop a process that emphasizes good inventory, not test-and-reject-to-be-fixed inventory.

Feedback to the lab was established in an unusual way. The manufacturing team was established before the laboratory prototype phase. Technicians and production engineers worked with the R&D lab during this phase to gain familiarity with the tools and the product. All remaining product phases were performed at the manufacturing site by the manufacturing team with technical support from the lab. Thus the lab team received feedback on the strategy and hardware before concepts were too developed to change. Manufacturing provided weekly summaries of all problems and concerns to the lab for review. This was successful even though the lab and manufacturing were separated by over 150 miles.

The R&D team put great effort into realizing the key diagnostic tests as power-on self-tests (microdiagnostics), which clearly indicate to the operator whether the system is operational. Scripts that detailed step-by-step system verification procedures were established by production engineering early in the cycle. These were later augmented with a Diagnostic Utility System, which allows the streaming of tests, thus virtually eliminating operator intervention.

The Series 37 established a new concept in testing HP 3000 Computer Systems. Production personnel receive completed mainframes in the System Verify area (see Fig. 1) and attach typical peripherals to execute microcoded self-tests and other higher-level tests. The operator is required to ensure that each system completes this testing phase without error. If an error occurs, the failure mode is noted and the unit is rejected. No problem isolation methods are allowed. This lets the operator focus on the system's operation and not merely on getting a system ready to ship by swapping in other material. The defective units are sent to the Defect Analysis stations. These are equipped with all the hardware debug tools and are operated by highly trained technicians, who determine the cause of the defect. Because the entire system is available, these stations are able to locate most of the otherwise elusive failures that occur when hardware is swapped between systems. The causes of defects are reviewed weekly so methods can be established to eliminate the defects from the process. The goal is to eliminate all defects, so that none are found in the System Verify area.

Acknowledgment

Laurie Schoenbaum implemented the memory test used in production.

Dennis Bowers
Manufacturing Engineer
Computer Systems Division

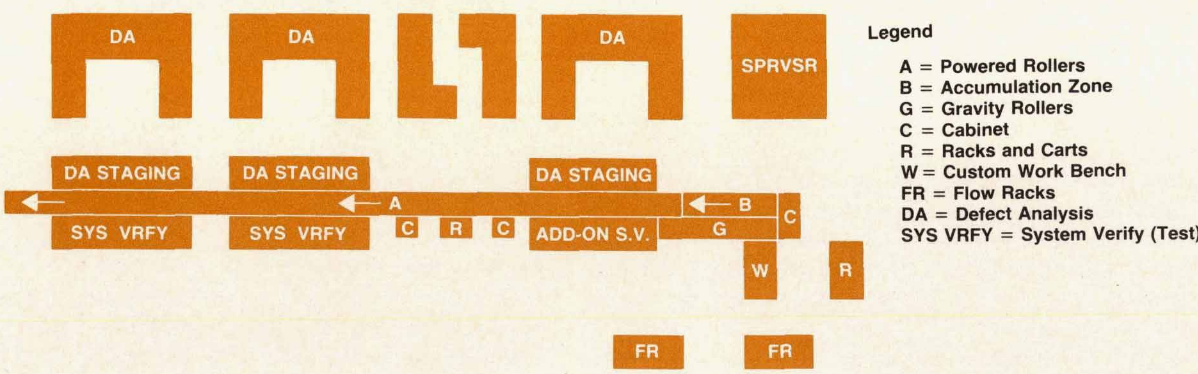


Fig. 1. *HP 3000 Series 37 build-test production line.*

needed to continue to run.

The final key to success was full family membership. We had to make the Series 37 a real HP 3000 in the eyes of the users. Fortunately, we were shooting at a fairly stationary target and were able to achieve this goal. As a result, the HP 3000 software team decided to include the Series 37 in their effort to consolidate operating systems into one version. All of HP's currently manufactured HP 3000 systems can run the same version of MPE (Multiprogramming Executive, the HP 3000 operating system).

The Series 37 had already completed a number of successful production runs in our manufacturing area before the project was completed. This allowed us to ship a large number of systems as soon as the system was released, and gave us a chance to stress a large number of systems environmentally before release. Systems were subjected to high and low temperatures, high humidity, and a packaged drop test. We used the information gained by our stress testing to improve the systems before shipments began.

Conclusion and Summary

The Series 37 is an incremental member of the HP 3000 family. It runs the newest version of the MPE operating system, provides powerfail/auto-restart capability and allows remote support. It is the smallest and lowest-priced HP 3000 there has ever been.

The Series 37 reduces the need for operator control to the point where an operatorless environment is achievable for some customers. It is the most reliable HP 3000 system ever produced and is suitable for the office environment.

The time to start a system from cartridge tape has been greatly reduced from that of older versions of MPE. This is because of the ability to stream the tape during the boot process. The Series 37 contains a real-time clock that continues to run when the system power is removed. This clock allows the boot process to set the time of day without operator input.

Acknowledgments

The Series 37 is the result of Rick Amerson's ideas and a lot of hard work. Peter Rosenbladt led the first half of the project and one of the authors and Alan Christensen led the second half. Rick's team developed the VLSI portion of the CPU and the memory. The rest of the hardware was proposed by Mark Linsky's group. Barry Bronson replaced Mark after the investigation was complete. The author's group developed the system microcode and the service tools. Greg Gilliom led the microcode team during the last half of the project. The other author led the serviceability group after it was separated from the microcode team. The industrial design was done by three teams: Manny Kohli's team worked on the initial design, Gerry Gassman's team took over from them, and Frank Sindelar's team took care of the final set of challenges. The software team was led by Kathy Hahn. Kathy and system manager Hank Cureton led the effort to release the system.

Simplicity in a Microcoded Computer Architecture

by Frederic C. Amerson

A SIMPLIFIED APPROACH to the design of a microcoded architecture can produce a design that is more efficient in its use of silicon than one based on specialized hardware functional units, without sacrificing performance. The HP 3000 Computer, first introduced in 1972, has had a number of different implementations using various degrees of specialized hardware. The most recent of these, the Series 37, is the first HP 3000 CPU to be implemented in VLSI technology. This article describes the design approach used to implement the CPU chip and the efficiencies achieved. From the initial concept of the design to the final working production parts was less than one year.

There are two principal types of computer architecture in widespread use today: *stack architecture* and *register architecture*. Stack architecture is so named because the computation is done on a data stack. Numbers are moved to this stack from memory, an operation is performed leaving the result on the stack, and then the result is stored at some (other) location in memory. Register architecture performs computations in general-purpose registers. Numbers are first moved to one or more registers, an operation is performed leaving the result in a specified register, and the numbers are stored at some (other) location in memory. Some stack and register machine implementations allow operations that use memory operands directly without first

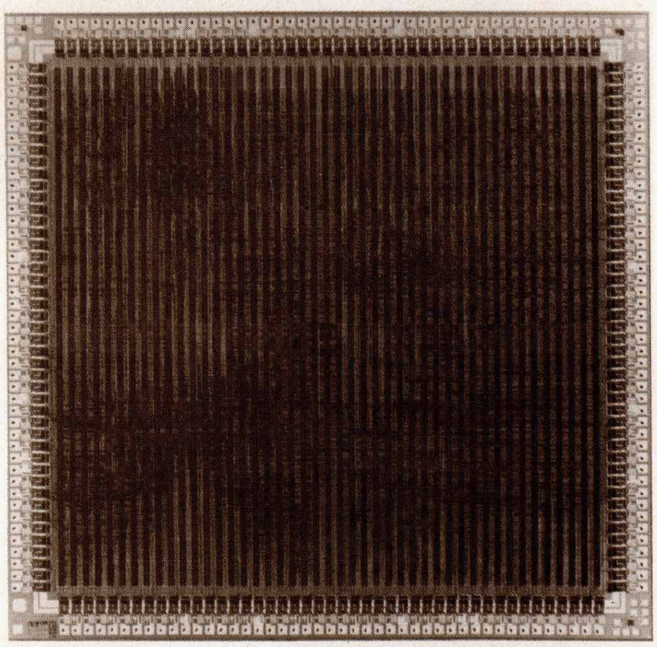


Fig. 1. *The Series 37 CPU chip is a CMOS gate array using nearly 8000 gates.*

moving them to the stack or registers.

The HP 3000 is a stack machine that has a very rich instruction set, which has been enhanced over the years and has grown to provide the user with a robust capability for data processing in commercial applications. Although many of its operations are performed directly on the stack, it is possible to perform some operations using stack operands with memory operands, and others (e.g., COBOL instructions and extended-precision floating-point instructions) using memory operands only. Instructions that process only data on the stack are called *stackops*. In the HP 3000, two of these instructions can be contained in the same space that a single instruction would normally occupy and are referred to as *paired stackops*. This makes for more efficient use of code space in the event that two stackops occur in succession. Another very powerful group of instructions is the *memory reference* instructions. These instructions access a memory operand directly. In some cases it is merely loaded onto or stored from the stack, but in others, computation is performed using the memory operand. There are several addressing modes available for the memory reference instructions, allowing data to be accessed relative to any of several different base registers. A scheme of encoding these, referred to as *Huffman encoding*, enables this important information to be encoded in fewer bits.

Series 37 Design Methodology

Putting a processor onto a single chip is an extensive effort that requires thoroughness and careful attention to detail. The Series 37 presented a particularly difficult challenge because of the powerful HP 3000 instruction set and the flexibility it provides. There exist several predecessors that might have been used as models, but none of these had been implemented on a single chip. The processor offering the closest comparison is the Series 48, which is contained on two boards. Its extensive use of high-density memory parts precludes its implementation in a single-chip CMOS design. The designers of the Series 37, therefore, created a new design, structured specifically for the VLSI technology available.

The Series 37 uses gate-array technology (see photo, Fig. 1), so there is an absolute upper bound to the amount of logic that can be contained in the chip, making it imperative to conserve chip space while optimizing performance. The design approach had to be as simple as possible yet elegant enough to achieve performance goals. Special hardware assists were kept to a minimum, with preference given to simplicity and ease of design rather than maximum performance.

It was decided to eliminate interdependencies between portions of the design as much as possible. This approach gives rise to a multiplicity of independent functional units, each capable of performing one small function without interaction from other functional units. The advantage of this approach is that if one of the unit designs encountered a problem, it did not impact the others. This also allowed different designers to work on separate portions of the design without concern about the impact on other areas.

Instruction Decoding

Because the instruction set is full and the dense encoding does not lend itself readily to a simplified method of decoding, previous implementations of the HP 3000 architecture have used specialized hardware to determine the mapping from the instruction to the microcode that executes that instruction. Generally, this hardware involves a fixed table of entry points in the microcode for each instruction, and a method of mapping the instruction encoding into this table. The table requires one entry for each instruction. However, it is frequently easier to duplicate entries rather than create a mapping that can resolve each instruction to a single location in the table. In the HP 3000 Series 64, the upper ten bits of the instruction are used to address this table directly, so there is no logic to map instructions to a smaller table. This is effective for a high-performance machine, because time is not lost for the function of the mapping logic. For its predecessors, where memory components were more expensive and less dense, mapping logic was more effective.

When placing an architecture onto a single chip, memory

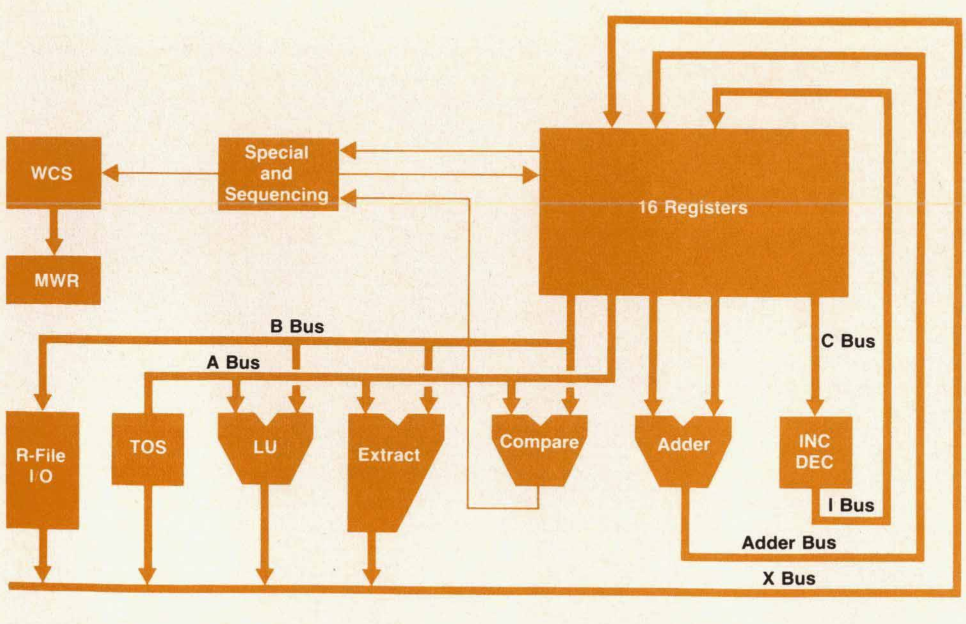


Fig. 2. *Instead of the usual arithmetic-logic unit (ALU), the HP 3000 Series 37 CPU has an adder and a separate logic unit.*

and mapping logic once again become expensive, so in the case of the Series 37, this circuitry was eliminated. Even more important than eliminating the circuits was reducing the complexity of the design. Because it is so difficult to fix mistakes in VLSI designs once they have been built, it is important to use a methodology that is conducive to error-free design. Special instruction decoding and mapping circuitry must be accompanied by logic to alter the normal sequence of microinstruction execution. Determining whether the next microinstruction address should come from the regular microcode sequencing logic or from the special instruction lookup logic is a complex, error-prone task. However, eliminating the special decode function is expensive in performance because it adds the time required to decode an instruction in microcode to the execution time of each instruction.

It was necessary to find a method of decoding instructions that is both simple and fast. A jump table is a simple method of decoding, but it is not fast. However, by making decoding a fast operation while allowing other operations to occur in parallel, the Series 37 is able to overcome the inherent slowness of a jump table. To decode instructions, the microcode extracts the upper eight bits of the instruction and then branches to a jump table of 256 entries. The most frequently executed instructions are decoded quickly by this method. The time required to perform the decode is only two cycles. It is not wasted, however, since instruction fetch, incrementing of the program counter, and testing for interrupts occur in parallel with the instruction decode.

16 Registers

Fig. 2 is a block diagram of the CPU chip. The heart of the chip is a bank of sixteen registers controlled by microcode. Their definition does not change from instruction to instruction. However, it is possible to use them as scratchpad registers if the information they contain is saved elsewhere. The data contained in them is maintained by microcode convention only; there is no hardware requirement for a particular register to contain particular data. They usually contain the important base register information needed by high-execution-frequency instructions. Not all of the registers can be used for any purpose whatever. In fact, most of them double as special registers for a particular specialized function. See Fig. 3 for a list of these registers.

Top-of-Stack

Because the HP 3000 is a stack machine, many computations are done with top-of-stack (TOS) data. To the programmer, the TOS appears to be in memory, but this is very inefficient, so it is necessary to provide some hardware support for the stack. Four registers are used to contain (up to) the top four elements of the memory stack at any time. These registers, together with all of the supporting logic to keep track of the stack data, are contained in a separate functional unit. There are special operations that allow the microprogrammer to control these registers and to make the various tests necessary to implement efficient algorithms for their control. Thus this important function is retained so that performance is not compromised, yet the simplicity of the architecture is preserved. The TOS logic consists of four registers to contain data, a two-bit *namer* register to identify the data register currently named as the top of the stack, a three-bit stack-valid counter to indicate how many of the data registers contain valid data, and a two-bit adder to allow accessing relative to the top of the stack.

Extractor

One of the most powerful functional units is the extractor. It concatenates two sixteen-bit quantities to form a 32-bit number. Any arbitrary sixteen bits from within this 32-bit number may be selected and then any arbitrary rightmost or leftmost bits may be selected from this sixteen-bit quantity, effectively allowing the extraction and either right or left justification of any arbitrary bits from a 32-bit quantity. Additionally, the contents of one of the sixteen registers may be selectively ORed logically with the result. Although this sounds somewhat convoluted, it is actually quite straightforward, simple to implement, and extremely powerful (see Fig. 4). Its most creative use is replacing the instruction decoding logic found in earlier implementations of the HP 3000 architecture, but it is used in several other decoding situations as well. A target address into a jump table can be created in a single microinstruction and then control passed to that address on the next microinstruction.

Logic Unit, Comparator, and Adder

One of the most common operations performed in a microcoded architecture is logical arithmetic: AND, OR, etc. Because these operations can be conveniently generated as byproducts of arithmetic operations of addition and subtraction, they are usually included with the arithmetic unit, which then becomes the arithmetic-logic unit or ALU. The CMOS logic used in the Series 37 required special considerations to achieve high performance, making combining

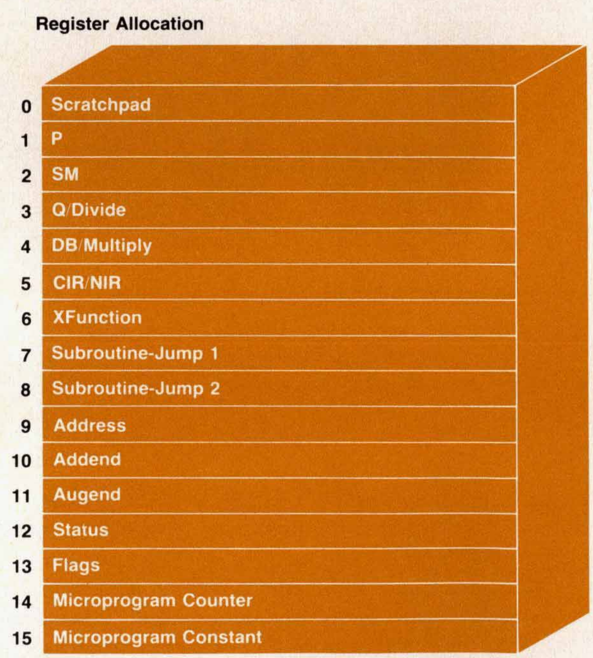


Fig. 3. *A bank of sixteen registers allocated as shown is an important part of the Series 37 CPU chip.*

Using a Translator for Creating Readable Microcode

The HP 3000 Series 37 Computer is built around a very powerful, flexible microprocessor. Because of this flexibility, the control language for the microprocessor is complex and can be hard to understand.

The major feature that makes the code difficult to read is the ability to do many different operations concurrently. One instruction can simultaneously increment, do logic operations, add, and do an IF-THEN-ELSE control branch. Clearly, it is important for the microprogrammer to keep track of all of these simultaneous operations.

To make line-by-line analysis in a debugging environment easy, a very rigid 17-field source language was developed to express these constructs (see Fig. 1). However, none of the system microcode was written in this form. Although allowing easy analysis of what the micromachine is doing on any given clock, the splitting of a single function into multiple fields and then interleaving these fields with other operations makes the intent of the microcode almost impossible to follow accurately.

To overcome these limitations of the rigid fixed-field language, a much more flexible language was developed (see Fig. 2). All of the system microcode was written in this language. Its key features are:

- Any construct can be expressed (the microprogrammer is not limited by the language).
- The language is free-field. Spacing, column alignment, and the positions of new lines and/or comments are not important.
- The language allows an operation-by-operation expression of what is coded (the DEC function in Fig. 2 is expressed independently of the ADD function).
- The user can DEFINE registers and constants to improve readability.

```
LABEL... A.. B.. C.. ADD IN I.. R-FCN-XCNTR-SEC X.. SPEC JTD JFD CONSTANT
LOOP     R1  R2  R0  R9  -1 R0  SZL  Z5      S5 R1  JL=0 R8  R15 LOOP
```

Top Line: Field Names
Bottom Line: Typical Instruction

Fig. 1. *Fixed-field source language specification.*

```
 Define  Counter = R0,
         Left = R1, Right = R2;
Loop:
 Dec (counter) - > counter /                        *Decrement
  Add (R10,R11) - > ADR /                           *compute address w/ADD
  ExtractR (Left | | Right (10..20)) - > Left /     *use extractor
  If Logic = 0 then ReturnSub1 else Loop;           *IF..THEN.. ELSE
 Undefine counter, left, right;
```

Fig. 2. *Flexible language used for writing system microcode.*

This free-field language is implemented as an independent source-language preprocessor. It is designed to complement, rather than replace, the fixed-field language. This preprocessor (known as the Translator) converts the microprogrammer-written source code into the fixed-field language. This is then assembled using a separate assembler into executable object code. The powerful user-defined-command (UDC) capability of the HP 3000 has been used to make these two tools appear to the user as a single well-integrated one. The final assembly listing (see Fig. 3) contains the source code as written, the translated fixed-field code, and the emitted object code, all presented in a format that is easily read and understood by the microprogrammer.

Skip La Fetra
Project Manager
Computer Systems Division

Object Code | **Fixed-Field Source Language** | **Source Code as Written**

```
                         | LABEL... A.. B.. C.. ADD IN I.. R-FCN-XCNTR-SEC X.. SPEC JTD JFD CONSTANT |
                         |                                                                          | Define  Counter = R0,
                         |                                                                          |         Left = R1, Right = R2;
                         |                                                                          |
                         |                                                                          | Loop:
                         |                                                                          |  Dec (counter) - > counter /                     *Decrement
                         |                                                                          |   Add (R10,R11) - > ADR /                        *compute address w/ADD
                         |                                                                          |   ExtractR (Left | | Right (10..20)) - > Left /  *use extractor
0000: 1207 0555 1880 FFFF | LOOP  R1 R2 R0  R9 -1 R0  SZL Z5      S5 R1 JL=0 R8 R15 LOOP           |   If Logic = 0 then ReturnSub1 else Loop;        *IF..THEN.. ELSE
                         |                                                                          |
                         |                                                                          | Undefine counter, left, right;
```

Fig. 3. *Final assembly listing shows source code as written, translated fixed-field source language code, and object code.*

these two functions difficult. Also, an arithmetic unit that can both add and subtract requires more circuitry than one that can only add. Therefore, no subtract function is available to the microcode; only add is available. Instead, a separate logic unit provides the functions normally associated with an ALU, including the one's complement necessary for subtraction. To subtract, the microprocessor must form the one's complement of the subtrahend and add it to the minuend with a carry. Because subtract is a relatively infrequent operation, performance impact is minimal.

In examining the functions normally performed by microcode to implement instructions, it was discovered that the arithmetic functions are used primarily for comparing and not for arithmetic. This comparing is generally a comparison of an address with certain bounds registers to see if the address lies within the area that can be accessed by the user. If it is not, a bounds violation is generated by the microcode and the software aborts the program that is run-

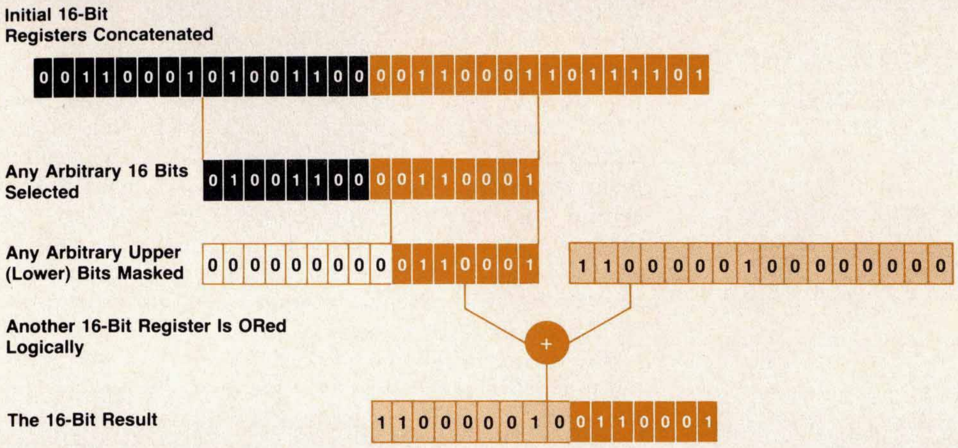


Fig. 4. *The extractor, a functional unit of the Series 37 CPU chip, replaces the instruction decoding logic found in earlier HP 3000 implementations.*

ning. Typically, previous HP 3000 machines have special hardware circuitry to examine the carry signal from the ALU and force the microcode to specific error addresses when a compare indicates a bounds violation. In keeping with the philosophy of simplicity and explicit microcode control of the machine at all times, hardware to force certain microcode addresses was summarily rejected. Instead, a simple comparator circuit is included, allowing a quick comparison of two addresses and a transfer of control based on the result.

The specialized comparator circuitry, which is completely self-contained, is able to perform all of the necessary bounds checking with no performance penalty.

Incrementer

Although the microcode seldom needs to perform arithmetic beyond addition, it frequently needs to either add or subtract one from a number. To provide this capability, an incrementer/decrementer is available and can be used on every cycle. This proved to be one of the most valuable additions to the design, because it allows an arithmetic operation to occur in parallel with a separate logical operation. The incrementer can perform four functions: add one, subtract one, pass unchanged, and add two.

One useful function of the incrementer is to simulate subroutines. Because the microprogram counter is a register like any of the other registers, the incrementer can pass its value to one of the subroutine jump registers easily. If the incrementer is needed on the line that jumps to the subroutine, it can pass the microprogram counter incremented by one on the line before, or by two on the line before that. The return from a subroutine is accomplished by selecting this register as the next address for the microprogram sequencer.

Microcode Address Selection

The next line of microcode to be executed is selected from one of four registers by the current line. Two fields in the microinstruction specify the next line of microcode; these are the *true target* and the *false target* fields. There are 32 conditions that can be tested by each line of microcode. If the condition is met, the address for the next line of microcode comes from the register specified by the true target field. If not, the address comes from the register specified by the false target field. Each bit in the flag register can be independently tested, as can sixteen other conditions and bits in the machine. The four registers that can be used as the source for the next address are the microprogram constant (useful for jumps to specific addresses), the microprogram counter register (executing microcode in sequence), and the two subroutine-jump registers. The separate true and false targets allow complete symmetry as well as the freedom to execute either of two lines, neither of which is the next line in sequence.

Diagnostic and Test Capability

Because all of the registers are easily accessible and there is very little specialized hardware, the design does not require much additional circuitry dedicated to diagnostic test purposes. There are special commands allowing access to bits that could not otherwise be accessed, but no other specialized logic. This makes testing a straightforward task instead of the labyrinthine jumble of convoluted code required with conventional designs containing a multiplicity of untestable bits. Since each of the functional units has limited well-defined side effects, it is easily verified that a failure of one of these has occurred, and that no others have failed.

Performance Results

The HP 3000 Series 48 provides a good comparison for the design of the Series 37. Although the Series 48 has specialized hardware to help with instruction decode and bounds checking, it is basically a much simpler design than the Series 68 and contains far fewer circuits. Measured in the same terms, it contains about twice the circuitry of the Series 37. Therefore, it is of particular interest to compare the relative efficiency of the two designs. Because the memory reference instructions are so critical in the instruction mix, it is especially worthwhile to compare these instructions.

The Series 37 has an immediate disadvantage compared to the Series 48 because it does not have special instruction decode circuitry. Since this function is executed by microcode, it must be slower. Also, there is no hardware prefetch of the following software instruction as in the Series 48, so this must also occur in microcode. Consequently, the first few cycles of every instruction are spent initiating a memory

Booting 64-Bit WCS Words from a 32-Bit-Wide ROM Word

The processor design for the HP 3000 Series 37 Computer emphasizes small size, low cost, and reliability. To realize this, the computer is designed with few parts and without the separate control processor used on other members of the HP 3000 product line.

Because the Series 37 does not have a separate control processor to load WCS before launching the main processor, the main processor must load its own microcode. It does this by executing a short power-up program directly from ROM whose sole purpose is to initialize the writable portion of control store.

The Series 37 requires four ROM chips to hold all of the cold-load and self-test microcode. These chips hold 128K bits of information each. However, eight ROMs would be required, each supplying eight bits of data at a time, to supply the full 64-bit microinstruction word required to run the processor. To avoid the need for these extra four ROMs, a scheme was devised that allows the processor to boot with only 32 bits of each microinstruction supplied.

This scheme is simple in principle: choose a subset of the processor's capability that is adequate to boot the system but simple enough that it can be expressed with 32 or fewer bits of microinstruction, and then force the unused 32 bits of the 64-bit control word to constant values. For example, the CBUS field allows access to all 16 internal registers (requiring four bits of control). The boot code only uses two of these registers. By tying three control lines to a constant zero we require only one bit of ROM control for the CBUS field. Similar reduction of control requirements was done with each microinstruction field as follows:

ABUS	2 of 4	bits used
BBUS	4 of 4	bits used
CBUS	1 of 4	bits used
ASTOR	0 of 2	bits used
INC/DEC	0 of 2	bits used
ISTOR	2 of 4	bits used
Indirect	0 of 1	bit used
X-control	12 of 12	bits used
XSTOR	3 of 4	bits used
Parity	1 of 1	bit used
Special	1 of 6	bits used
Jump	4 of 4	bits used
Constant	0 of 16	bits used
	30 of 64	bits used

The limited capability provided by this control is enough to step byte-by-byte through one ROM and transfer its contents to writable control store. After confirming that the microcode was successfully transferred, control is passed to the newly loaded code. This code is of full 64-bit width and has all of the power of the processor available to it. Thus it has the capability to run extensive microdiagnostics, load more microcode from the other three ROMs, and boot the operating system.

Skip La Fetra
Project Manager
Chris Shaker
Development Engineer
Computer Systems Division

fetch of the following instruction and decoding the current instruction. By using the capability of the extractor, the microcode is able to decode an instruction and transfer to the first line of the instruction with only three cycles of overhead. The first cycle extracts the upper eight bits of the instruction and combines them with the address of the jump table, storing this final address into a jump register. The second cycle transfers control to the address specified in the jump table. The third cycle is the line of microcode in the jump table that transfers to the instruction itself. This third cycle can also begin execution of the instruction, thus reducing the effective overhead.

It would seem obvious that with a three-cycle overhead, the instruction must be slower than its counterpart on the Series 48. Indeed, the performance of the Series 37 is less than that of the Series 48, but this is because of the Series 37's lower clock frequency and not the efficiency of the microcode. Comparing the number of cycles required to execute the instruction rather than the amount of time, the surprising result is that the Series 37 memory reference instructions require approximately the same number of cycles as the Series 48. When the overall performance of the entire instruction set is compared, the Series 37 and the Series 48 require approximately the same number of clocks to execute a typical mix of instructions for the HP 3000.

Although the amount of logic is far greater to realize an architectural implementation for the Series 48 compared to the Series 37, the relative amount of work done for each machine cycle is the same. The simple approach used in the design of the Series 37 resulted in an extremely efficient use of silicon. A design need not be complex to be cost-effective.

Acknowledgments

There were a number of people whose dedication and patience made this design possible, and incredibly in less than six months from initial concept to final tape release. Paul Smythe and Greg Gilliom led the microcode effort working with Brian Feldman and our prolific summer student, Michael Goo. Norm Galassi generated the test microcode which was successfully used in the simulations to ensure that the first-pass chips were the last-pass chips. Karen Murillo designed several of the functional units on the chip. The tools were created by Frank Hublou, who made sure that there were no problems interfacing with any of the other groups with whom we worked, and Daryl Allred, who was able to respond to the many requests for specialized tools in a remarkably short time. Special recognition is due Barry Shackleford, without whose encouragement, insistence on maintaining simplicity, and dedication to excellence in organizing and leading the design, this project would not have been possible.

Simulation Ensures Working First-Pass VLSI Computer System

by Patria G. Alvarez, Greg L. Gilliom, John R. Obermeyer, Paul L. Rogers, and Malcolm E. Woodward

ONE OF THE OBJECTIVES for the HP 3000 Series 37 project team was to produce two fully functional VLSI chip designs, each in a single pass with no errors. The advantages of this objective are obvious. With first-pass chips, the project schedule is shortened considerably, leading to lower project costs. This also frees laboratory resources to be applied to the next project sooner.

This goal seemed formidable when first proposed, since it had never been done before at HP's Computer Systems Division. Formerly, all designs were done in several passes: a breadboard, a lab prototype, and a final-release version. At each stage, the design was fine-tuned to match the specifications of the project.

To release single-pass designs required careful investigation and thorough definition before the logic was laid out. Gate arrays were chosen for the two chips, and their simple, regular architecture was a strong ally in the construction of first-pass chips. However, the tool that absolutely guaranteed first-pass VLSI chips was the design simulator, FTL. By using a simulator capable of logic and timing simulation, the project team was able to detect errors before a chip was masked or fabricated. This early error detection allowed faster turnaround on logic design and encouraged more effective verification.

FTL is an acronym for Faster Than Light, a name given to the simulator by its creator. FTL uses the output files of the Design Capture and Documentation Facility (DCDF), an interactive menu-driven logic design tool used on the Series 37 project to enter and generate circuit schematics. During the design stage of the Series 37 project, DCDF allowed engineers' designs to be captured for input to the simulator.

FTL lets the user watch a design in action as inputs are provided. It is written in IBM 370 assembly language, and in our case, runs on an Amdahl V6. The Amdahl provided the design team with immense horsepower, greatly reducing the time necessary to simulate the design.

FTL Features

FTL provides many features to assist the hardware designer in designing and debugging a circuit. It can combine several different design files into a single simulation, so that several designers working on the same chip can simulate their portions of the circuit and then combine their designs into a single simulation to see how the different parts of the circuit work together. Chip designs can then be combined with board logic so that entire boards can be simulated. Finally, several boards, such as CPU, memory, and I/O, can be simulated together.

A nice feature of FTL is the ease with which a designer can look at a logic signal. Like a conventional logic analyzer, the FTL user interface is a display screen (see examples, Figs. 1 and 2). All signals on the screen are labeled with the names used in the DCDF design file. To view a logic signal, the engineer simply inputs the name. The logic signal is added to the screen, and the engineer can observe the logic transitions. Logic signals can be viewed singly or as octal or hexadecimal representations if several signals are grouped in a bus structure.

Through an addition to the simulator made specifically for the Series 37 project, the design team was able to preload RAMs and ROMs on the board from separate files.

CPU Chip Simulation

The two VLSI gate array chips in the Series 37 are the CPU chip and another gate array that is used in the terminal interface controller (TIC).

A difficult problem for most designs of chips as large as the CPU gate array is the generation of good test vectors. In many cases, test vectors are painstakingly generated by hand, with an engineer toggling each input and observing the outputs to see if they respond as expected. This often

```
                              MACRO SCREEN
----+----1-|--+----2----+----3----+----4----+----5----+----6----+----7----+
SET          0 -SYNC
SIMULATE    3
SET          1 -2XCLK
SIMULATE    3
SET          1 -SYNC
SIMULATE    3
SET          0 -2XCLK
SIMULATE    3
SET          1 -2XCLK
SIMULATE    3
SET          0 -SYNC
SIMULATE    3
SET          0 -2XCLK
SIMULATE    7
```

Fig. 1. *Macro screen produced by the FTL simulator for generating simulation clocks for the terminal interface controller (TIC) board.*

```
       SIMB              DMA SEQUENCER            LYNX BUS          CMDSM
  ------------        --------------------      ------------     -I12345678-
  IN                              0:7 8:9        IN              100000000
   DATAIN   = 0       DMASQ    = FF    0          BUSEND   = 1
   CMDIN    = 0                                   BUSOP    = 0       RWSM
   DONECY   = 0        MEMORY                    OUT             -I123456---
  OUT                 --------------------        BUSGO    = 0   1000000
   STRTCYC = 0        MEMWRT   =                  POLLB    = 1
  BI                  RAMIO    = 00 0             FRZ      = 0
   SIMBWRT = 0        MEMCY0-4 = 0 00000         BI
   SIMBIO   = 1000     MISC                       LBDATA   = 00
   (INV) .. EFFF      --------------------        LINE NO = 0
RESET LINES     !     PFW = 0  IRQ      = 0
===============!CLOCKIN = 1  SYNC     = 1
PON .... 0     !SYNCIN  = 0  2XCLK    = 1
CINIT ... 1    !CLKGT0-3  0000
               !
               !
               !
 SCOPE= TICBD             FROM= 00 TO= 00   SCR=   0              X=  1 Y=  1
COMMAND= PRINT       #CYC=      1                         ACCEL TIME=          12
TXT:
```

```
                         SIMB INTERFACE
                      !                     !                         !
SIMBIO ....... 1000   !IMBMRQ .. 0          ! COMMAND STATE MACH !IMBRQ ..... 0
DRVSIMB ...... 0      !RIOC .... 0          ! ---I - 12345678 -- !HLDRQ ..... 0
SIMBI ........ 1000   !WIOC .... 0          !    1   00000000    !MYDO ...... 0
SIMBO ........ 0000   !OBII .... 0          !                    !RWENT1 .... 0
LEGAL ........ 1      !RSTB .... 0          ! READ/WRITE STATE M !RWEXIT4 ... 0
IMBA0;2;4:7 .. 111111!SMSK .... 0           ! ---I -  123456  -- !RWEXIT5 ... 1
IMBRD ........ 0      !RDGT .... 0          !    1    000000     !RWENT6 .... 0
DMARD ........ 0      !WRTGT ... 0          !                    !
DATIN ........ 0      !RCONFIG . 0   E/F !DATA BUS .... FFFF   !STARTCY ... 0
DNCY ......... 0      !RDIAGB .. 0   B   !MYADN ....... 0      !
COMMAND ...... 0      !RDIAGA .. 0   A   !MYDDN ....... 0      !
PONB ......... 0      !RINT .... 0   8/9 !SIMBDAT ..... 0000   !
PFWB ......... 0      !WDC ..... 0   B   !                     !
CMDDN ........ 0      !WDIAG ... 0   A   !SLAVETO ..... 0      !
DFIMB ........ 0      !WBDEN ... 0   9   !                     !
CMDINL ....... 0      !WTPTR ... 0   8   !                     !
DONECYL ..... 0       !                  !CLKGT0-3 .... 0000   !
 SCOPE= TICBD             FROM= 00 TO= 00   SCR=   1              X=  1 Y=  1
COMMAND= PRINT       #CYC=      1                         ACCEL TIME=          12
TXT:
```

Fig. 2. *(top) FTL simulator screen showing overall TIC board activity. (bottom) FTL screen showing detailed SIMB (synchronous intermodule bus) activity.*

leads to bored engineers reaching a frustration limit and releasing a circuit before it is fully verified.

In the case of the Series 37 CPU chip, we were building a machine that had its own microcode structure. In addition, the CPU and memory boards were also being designed in DCDF, so it was possible for FTL to simulate them together. Taking this one step farther, it was possible to load actual microcode into simulated ROMs on the CPU board and execute this microcode on the simulator. Therefore, we used the self-test microcode for the system, which had to be written for later use in manufacturing and field support, as the first test program for simulation of the design using FTL. The first version of the self-test was approximately 1000 lines of microcode and took three hours to run on the Amdahl. Using the clues provided by this first simulation, the CPU designers reworked their design and rechecked it, while the self-test engineer expanded those tests into new areas of the design. This same philosophy was used on the memory and I/O boards.

By modifying the FTL simulator slightly, it was possible to generate test vectors for the gate array chip automatically. The microcoded self-test was executed on FTL and the inputs and outputs of the gate array were recorded. These inputs and outputs were later used by the gate array vendor to verify the first prototype chips and to verify the chip timing and parameters using the vendor's simulation equipment. The vendor's simulation after routing and masking of the chip led to the final timing specifications for the chip.

TIC Gate Array Simulation

The other gate array in the Series 37 system, a 4000-gate VLSI chip, is used in the terminal interface controller (TIC). The entire TIC, including the gate array, was designed in DCDF and simulated using FTL. In the TIC case, the chip is not a processor capable of executing a self-test. Instead, it consists of several complex state machines and control logic. To simulate it, a special assembler was written that made it possible to write a verification test in a simple, high-level language that corresponds to the functional operation of the chip. The assembler translated the high-level functions specified into the proper inputs for the chip. This verification test was then executed in a pseudoboard environment created to aid in the simulation. Around the VLSI TIC chip was added a sequencer, a RAM to supply the input signals, the DMA state machine ROMs, and the TIC register RAMs (see Fig. 3). The DMA state machine ROMs were loaded with the state machine control. The input control RAM was loaded with the patterns from the assembler and was accessed via the sequencer. The test vectors for the gate array chip were collected from the chip inputs and outputs during this verification test. The assembler made it feasible to generate an exhaustive set of test vectors for the gate array. Thus it was a major factor in the design of a first-pass TIC gate array chip.

As a result of these methods and tools, the CPU and TIC gate arrays had only one design pass. The chip designs used in the first hardware breadboards are still used today in production units. The next step was to achieve the same results at the printed circuit board level.

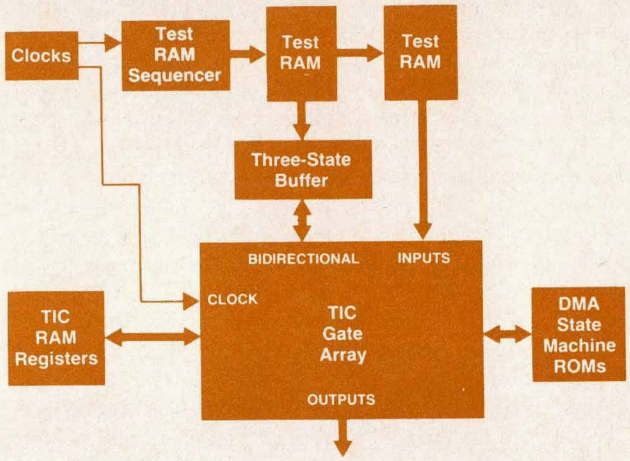


Fig. 3. *Pseudoboard environment created to aid in the simulation of the VLSI TIC (terminal interface controller) gate array.*

Models Created

Before simulation of any of the boards could begin, we had to create models for each of the TTL and CMOS parts, PLAs, and other special parts used in the design. The models had to be understandable to the FTL simulator. Some basic building blocks were created to aid in the modeling of standard logic parts. These basic blocks were generic OR, AND, and XOR gates and various types of flip-flops and memory elements. Each standard part was modeled using these basic blocks. Then the model was verified by testing just that part using FTL. This process was tedious, but necessary to ensure the accuracy of the system simulation.

The next step was to simulate each of the printed circuit assemblies and custom VLSI chips separately. All of the printed circuit boards and custom VLSI chips were simulated more or less in parallel by the individual design teams. FTL lends itself to parallel simulation, thereby saving a great deal of time. The custom VLSI CPU gate array chip was simulated first, as described previously. After this simulation was complete, the next step was to begin simulation of the CPU board.

In simulating the CPU board, first the clocks were connected to the board. An 8 × clock drove the CPU clock as well as the system clocks. This clock, 8XCLK, was the driving clock used during simulation. For each eight ticks of 8XCLK, there was one tick of the CPU clock (CLOCK). The screen of the simulator was set up so that the particular nodes being tested were displayed on the screen. Any of the nodes on the board could be displayed. Once the screen was set up, the inputs to the particular subassembly of the CPU board were put into their desired states. Then the required number of clock cycles was entered (eight for one CLOCK

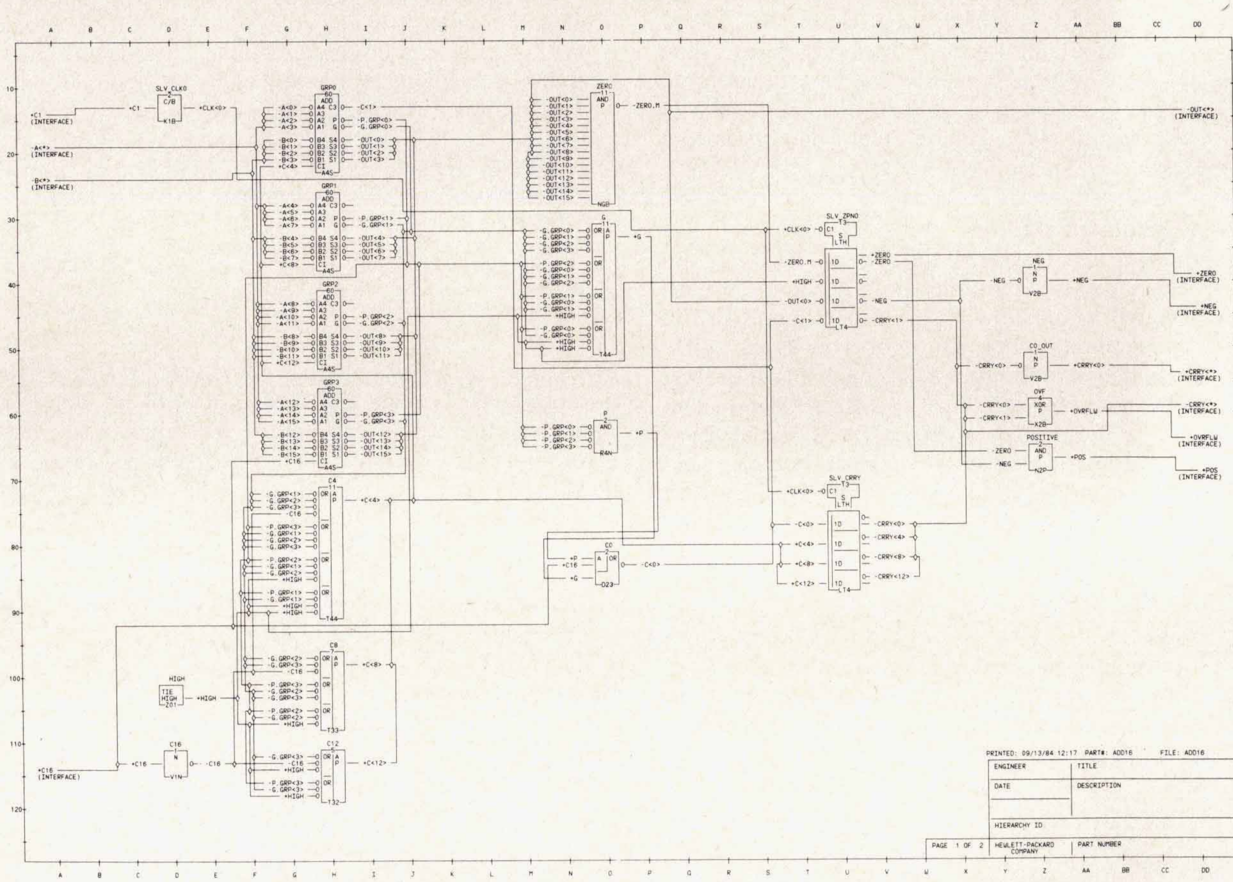


Fig. 4. *An ALD schematic. ALD stands for Automated Logic Drawings, a subsystem of the Design Capture and Documentation Facility (DCDF) used in the design of the VLSI chips for the HP 3000 Series 37 Computer. DCDF output files are the inputs to the simulator, which is called FTL.*

cycle), and the simulation was begun. Once simulation had ended, the final state of each of the nodes on the screen was displayed. No intermediate results were available. If intermediate results were desired, the number of clock cycles to simulate could be reduced.

Once all of the subassemblies had been simulated, the processor gate array was connected to the CPU board, and the simulation of the entire CPU began. This was done by loading test microcode into the simulation control store memory. At this point the microcode controlled the operation of the CPU, so that different code was written to test the different functions of the CPU.

After the CPU and memory simulations were completed, the CPU and the memory were simulated together. The peripheral interface channel (PIC) was added to the system simulation model after the CPU/memory simulation had been completed succesfully. Finally the TIC was added to the system simulation model after the CPU/Memory/PIC simulation and the VLSI TIC gate array chip simulation had been completed.

During every phase of the simulation, designs were corrected or modified as necessary to ensure correct operation of the system. The result of the individual and system simulations was a set of first-pass printed circuit boards that were able to run as a computer system. Both of the custom VLSI chips also worked on the first pass. This is a major accomplishment and was only possible with the help of FTL and Delay, a timing analysis program.

The full system simulation model was used even after the first successful printed circuit boards were up and running. The system model was used to check out new or anticipated changes to the system before they were ever implemented.

Providing Documentation

During the design stage, as the Design Capture and Documentation Facility captured the engineers' designs, ALD (Automated Logic Drawings), a subsystem of DCDF, provided documentation in the form of schematics. DCDF runs on an Amdahl 470 Computer. Through an MRJE link, the schematics are transferred to a remote HP 3000 Computer and are printed out on an HP 2680A Laser Printer.

With the facilities available through DCDF, complete and up-to-date documentation can be attained anytime a new design is created or a modification is made.

Schematic Generation

Fig. 4 shows an example of an ALD schematic. A feature of ALD is its autorouting facility. All of the circuit blocks on a single ALD page having the same interconnect are automatically routed together. ALD gathers all connectivity information from the DCDF file and automatically routes one signal line to another. Routing executes a variation of Lee's algorithm[1,2] using direction numbers. The algorithm finds the paths for routing the signals in the design. Through the routing facility, an engineer can specify the maximum number of bends in the line joining the signals. Should the autorouter not find enough room to draw a wire from one gate to another, the line is not drawn, although the signal name of the line is shown so it is not mistaken for no connection.

All logic symbol pictures come from a master library or catalog. For example, when a NEW command is executed from the DCDF gate editor, an element with the number specified is fetched from the catalog. A pictorial description of that element and other documentation elements (BLOCK TYPE and BLOCK NAME) are brought to the terminal screen. All design elements are automatically given a default name as they are inserted in the design, but at any time the engineer can replace this default name with a user name. The designer may place the logic symbol at any valid location on the schematic, and when it is printed, the picture appears with the default or given block name.

The schematics generated by ALD are typical of most schematics, but they also provide the following:

- Global comments: A designer's personal comments can be included on the schematic and placed and displayed as desired.
- Title blocks: A box is automatically placed in the corner of a schematic. Information such as designer name, block name, and block function can be placed here.
- Hierarchy information: Hierarchy information is provided by ALD so that a designer can tell what level of the design is being displayed.
- Cross reference: The cross reference generates a symbol table listing for bus cross references and connectivity reports. It includes a list of the external signals to the circuit, and it includes the interface signal name, connector pin number, and signal nature. All the interconnects in the schematic are alphabetically listed with all the connection points where that signal is attached.

  The following is a sample cross reference:

```
+MPY_IN<*>
GATE-NAME    X  Y  PG  TYPE CP          BUNDLE

MPY_0        M  20  1  OUT  +OUT        <0>
OUT_SLV      L  20 47  OUT  +Q          <0>
S4           G  64  1   IN  +D_0<0>     <0>
```

In this list, GATE-NAME refers to the name of the gate where the connection is made, X is the X location of the connection point based on a scale printed on the schematic, Y is the Y location of the connection point based on a scale printed on the schematic, PG is the page of the schematic where the connection point is located, TYPE is the type of connection point (input, output, bidirectional), CP is the internal pin designator for the node, and BUNDLE is used to show which interconnect signals from the bundle are attached to the node.

Acknowledgments

We would like to acknowledge Daryl Allred for the work he put into all of the design tools (DCDF, FTL, Delay) so they could be used in the design of the Series 37. The CPU self-test and test vectors were contributed by Norm Galassi.

References

1. M.A. Breuer, *Design Automation of Digital Systems, Volume I: Theory and Techniques*, Prentice-Hall, Inc., 1972, pp. 312-313.
2. J.H. Hoel, "Some Variations of Lee's Algorithm," *IEEE Transactions on Computers*, Vol. C-25, no. 1, January 1976, pp. 19-24.

Creative Ways to Obtain Computer System Debug Tools

by William M. Parrish, Eric B. Decker, and Edwin G. Wong

WHEN SOMETHING GOES WRONG with a computer system, it is imperative that the problem be found and corrected quickly. A determination of whether the problem is in the hardware or the software must be made, and the faulty hardware or software module must be replaced. In the HP 3000 Computer, a major diagnostic tool is the maintenance panel or debug panel. In the HP 3000 Series 37, an off-the-shelf microcomputer is used for the maintenance panel. For software diagnostics, the standard HP 3000 debugging facilities are supplemented by a virtual software debugging panel called SoftPanel, which is implemented in microcode. The maintenance panel requires extra hardware and is used in the factory and optionally in the field. SoftPanel is a built-in tool, available in any system at any time.

System Consoles

Historically, HP 3000 systems have had consoles that are combinations of hardware and firmware. These devices provide operator functions such as loading, starting, and dumping the system, routine diagnostic functions such as running built-in self-tests and checking hardware I/O configurations, and firmware, hardware, and software debug facilities.

The earliest consoles consisted of a special interface card on the CPU backplane and large assemblies of LED indicators and switches. On the HP 3000 Series II and III, these consoles were made service-only tools, and the operator functions were put into smaller panels which were actually shipped to customers. If the diagnostic capabilities of the maintenance panel are required, a Customer Engineer brings the panels to the customer site and connects them with several bulky cables.

Later consoles, those for the Series 44 and 64, for example, also use a special interface card on the CPU backplane, but are designed to work in parallel with the MPE system console. A special sequence of characters is employed to get the attention of the maintenance panel functions, and the commands can be entered through the same terminal used for the MPE console and the operator's session.

The Series 64/68, with its large number of assemblies, requires all boards on the CPU backplane to contain shift strings, which can be read out and written by the console (known as the Diagnostic Control Unit, or DCU) to allow detailed troubleshooting of the different assemblies and data paths. By this mechanism, information contained in storage elements on any assembly can be read and modified. Special microdiagnostics can be loaded through a flexible disc drive connected to the console terminal.

Series 37 Requirements

The Series 37 is designed to be a high-volume, low-cost system relative to other HP 3000 systems. As such, we needed to minimize the special hardware required for the console. We also had to have the debugging system available almost immediately after the receipt of first VLSI parts, so there was not a lot of time in the schedule to debug our debugging system.

The hardware of the Series 37 consists of a small number of field-replaceable units. Sophisticated shift-string capability, such as was provided by the Series 64 DCU, was not appropriate. If a hardware problem develops in the field, there are few enough (and inexpensive enough) field-replaceable units in the Series 37 that a temporary exchange of SPUs (system processing units) can show whether a problem is with intermittent hardware or a design problem. Hardware and microcode design problems should be found and corrected in the factory, not the field, so such tools were deemed unnecessary in the product.

It was decided, therefore, to partition the console functions into functions required in the product and functions required in the factory. Functions required in the product include:

- The ability to LOAD, START, and DUMP the system
- MPE console capabilities
- Remote console capabilities
- Ability to run built-in and external software diagnostics and check the hardware I/O configuration
- Ability to look at and modify software registers and main memory (SoftPanel functions).

Functions not required in the final product, but required for bringing up the system and factory debug include:

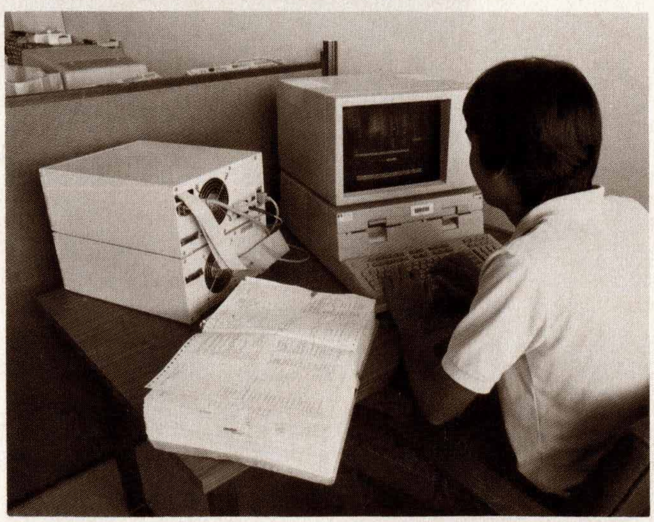


Fig. 1. *An HP 9000 Model 236 Computer serves as the maintenance panel for the HP 3000 Series 37 Computer.*

- The ability to load a microcode image into control store
- The ability to read and modify writable control store
- The ability to read and modify registers used by the microcode
- Breakpoint in control store
- The ability to do I/O commands on the synchronous intermodule bus (SIMB)
- The ability to do 32-bit writes and to read error status information from the memory subsystem.

The microcode and software teams, being the primary initial customers for the maintenance panel, had considerable interest in the specifications for the tool. Their inputs caused us to come up with the following constraints:

- The major CPU registers should be visible on the screen at the same time and should be updated automatically as execution progresses. The contents of the screen should reflect the state of the micromachine when microcode execution is interrupted by the maintenance panel or while microstepping through microcode. Since the CPU contains 15 high-speed registers plus a number of register-file locations that are frequently updated by microcode, it was deemed necessary to use a product with a large enough screen to display all of these registers.
- The microstep function, including screen updating, should occur in about a second or less.
- Full support of the SIMB breakpoint board should be provided, including use of the range and data pattern features (see box, page 20).
- The microcode required for support of the maintenance panel should be small enough to be debugged easily with the microcode simulator tools, so that it could be fully tested before committing it to ROM.
- A means of loading initial microcode and memory images was needed. Since the microcode files were developed on an HP 3000 Series 64, and the cold-load microcode was not to be initially available, a means of transferring such data from the development system to the new hardware had to be developed.

Series 37 Maintenance Panel

The HP 9000 Model 236 Computer (formerly HP 9836) was chosen as the maintenance panel (Fig. 1). It has a large enough screen to display all relevant registers. Its high-performance CPU and BASIC operating system are fast enough that I/O and the screen updates required for a microstep occur in about one half second. It supports HP's LIF (logical interchange format), allowing a limited file transfer capability from the HP 3000 via flexible discs.

It was initially unclear whether interpreted BASIC had sufficient performance to provide a one-second microstep. Early in the development, we wrote an experimental screen update program (not including I/O to the Series 37) and determined that screen update and data formatting times would not be limiting factors. The I/O time to read the required registers over the parallel interface was computed and it was determined that we would be able to meet the specification of a one-second microstep time easily.

The required functions were implemented in a combination of hardware and firmware. The remainder of this article describes the implementation and functionality of the bring-up and debug tools.

Maintenance Panel Hardware

The hardware required for the Series 37 debug panel includes an HP 9000 Model 236 Computer with 768K bytes of memory and an HP 98622A GPIO Interface card (16-bit I/O plus handshake lines). This card resides in the Model 236 card cage. A custom cable connects the Series 37 CPU board to the GPIO card. A connector and drivers for the debug console are standard on every Series 37 CPU board. The optional SIMB breakpoint board is required to set breakpoints in main memory. This board requires a card cage slot in the Series 37 Computer.

The mechanism for entering the maintenance panel code is a process known as "force magic data." This is a means of breaking the microinstruction stream that is executing

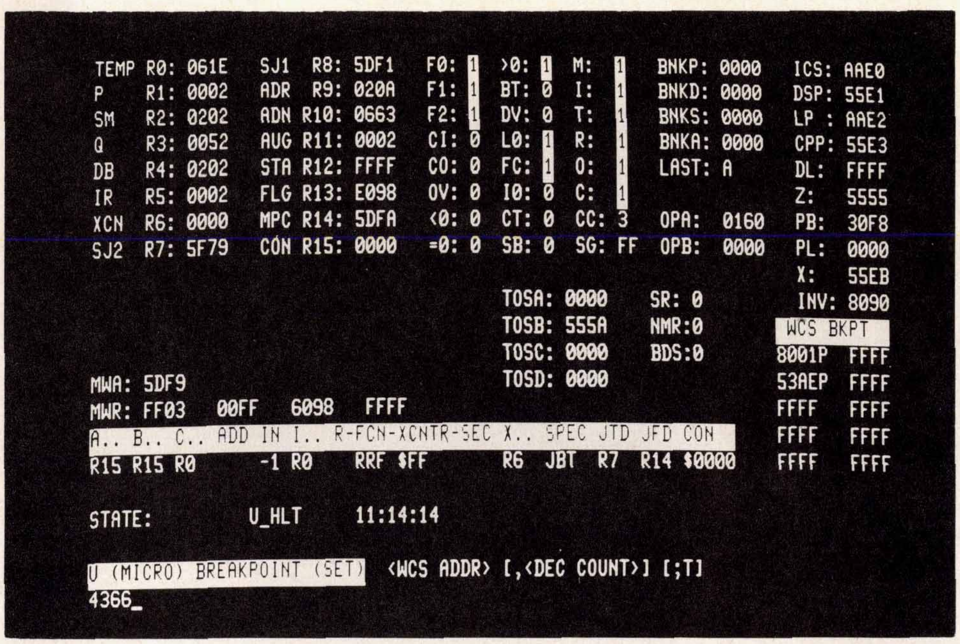


Fig. 2. *Maintenance panel firmware display.*

and passing control to maintenance microcode. The mechanism works by forcing a particular data pattern onto the microinstruction bus, causing a branch to a particular address in the control store. This mechanism is implemented in hardware on the Series 37 CPU board. It should be noted, however, that this mechanism is used for handling microinterrupts other than the maintenance panel, and hence is not unique to the debug panel.

Maintenance Panel Software

The software in the Model 236 Computer consists of a BASIC program, which uses Advanced BASIC constructs to control the displays, process commands, and handle interrupts from the user and the Series 37. The BASIC software is partitioned into user interface portions and I/O portions, allowing us to be flexible in defining new user commands as required during development.

The user normally sees one of two screens, a firmware display or a software display, providing two viewpoints into the machine. The firmware display (Fig. 2) provides a window into the micromachine, including the state of the firmware registers at the last invocation of maintenance mode, along with the currently executing microword in hexadecimal format and disassembled symbolically. The addresses of microbreakpoints currently set are also displayed.

The software display (Fig. 3) shows the HP 3000 programmer's view of the machine (i.e., the state of the macromachine). The display is arranged to show information about the code (PROGRAM) currently running, the stack, and global and status information. A display of the top 24 stack locations is shown at the bottom of the screen. The TOS cache is transparent to the user in this mode; in the firmware display, it is not.

Maintenance Panel Firmware

The work of obtaining and modifying values in memory, registers, and the SIMB is done by a small set of microcode routines. Two sets of these routines exist; one set is in ROM, and another set overlays the ROM routines when the operating system is running.

Operation

During normal system operation, the Model 236 can be connected to the Series 37 CPU board and the impact is minimal. The power-on self-test code checks for a maintenance panel, and causes the panel to become active before attempting to prompt the user for START/LOAD/DUMP commands. Following that, if the system is operated normally, the Series 37 user will not be aware that the panel is connected to the system.

If certain microinterrupts occur, indicating hardware problems, the panel will gain control if it is connected. Alternatively, the user can hit the **HALT** key and force the machine to stop.

While in maintenance mode, the user can issue commands to the Model 236 which are formatted into data patterns and sent out over the GPIO card to effect the various user commands such as Modify Register or Display Memory. These data patterns are interpreted by the firmware, which invokes microsubroutines to make the changes occur in the Series 37 CPU environment. Operations on the registers normally used by microcode are performed on shadow registers while in maintenance mode.

To terminate maintenance mode and resume normal execution, the user presses the CONTINUE(URUN) softkey, which sends a command over the GPIO requesting microcode to restore the state from the shadow registers and resume where execution was left off.

Microbreakpoint

A breakpoint was defined in control store by use of the WCS parity error interrupt. A breakpoint in control store is set by changing the parity bit; the Model 236 keeps a list of locations that have been so modified. The microcode

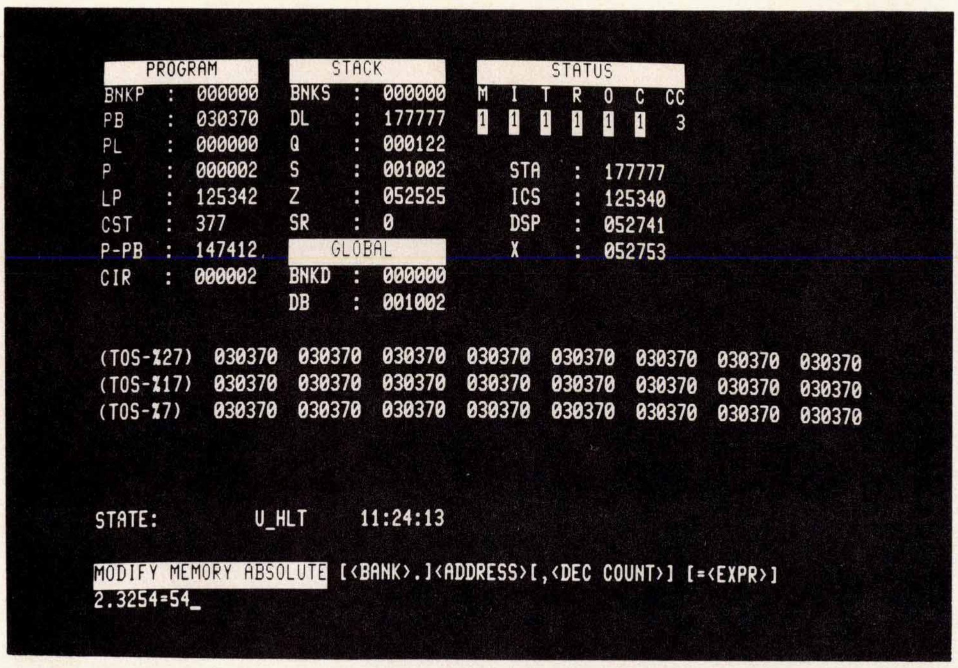


Fig. 3. *Maintenance panel software display.*

The Role of a Programmable Breakpoint Board

In a microcode and software development environment, one of the problems difficult to debug is the case when memory locations appear to contain the wrong data. This normally occurs when a bug in software or microcode causes an illegal write to the suspected memory location. These so-called memory hits may happen at just one memory location, within a range of locations, or over the entire range of the memory. They may happen with a fixed data pattern or in a random pattern. They may happen often or occasionally.

A simple approach to the solution of this problem is to connect a logic analyzer to the CPU bus and program the analyzer to trigger and record the bus transactions when an access to the suspected memory location happens. There are a few drawbacks to this approach. One major drawback is the fact that the logic analyzer just captures what appears on the bus and cannot freeze the CPU for further examination of the internal CPU registers and flags. The other drawback is that this approach is not friendly and requires specific knowledge of the hardware to connect the logic analyzer, a task that most software developers and microcoders try to avoid. A better solution to this problem is to design a board that resides in the computer system like any other board and does the job of a logic analyzer. The advantage of this approach is that no analyzer needs to be connected, and programming is through normal computer instructions.

For the HP 3000 Series 37 Computer, a special breakpoint board was developed to provide memory transaction monitoring capabilities to the system. This board is a programmable I/O channel that is plugged into the system like any other board and monitors memory transactions. Normally, it is transparent to the system and does not interfere with the normal system operation except at system initialization, when it responds to the SIMB ROCL (roll call) instruction to indicate its presence. The programming is through SIMB WIOA (write I/O adapter) commands issued to the board to set its internal registers. Therefore, programming can be done using any facility capable of issuing SIMB commands, including an HP 9000 Model 236 Computer or SoftPanel's microcode.

The breakpoint board consists of nine write-only and three read-only registers. The write-only registers include two bank and address registers, two data registers, two opcode registers, and one control register. The board is configured by writing into the control register. The bank and address registers can be programmed to break on individual addresses or on any address between two limits (RANGE mode). The data registers can be programmed to break on patterns of 1, 0, or X (don't care) bits. The read-only registers capture the FROM CODE, bank, and address of the breakpoint event.

Once the board is programmed, it compares every memory transaction that appears on the SIMB with its internal registers. When a match occurs it sends a nonmaskable interrupt (SIMB WARMSTART) to the CPU. Upon receiving this interrupt the CPU virtually freezes and executes special microcode that permits the examination of the CPU internal registers and flags by SoftPanel or the maintenance panel's Model 236 Computer. The SIMB information that caused the break (memory address, SIMB FROM CODE, and SIMB OPCODE of the transaction) are all captured in the internal registers of the breakpoint board and can be accessed by issuing the SIMB RIOA (read I/O adapter) command to the breakpoint board.

The board was immediately put to use when it became available and solved many microcode bugs in its first weeks of existence. After a few weeks, users began to develop some unconventional uses for the board. An interesting unconventional application is to use the board as a smart single-step facility for the MPE operating system to single-step over PCAL instructions, interrupts, etc. In this mode of operation, the MPE system Debug facility is used to freeze the segment in question to prevent the operating system from moving the segment. Then the breakpoint board is programmed in RANGE mode such that the two breakpoint registers point to the start and end of the procedure or segment in question. The CONTINUE key of the Model 236 Computer can then be used to single-step over the segment.

One dramatic case of breakpoint board application occurred when the microcoders had been chasing a problem for several days and had totally forgotten about the already programmed breakpoint board inside the system. Several days later, while they were working on another problem, the Model 236 maintenance panel beeped, indicating a break from the breakpoint board. From there it was a matter of minutes to find the bug they had been looking for for days.

Mehraban Jam
Development Engineer
Computer Systems Division

for handling WCS parity interrupts includes going through the "force magic data" process and subsequently checking for the presence of the Model 236. When the Model 236 sees an interrupt from a WCS parity error, it checks to see if the location is one it knows to be a breakpoint. If so, it treats the interrupt as a breakpoint, and indicates this to the user. If not, it treats the interrupt as a valid WCS parity error, and indicates this to the user.

With this general process, it was possible to create in BASIC an arbitrarily large breakpoint table (we had eleven entries: one special-purpose and ten general-purpose). It was also possible to create temporary, permanent, and counting breakpoints by changing only BASIC code. With the programmable BEEP statement, an ALARM function is provided: a short beep for a hit on a counting breakpoint (different tones for different breakpoints), and a warble for a complete step on expiration of a breakpoint.

The special-purpose breakpoint is used to implement the single-step function for the software display mode. There is an overhead line that always occurs between machine instructions in the main microcode. When a single-step at the macromachine level is requested, the CPU is allowed to free-run until this line is executed between instructions.

Memory Breakpoint

Memory breakpoint capability requires that an additional hardware module, the SIMB breakpoint board, be installed in the Series 37 card cage. I/O is done to this board by the maintenance panel by general-purpose I/O routines that talk to the SIMB. The user interface code formats these calls to the I/O portion of the BASIC code, which in turn does I/O over the GPIO board, which causes microcode routines to write and read the various registers on the breakpoint board. The memory breakpoint is discussed further in the box above.

SoftPanel: Virtual Software Debugging Panel

For software debugging on the HP 3000 Series 37, a special software debug facility, a virtual software debugging panel called SoftPanel, is implemented in microcode. SoftPanel is not the only software debugging tool available in the HP 3000 system. The MPE operating system has its help facility (not the same as the HELP command) and there is the Debug facility. However, there are times when these facilities may be hard to activate and/or cannot run at all. SoftPanel is the only debugger on the Series 37 guaranteed to be there when needed. It can always execute.

SoftPanel is a fundamental debugging tool. Through the use of its commands, the user can gain low-level information about the system state. This information is vital when one is trying to determine the cause of many failures. Without SoftPanel, the user has a significantly lower chance of determining what is going on.

Basic Characteristics

The Series 37, like other HP 3000s, is a microcoded machine. Thus it is really two machines in one. The macromachine executes the instructions of the HP 3000. The micromachine implements the macromachine. SoftPanel is designed to debug software at the level of the macromachine. Other tools exist for dealing with the system at the micromachine level, primarily the HP 9000 Model 236 maintenance panel described above.

HP field engineers, being the primary customers for SoftPanel, had considerable influence on its implementation.

A major requisite for a debugger is machine state visibility. SoftPanel provides this easily because of its microcode implementation. Because it is implemented as microcode and executes directly on the target machine, visibility of different machine states is there for the asking. This includes memory, macromachine state, direct I/O, and some micromachine control cells that directly affect the macromachine.

Another major requisite is transparency. The user should be able to invoke the debugger, view the desired machine state, and return to the software that was interrupted with out any undesired effects. This, of course, assumes that the machine state was not modified by the user. SoftPanel has knowledge of what determines a macromachine state and carefully preserves this information. This includes such things as I/O system state, memory state, macromachine state, and interrupt system state. None of this information is altered unless specifically requested by the user. At any time, the user can tell SoftPanel to return to the software that was interrupted.

Yet another requisite is remote access. SoftPanel is usable from a remote diagnosis center. As the number of systems in the field increases, it becomes economically unfeasible to service our machines any other way. SoftPanel accomplishes this by making use of the Series 37's remote operator interface. Anything that works on the local console will also work through this interface.

A last major requisite is ease of use. To accomplish this, the SoftPanel syntax is closely modeled after Debug's. Debug is a very well-known (in the HP 3000 user community) debugger for the HP 3000. By choosing this syntax, we avoided a large portion of the learning curve for many users.

SoftPanel Structure

To facilitate quick design and implementation, SoftPanel is partitioned into four major sections: command recognizer, command parsers, command executors, and special

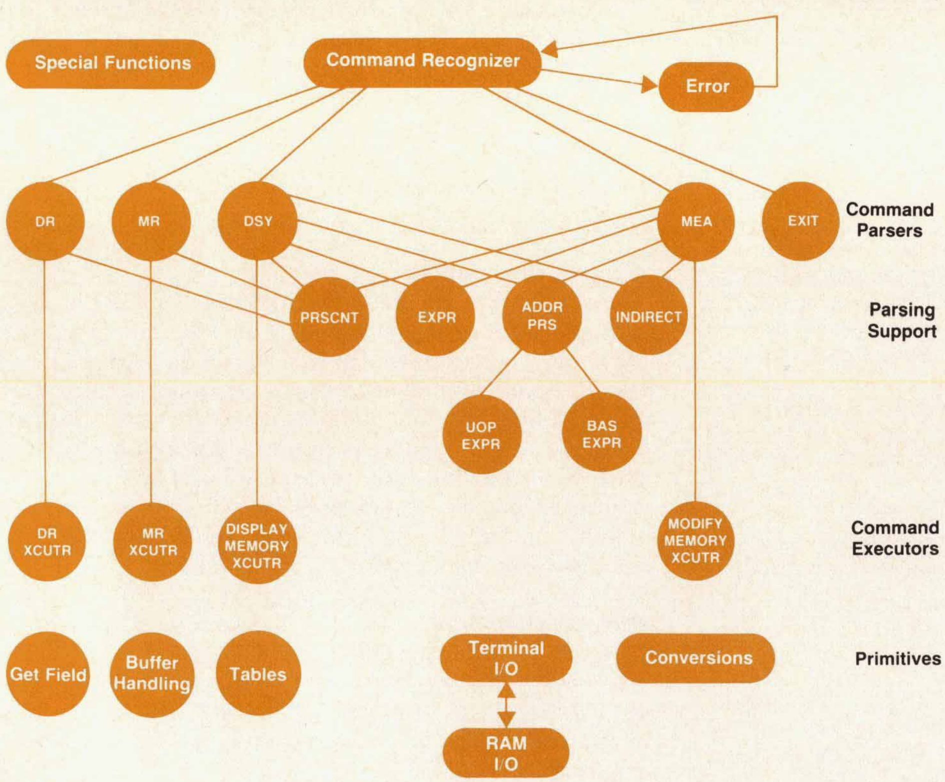


Fig. 4. *The structure of SoftPanel, a microcoded virtual software debugging panel for the Series 37.*

functions. Fig. 4 details the structure graphically.

The command recognizer is responsible for accepting user input and determining which command should be given control. Included in this activity are prompting, reading the user response, scanning the first field, looking for a valid command, and activating the appropriate command parser. If an invalid command is input, an appropriate error message is displayed.

The second major section of SoftPanel consists of the command parsers. This software is responsible for scanning the rest of the command line for the command's parameters. There is one parser for each command. If valid parameters are found, a module in the third major part, the executors, is activated.

The command executors actually implement the command. Each executor is passed a set number of parameters by the corresponding command parser. It performs its action and returns to the command recognizer. What each executor does is entirely dependent upon the command. Some commands simply take a starting memory address and display consecutive locations. Other commands display some information and wait for user input. Depending upon this input, some portion of the machine state may be altered.

The last major section deals with special functions. Included are memory breakpoints, and the interface to the macromachine.

Underlying these major sections are a significant number of subroutines. There are subroutines that directly support the command recognizer, parsers, and executors, and there are primitive subroutines that support base functionality on all levels of the structure. Included are the following:

- Field scanner. This routine recognizes the next field in the input buffer, determines its type (ASCII string, null, or numeric) and returns this information.
- Conversion routines. These routines convert between internal (binary) and the appropriate external form. This external form is either the octal or the hexadecimal ASCII equivalent as determined by the current radix. This is under user control.
- Expression handling. This family of routines deals with processing expressions of various forms. Each one starts scanning from the current place in the input buffer and terminates appropriately. The value of the expression is returned to the caller. These expression handlers can deal with addition, subtraction, multiplication, indirection, unary operators, base address modification, and counts.
- Primitives. These provide buffer manipulation, table management, address handlers, basic terminal I/O, and connections back to the command recognizer for command termination and/or error conditions.

SoftPanel Commands

SoftPanel is command-driven. A prompt (SP>) is displayed, the user types the desired command followed by its parameters, the command is processed and executed, and the cycle is repeated. All activity of SoftPanel is in some way caused by the input of a command. This includes the special functions (memory breakpoint processing) as well as regular functions.

There are six major groupings of commands: display memory, modify memory, input/output, miscellaneous, code breakpoints, and memory breakpoints. The command syntax is strongly dependent upon the architecture of the HP 3000, with its segmented memory system and separation of code and data spaces.

The command-driven approach was chosen for two reasons. The first was ease of implementation, and the second reason is related to the requirement for remote access. This access is primarily through 1200-baud dial-up lines. A full screen debugger becomes extremely obnoxious running at this speed.

Memory Breakpoints

One very nice feature of SoftPanel is its ability to deal with the Series 37 memory breakpoint board (see box, page 20). This is a hardware tool that allows debugging of extremely difficult problems. One such problem is the trashing of particular memory locations that then cause a system crash (bank 0 is a particularly nasty place to trash). The memory breakpoint board allows the user to trap writes and reads to particular addresses or ranges of addresses.

SoftPanel supplies an interface for setting this board up and handling the results when a trap occurs.

Acknowledgments

Harish Joshi implemented the maintenance microcode, including the microinterrupt handler, and wrote the low-level I/O routines in Model 236 BASIC. Paul Rogers implemented the special circuitry on the CPU board that was required for the maintenance panel.

Virtual Microcode Memory

The HP 3000 Series 37 self-test ROMs use an unusual implementation of a virtual microcode space in the four 128K-bit ROMs that enable the 64-bit CPU to access and run self-test functions, boot routines, and diagnostics, simulating a separate control processor. This separate control processor, if present, would use writable control store (WCS). Since the separate processor is not present, diagnostics, boot code, and other microcode can be loaded and run without taking valuable WCS space, which is also needed for the main instruction microcode. This extra microcode is loaded into WCS only when its functions are needed. It is present in ROM on the CPU board and does not need to be loaded from disc.

This virtual microcode space is implemented by a routine called LoadNxt, which enables a specified program or data file to be loaded from ROM into any arbitrary area of WCS so that it can be executed or accessed.

This solution minimizes the use of scarce WCS resources, which must be shared with HP 3000 instruction set microcode, without sacrificing self-test, boot, or diagnostic functionality.

Acknowledgment

Norm Galassi wrote LoadNxt.

Chris Shaker
Development Engineer
Computer Systems Division

New Cardiograph Family with ECG Analysis Capability

These three new HP cardiographs, in addition to recording traditional ECG waveforms, can perform differing levels of measurements and analysis to aid diagnosis of heart behavior.

by Robert H. Banta, Jr., Peter H. Dorward, and Steven A. Scampini

DURING THE LATTER HALF of the 19th century, it was discovered that the contractions of heart muscles generated electrical signals that could be detected on the surface of the body. In 1903, a Dutch physiologist, Willem Einthoven, was the first to record these signals accurately. This was done by placing the subject's limbs in buckets of saline solution that were connected to a string galvanometer (Fig. 1). For his efforts, Einthoven was awarded the Nobel Prize in physiology and medicine in 1924. Over the years, a body of knowledge has developed by which a physician can deduce the condition of the heart from these electrical signals. By examining the shape, or morphology, and timing, or rhythm, of these electrical waveforms, a physician can determine the condition of the heart muscles and the conduction mechanism used to trigger muscle contractions.

A variety of instruments for recording these waveforms has been developed and the electrocardiogram (ECG) continues to play an important role in the diagnosis of heart disease. It is estimated that 200,000,000 ECGs are taken annually throughout the world. Because of its simplicity, noninvasive nature, and widespread acceptance, the electrocardiogram is used as the first level of screening in the detection of heart disease. Further screening involves increased expense and/or risk to the patient, requiring such methods as stress testing, ultrasound scanning, ambulatory (Holter) monitoring, catheterization, and surgery.

In the 1960s, computers began to find use aiding the physician in the diagnosis of the electrocardiogram. Today these analysis programs are attaining widespread acceptance as a means to help contain the cost of health care. These computer systems are expensive, however, and can only be justified where large volumes of ECGs are processed.

Hewlett-Packard's new HP 4760 Cardiograph family (Fig. 2) combines the analysis capabilities of HP's Model 5600C ECG Management System with the technology of the HP 4700A PageWriter Cardiograph introduced in 1981.[1] The result is a compact and inexpensive tool to aid the physician in the diagnosis of heart disease.

Role as a Diagnostic Tool

This new family of cardiographs consists of three members: the HP 4760A, the HP 4760AM, and the HP 4760AI. All three provide ECG waveforms along with patient ID information (name, sex, age, weight, etc.) in a clear manner for quick review by the physician (Fig. 3). The information contained on the report can be edited using the instrument's alphanumeric keyboard and liquid-crystal display. The ECG waveforms can then be stored for later retrieval, or transmitted over phone lines to another cardiograph or

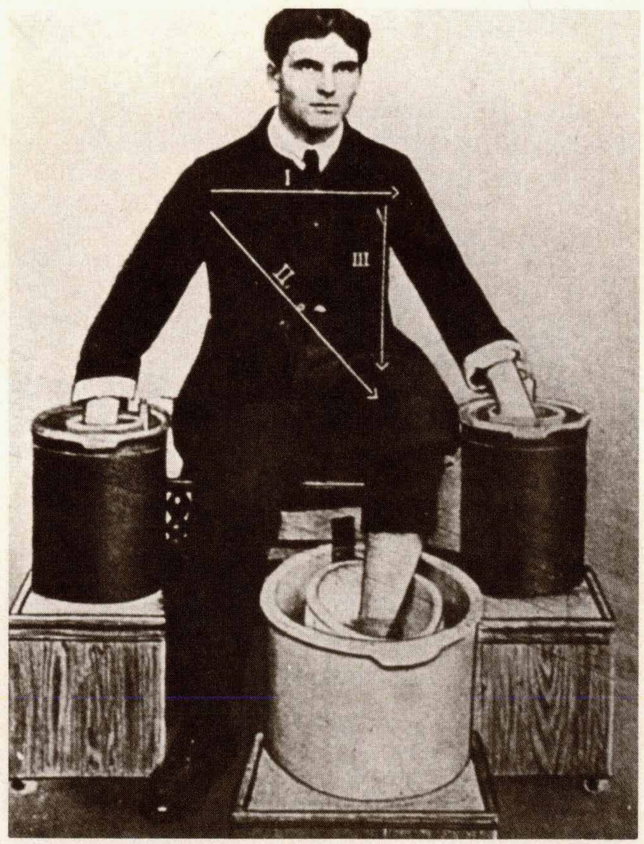


Fig. 1. *Early ECG measurements required putting the subject's limbs into buckets of saline solution to make sufficient electrical contact for detecting the electrical activity of the heart muscles on a string galvanometer (source of picture unknown). The Roman numerals indicate three of the commonly recorded levels. They are derived by taking voltage differentials referenced to the right leg. That is: I = left arm − right arm, II = left leg − right arm, and III = left leg − left arm.*

ECG Storage and Transmission

In the past, an ECG was usually transmitted to a remote ECG analysis system as it was acquired. Poor signal quality, which could come from the original signal or be induced during transmission by phone line noise, could not be checked until afterwards. If retransmission was required, it was necessary to go through the entire process again.

One solution added an optional analog tape recorder to the cardiograph cart to provide local storage of ECGs. This provided the flexibility of storing the ECGs as they were acquired, and transmission of them as a batch afterwards, but the option was costly, bulky, and difficult to manage.

To provide flexibility and reduce errors in the acquisition and transmission of ECGs, the HP 4760A and its predecessor, the HP 4750A, incorporate solid-state ECG storage and digital transmission facilities. Local ECG storage is realized by including a 256K-byte, battery-backed, nonvolatile CMOS RAM capable of holding data for up to 40 ECGs. Transmission flexibility and integrity are managed by a custom digital transmission protocol developed for the HP 4750A and HP 4760A.

ECG storage space and telephone connect time are minimized by the use of a first-difference encoding algorithm. This compression algorithm is based on the normal distribution of the first differences of successive points of an ECG. Since most of the signal consists of small excursions about the baseline, the original 12-bit data can be greatly compressed by encoding small first differences into three- and five-bit codes. The algorithm is non-distorting, so the original data is faithfully recreated upon decompression with no loss of accuracy.

An important part of the transmission protocol is its error detecting and handling capabilities, since the compression algorithm is extremely sensitive to error. The lower level of the protocol provides blocking, line arbitration, and link integrity besides CRC-CCITT-based error detection. Data blocks are accompanied by CRC (cyclic redundancy check) codes, and any detected errors result in retransmission of data. If a block cannot be transmitted without error after several attempts have been made, the link is terminated.

The upper level of the protocol supports an expandable command interpreter that allows the transmission of ECGs, analysis reports, and ECG measurement information from cart to system, cart to cart, and system to cart. For example, by accessing the system data base through a local cardiograph, a physician can review a patient's previous ECGs while at the patient's bedside.

Charles C. Monroe
Project Manager
McMinnville Division

to a central computer system (see box above).

HP 4760AM. The HP 4760AM Cardiograph also performs a series of measurements on the ECG waveforms. These measurements fall into two major categories: morphology and rhythm. (For details about these measurements, see article on page 29.) In most cases, the HP 4760AM is able to perform these measurements more quickly and accurately than a human. This eliminates much of the drudgery of ECG analysis, allowing the physician to concentrate more time on patient care.

HP 4760AI. The physician's job is made even easier by the HP 4760AI Cardiograph, which applies a series of criteria

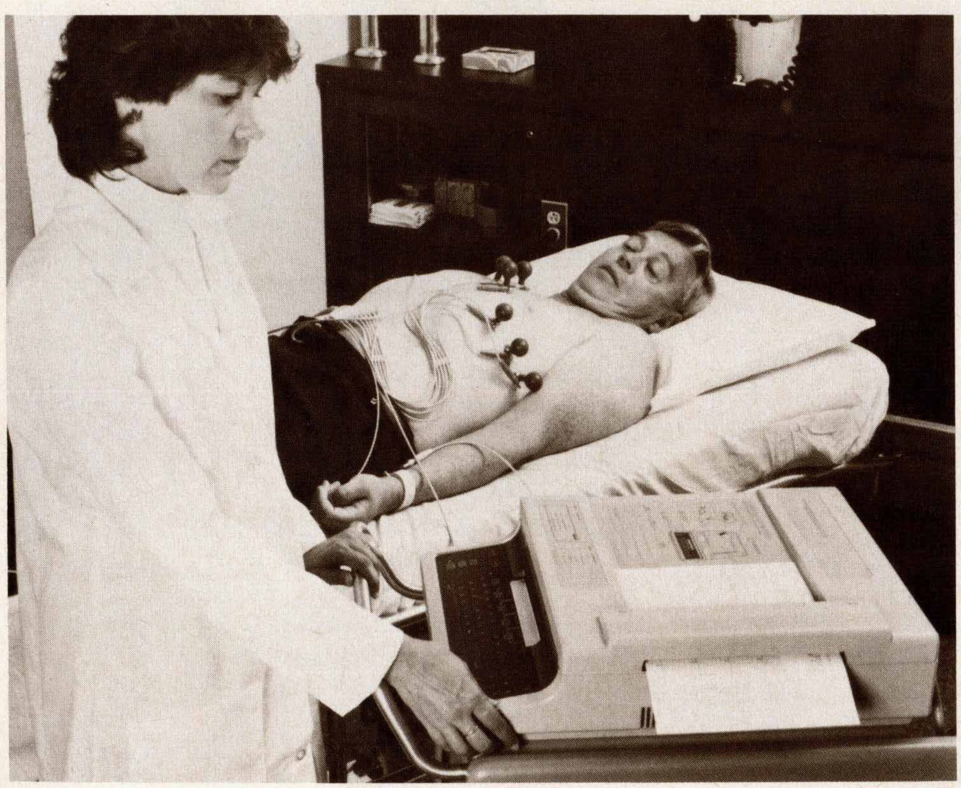


Fig. 2. *The HP 4760A family of cardiographs features the plotting technology of the earlier HP 4700A PageWriter Cardiograph with stand-alone computer analysis techniques based on HP's Model 5600C ECG Management System. Data can also be transmitted to a central system for storage and later review.*

(based on rules similar to those learned by the physician) to the ECG measurements, yielding a preliminary interpretation of the ECG (Fig. 4). Either an adult or a pediatric criteria set can be used depending on the patient's age. In addition to reducing the physician's work load, interpretive cardiographs are being used to generate preliminary interpretations in situations where physicians skilled in ECG diagnosis are sometimes not immediately available.

The interpretation produced by the HP 4760AI has been found to be diagnostically correct better than 90% of the time.[2] Because the analysis may not be always correct, the physician must continue to have the final say as to the actual condition of the heart and the resultant care of the patient.

Design Philosophy

The primary challenge in the development of the HP 4760 family was providing the analysis capabilities of ECG computer systems in a stand-alone instrument at an affordable cost. The existing technology of HP's PageWriter Cardiograph family seemed an appropriate base from which to begin since it contains the necessary cardiograph functions at a low price. It also offers an X-Y recording mechanism, unlike the traditional galvanometer writing system used in many other cardiographs. This recording mechanism allows the flexibility to generate ECG traces and interpretive reports in an easily handled format.

Very early in the design of the HP 4760A, the decision was made to use an exact copy of the analysis program used in the HP 5600C ECG Management System. This greatly reduced the development time, but more important, yielded an analysis program that was already clinically proven and well accepted. The major obstacle to this effort was that the HP 5600C ECG Management System is based on an HP 1000 Computer, while the HP 4760A Cardiograph is based on a 68000 microprocessor.

A multistage plan was developed to effect the program's translation. At each stage, the accuracy of the translation was verified by analyzing ECGs from a common data base and comparing the results to those produced on the HP 5600C. Some minor differences were found in intermediate results, which were caused by floating-point precision differences between the two processors. However, none of the ECG interpretations showed any differences in the final diagnosis.

Another design goal of the HP 4760 was to provide an easy upgrade path. A port is provided for a plug-in module, installable by the user, that can contain application programs or criteria sets for the analysis program (Fig. 5). Such a module can be used to upgrade an HP 4760AM to provide full interpretation. A criteria set in a plug-in module could supplement those already in an HP 4760AI, or another module could be used to replace a standard criteria set with an adjusted version. An application module is more open-ended, because it can modify the standard firmware or add a new capability. Application modules currently available include:

- HP 47611A Adult Criteria (Version 06). This module can be added to existing HP 4760AM Cardiographs to provide adult ECG analysis capability.
- HP 47612A Pediatric Criteria (Version P2). This module adds pediatric ECG analysis capability to HP 4760AM and HP 4760AI Cardiographs.
- HP 47619A ECG Collection (I). This module performs a very basic check of the analysis operation and contains eight stored ECGs of varying diagnoses. This data can

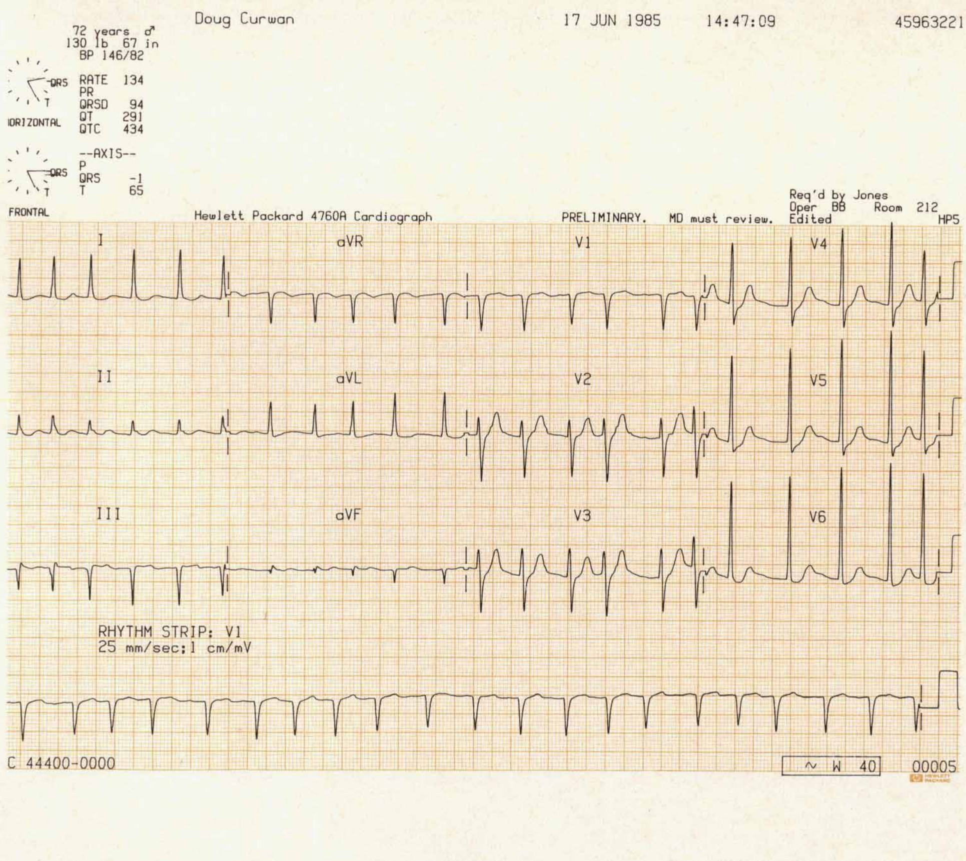


Fig. 3. *Example of typical ECG record (48% of actual size) taken with an HP 4760A Cardiograph. Patient information and physician's comments can be easily recorded with the waveforms in a format easily filed for future reference.*

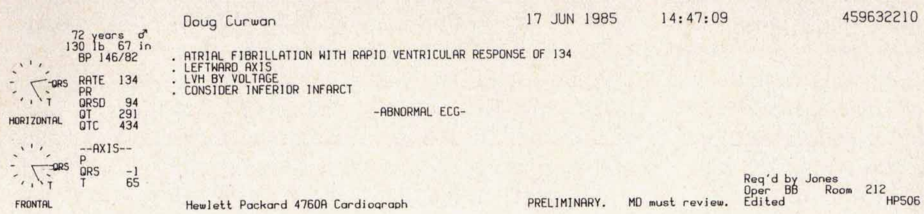


Fig. 4. *The HP 4760AI Cardiograph can perform a preliminary interpretation of the ECG data based on clinically established data to aid the physician. Shown is the interpretation output for the ECG record shown in Fig. 3.*

be used in learning or demonstrating the cardiograph, and eliminates the need for a test subject or patient simulator.

The HP 4760AI normally houses 576K bytes of ROM, 320K bytes of which contains the analysis program. Fully configured, the basic HP 4760A can support 1.7 Mbytes of ROM and RAM. To protect an investment in masked (custom) ROMs, and to provide fixed addresses for application programs, all module calls go through jump tables located in a pair of EPROMs (electrically programmable read-only memories). If it is necessary to change a software module contained in a masked ROM, the updated version is placed into an EPROM and the jump table is adjusted to point to the new version.

Operator Interface

An effective electrocardiograph has to be easy to use. One aspect is the design of the control panel. It was felt that a properly designed control panel would go a long way in making the operation of the HP 4760A self-explanatory. Another aspect is the provision for acquiring high-quality, noise-free recordings. When a list is made of the functions and parameters that can be selected on the HP 4760A, it becomes apparent that the operator interface had to be well thought out to avoid being overwhelmingly cryptic.

After considerable discussion, several strategies emerged to structure the design. First and foremost, the HP 4760A should be easy to use as a basic electrocardiograph so that in a possibly confused clinical setting, relatively untrained personnel can easily obtain a diagnostic electrocardiogram. Second, the values of the parameters that are subject to frequent modification should be visible and easily changed. Third, the functions that the HP 4760A performs should be grouped into two broad categories: basic electrocardiograph and data management. Under data management come the functions of storage, transmission, and editing. Fourth, many parameters can customize the HP 4760A to a particular setting and should be protected; these are typically set on receipt of the machine and are never changed thereafter.

Using these basic guidelines, a design emerged that makes extensive use of softkeys, using the HP 4760A's liquid-crystal display for labeling (Fig. 6). Dedicated keys are carefully limited to only the most frequently used functions, because it was felt that a vast array of keys would intimidate an operator. The customizing parameters are hidden behind an unlocking sequence of key presses so that they are not wandered into easily during day-to-day operation.

Careful attention was paid to the color scheme used on the keyboard. Keys are prioritized by brightness of color and grouped by commonality of color. The less frequently used alphanumeric typing keys are assigned low color priority so as not to overwhelm the operator looking for the **START** button.

The result is an interpretive cardiograph with data storage, transmission, editing, administrative report generation, and self-testing capabilities that are straightforward to use, yet highly flexible.

Signal Processing

The ECG signal is subject to contamination by noise from a variety of sources. Three important noise categories are ac power line interference (50 or 60 Hz), baseline wander, and muscle artifact.

Power line interference appears as a 50/60-Hz "buzz" on the electrocardiogram. Two primary sources are the electrostatic fields generated by power wiring and the magnetic fields generated by devices such as transformers and motors. The electrostatic fields produce a common mode voltage on the patient relative to ground. This is typically not a problem; high common mode rejection is relatively easy to achieve. Magnetic fields, however, induce differential signals in the loops formed by the lead connections to the body. Unfortunately, these 50/60-Hz signals are well within the passband (0.05 to 100 Hz) of a diagnostic electrocardiograph.

A notch filter is a practical solution if its stop band can be made wide enough to allow for normal ac power line frequency variations, yet narrow enough to avoid visible distortion of the ECG signal. Attempts to implement notch filters as conventional analog filters requires relatively wide bandwidths to accommodate component drift. Such

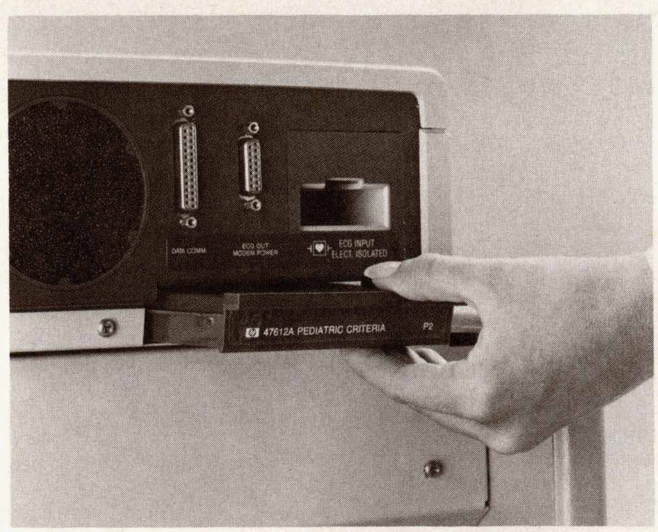


Fig. 5. *A small slot in the back of every HP 4760A is designed to accept any of the existing or future application modules.*

Artifact Indication

Acquiring a noise-free electrocardiogram requires careful technique. Possibly the most difficult source of noise to control is muscle artifact, since it is largely determined by the degree to which the patient is relaxed and comfortable. In the past, the operator of a cardiograph could not predict how much muscle artifact would contaminate the ECG waveforms. Obtaining a "clean" record might require several tries and consume multiple sheets of paper. The artifact indicator on the HP 4760A Cardiographs addresses this problem by providing a preview of the level of artifact before an ECG recording is started.

Muscle artifact appears on the ECG waveform as a relatively low-amplitude, relatively high-frequency signal with random amplitude and frequency variations (see Fig. 1). Most of the energy in the ECG signal lies below 35 Hz; much of the muscle artifact energy lies above 35 Hz. This suggests a simple implementation for detection of artifact: a high-pass filter with a corner frequency at 35 Hz, followed by a detector, a low-pass filter, and a display. However, it turns out that the R-wave excursions generated by the depolarization of the ventricles contain significant energy above 35 Hz. The simple system described above adequately displays muscle artifact, but is contaminated by the R-wave energy passing through the filter. This appears as a periodic heartbeat deflection.

This problem is solved by replacing the low-pass filter with a section consisting of a comparator and an integrator (Fig. 2). The rate of change of the output of this section is limited by the time constant of the integrator stage and the maximum output swing of the comparator. These two parameters can be selected

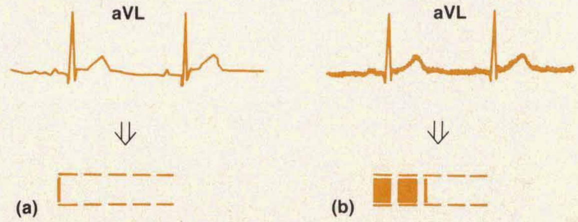


Fig. 1. *ECG waveforms and HP 4760A artifact indicator display for signals (a) without and (b) with significant muscle artifact.*

such that the relatively slow changes in the amplitude envelope of the muscle artifact pass through the filter while the brief R-wave pulses are attenuated and produce minimal deflection.

To enhance the usefulness of the artifact indicator display further, a logarithmic compression stage is placed after the output low-pass filter. The increased dynamic range of the display provides feedback in situations where initially the artifact level may be very high. To a provide a simple bar display for the three channels being previewed, the channels are continuously monitored and the channel with the highest level of artifact is displayed. The cardiograph's filter and gain settings affect the signals sent to the artifact display process. Thus, one can preview the effect of filter and gain changes before a record is taken. The result is a feature implemented in software without additional hardware costs that makes low-noise ECG recording easier and more economical.

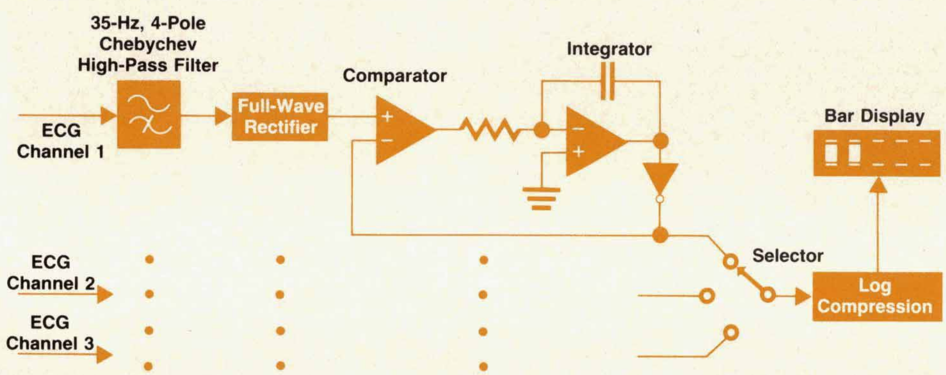


Fig. 2. *Signal processing block diagram of the HP 4760A's artifact indicator.*

filters have a tendency to ring following sharp transitions in the ECG waveform. Digitally implemented filters do not drift, allowing for very narrow notch filters. In the HP 4760A, the ac power line notch filter is implemented in software running on the internal 68000 microprocessor. The final notch width was determined by studying the effect of the filter on both actual and simulated ECGs. The final filter design very effectively removes power line interference and can tolerate expected frequency variations, yet does not visibly distort the electrocardiogram.

Baseline wander is the term used to describe low-frequency components that appear in the electrocardiogram but are not part of the ECG signal. These components typically are the result of patient respiration or slow electrochemical phenomena associated with the electrode-electrolyte-skin interfaces. Although high-pass filtering will remove these components from the signal, the problem has been that conventional analog filters also introduce phase distortion, which can seriously distort certain important features in the ECG waveform.

The high-pass filter's cutoff frequency can be increased significantly above the 0.05 Hz value recommended by the various standards committees without significant degradation of the electrocardiogram if no phase distortion is introduced. Here again, digital filter techniques make the desired response practical to implement. It is feasible by the use of digital memory, in essence, to "reverse" time and "undo" phase distortion introduced by a filter section. Such a zero-phase filter (Fig. 7) is implemented in software on the 68000 microprocessor. It reduces baseline wander without significantly distorting the ECG waveform.

The electrical signals generated by the activity of the skeletal muscles also interfere with the ECG signal. These signals are referred to as muscle artifact. They are similar

Fig. 6. *Operating panel of the HP 4760A family of cardiographs.*

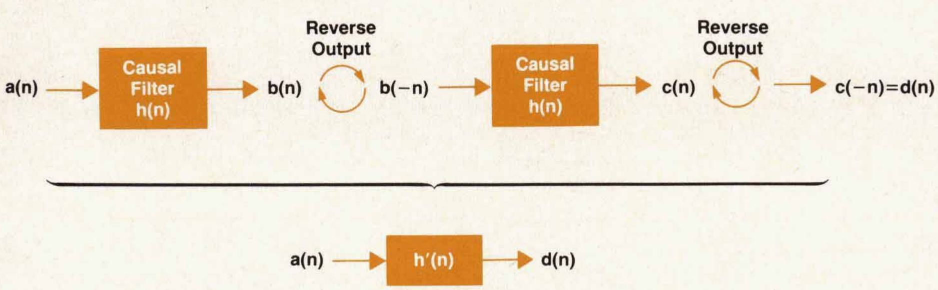
in amplitude and frequency to many of the small, high-frequency components of the ECG. Although a simple low-pass filter reduces muscle artifact, it also attenuates desirable high-frequency features. Alternatively, muscle artifact can be reduced by having the patient relax. However, the patient's state of relaxation may not be apparent to the cardiograph operator before a recording is started. The artifact indicator on the HP 4760A provides an indication of the level of artifact being generated by the patient. It allows feedback from the operator to the patient; the patient can be made more comfortable or reassured before the ECG is committed to paper (see box on page 27).

Acknowledgments

The team responsible for the cardiograph design included Doug Brown, John Goodnow, Bob Graves, Charlie Monroe, Toby Olsen, Dick Regan, and Greg Vogel, in addition to the authors. The translation of the analysis program was accomplished by Siobhan Charlesworth, Bob Cohn, John Doue, and Dave Sturges. The packaging for the HP 4760A and its cart, the HP 4721A, were designed by the industrial design team of Ray Jedrey, Ted Minor, and Pete Rhoads. Jeff Corliss was a major contributor to the definition of the HP 4760A, while concurrent manufacturing engineering was ably performed by Jack Lazzaro. The concerted effort of these individuals, and many others in the Andover Division, made the development of the HP 4760A family of cardiographs possible.

References

1. P.H. Dorward, et al, "New Plotting Technology Leads to a New Kind of Cardiograph," *Hewlett-Packard Journal*, Vol. 32, no. 10, October 1981.
2. M. Hodges, "A Clinical Evaluation of the HP ECG Analysis Program: Program Accuracy and Value of Adjustable Criteria," *Computers in Cardiology*, September 1979.



$h'(n)=h(n)*h(-n)$ Taking the Fourier transform, $F[h(n)]=H(e^{j\omega})$ and $F[h(-n)]=H^*(e^{j\omega})$

$$\therefore F[h'(n)]=H(e^{j\omega})*H^*(e^{j\omega})=H'(e^{j\omega})$$
$$|H'(e^{j\omega})|=|H(e^{j\omega})|^2$$
$$\angle H'(e^{j\omega})=0$$

Fig. 7. *Block diagram of zero-phase, high-pass filter.*

Computer-Aided ECG Analysis

by John C. Doue and Anthony G. Vallance

COMPUTER-AIDED ECG interpretation got under way about 25 years ago with the simultaneous development of two separate computer programs—the Pipberger program[1] and the U.S. Public Health Service's ECAN program.[2] In those early days when the programs' performance was, at best, "fair," and the ECG computer system's capabilities were limited, the concept of computerized ECG interpretation/management had yet to be proven. Today, the major ECG analysis programs are considered good, and close to 20% of the ECGs taken in the U.S.A. have a computer-assisted interpretation. The concept is now accepted and widespread use is a matter of continuing performance improvements and price reduction.

Hewlett-Packard entered the computerized ECG analysis field in 1968 by obtaining both the Pipberger and ECAN programs for incorporation into an ECG management computer system—the Model 5600C. These analysis programs went through about seven years of evolutionary improvements before they were replaced by a totally new HP analysis program in 1978. The new HP program was readily accepted by the market and new features such as ECG Criteria Language (ECL) and user-definable medical criteria have increased the general market acceptance of computerized ECG interpretation.[3] Since its introduction, the program has gone through several revisions in both its pattern recognition algorithms and its medical knowledge base. (For more discussions about the program, see references 4 and 5.)

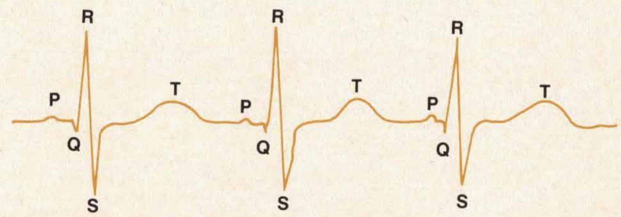


Fig. 1. *Typical ECG waveform for three consecutive heartbeats. The analysis program's assignment is to locate each of the PQRST complexes correctly, and to measure the height, the width, and a multitude of other parameters as depicted in Fig. 2.*

ECG Waveform

What is an ECG waveform, and what do we need to measure for analysis?

The ECG waveform (Fig. 1) has several parts that depict the action of the heart cycle. The principal parts are the P, QRS, and T waveforms, which together are called a complex. Irregularities in the shape (morphology) of the PQRST complex indicate heart muscle abnormalities. Irregularities in the timing of the waveforms (rhythm), either within one complex or between several complexes, indicate nerve conduction abnormalities. Therefore, the shape and regularity of the ECG waveform are both necessary in making a proper diagnosis of the condition of the heart. Normal variations and medically important ECG conditions make the practice of diagnostic ECG interpretation a complex and hard-learned profession.

The analysis program's assignment is to locate each of the PQRST complexes correctly and to measure the height, the width, and a multitude of other parameters as depicted in Fig. 2.

Program Structure

The HP ECG analysis program is composed of several

(continued on page 31)

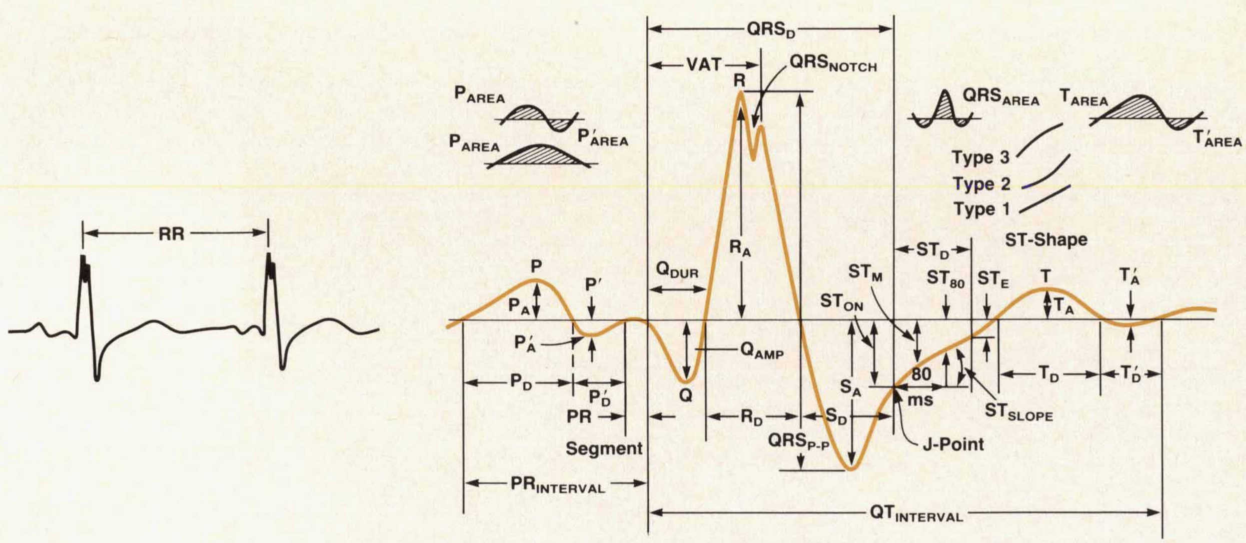


Fig. 2. *Measured ECG parameters for ECG shown in Fig. 1.*

ECG Criteria Language

Hewlett-Packard has developed a medically oriented computer language for the definition of electrocardiographic criteria. Called ECG Criteria Language, or ECL, the primary objective is to allow criteria definition in familiar electrocardiographic terms by physicians with no knowledge of computer programming. Basically, it provides a mechanism through which ECG criteria may be expressed in a form readable by both a cardiologist and a computer. By referring to electrocardiography textbooks and by communicating with a wide cross section of users, language constructs that are consistently used to describe ECG criteria have been chosen to form the foundation of ECL.

Categories

At the highest level, criteria expressed in ECL are broken into medically significant categories that are analogous to chapters of an electrocardiography textbook. Category headings take the form:

```
CATEGORY RVH   "RIGHT VENTRICULAR HYPERTROPHY";
                     •
                     •
CATEGORY LAE   "LEFT ATRIAL ENLARGEMENT";
                     •
                     •
CATEGORY LVH   "LEFT VENTRICULAR HYPERTROPHY";
```

Sentences

Within each category is a series of sentences in which the actual criteria are expressed. These sentences allow the program to PRINT a diagnostic statement when the criteria are met, to SUPPRESS a statement in the presence of a higher-priority diagnosis, GOTO another point in the program, or perform calculations and SET the result to a special variable for use later in the program. The PRINT sentence has the form:

```
PRINT   <diagnostic statement> IF <medical criteria>;
```

For example, the following statement causes the inferior infarction statement to be generated on the report if the criteria are met:

```
PRINT     #IMI10        AB
  "INFERIOR INFARCTION, OLD".
    "Q WAVES IN 2, 3, F OR SUPERIOR INITIAL VECTOR"

    IF     (Q:DURATION. . . ;
```

where IMI10 is the diagnostic code corresponding to the statement in quotation marks and AB stands for an abnormal finding.

It is important to remember that whereas a cardiologist reading an ECG can immediately discount many classes of diagnoses, a computer program must check them all sequentially. Within a category, criteria essentially become more and more restrictive. Consequently, in ECL, criteria met for any given statement automatically suppress any previous statements that would have been printed from the category. For complete control, however, the criteria writer must be able to suppress statements from other categories selectively and to branch over categories or sections of criteria inconsistent with the current criteria. These are expressed in the sentences:

```
SUPPRESS <list of statements> IF <medical criteria> ;
GOTO <criteria label> IF <medical criteria> ;
```

For example, the sentence:

```
SUPPRESS AXIS IF #LAFB0 OR #LPFB OR #BIFBA OR
     #BIFBP OR ANY WPW STATEMENT OR #CLBBB;
```

would cause suppression of any abnormal axis statement in the presence of a left anterior fascicular block, a left posterior fascicular block, a bifascicular block, a WPW (Wolff-Parkinson-White syndrome), or a complete left bundle branch block.

The following could appear at the beginning of the RVH category:

```
GOTO LAE0 IF MEAN:QRS:DURATION GT 120;
```

which would cause the program to skip to the top of the next category—left atrial enlargement—if the QRS duration is greater than 120 ms, thereby disabling the entire RVH category.

Finally, user-defined variables can be assigned values, perhaps derived from mathematical combinations of measurements, by the sentence:

```
SET <variable> = <expression> IF <medical criteria>;
```

In the sentence:

```
SET WIDE:VAT = YES IF (VAT GT 50) IN 1 OF V5 V6 AND NOT ANY
                    VCD STATEMENT;
```

a wide ventricular activation time parameter for use in LVH criteria is set if VAT is greater than 50 ms in either V5 or V6 in the absence of any ventricular conduction delay diagnoses.

Criteria

Medical criteria are expressed in terms of enhanced logical expressions that are both concise and unambiguous. The sentence:

. . . Positive P waves greater than 0.25 mV in two of leads 1, 2, or aVR . . .

is written in ECL as:

```
. . . (P:AMPLITUDE IS POSITIVE AND GT .25) IN 2 OF L1 L2 AVR . . .
```

where the relation ". . . greater than . . ." is abbreviated to GT and the lead coincidence requirements are specified in both English and ECL by nearly identical phrases starting with the word IN. The parentheses allow the lead coincidence requirements to be applied to more than one criterion.

In everyday communication of ECG criteria, knowledge of the basic form of an ECG complex is implied. For example, when a cardiologist says:

". . . Q amplitude is greater than 0.1 mV . . ."

it is assumed that the Q wave is negative. In ECL this would be written as:

```
. . .Q:AMPLITUDE GT 0.1 . . .
```

and it means exactly what the cardiologist expects.

If the sign of the amplitude is relevant, then this can be tested explicitly in ECL. For example:

. . .P:AMPLITUDE IS POSITIVE AND GT 0.25 . . .

Other Criteria

In addition to ECG measurements, there is other information available to the ECL criteria writer. For example, indexes of the quality of data specifying various types and levels of noise such as missing leads, muscle tremor, ac interference, etc. can be used to indicate that the interpretation may have been compromised because important information was missing. Also, patient information entered via the cardiograph such as age, sex, body build, medication, previous diagnosis (e.g., CAD, hypertension), blood pressure, race, etc. can be used in the analysis where appropriate.

Self Documentation

To make the final criteria documentation as readable as possible, the language is free from formatting restrictions. The compiler provides extensive error checking and reporting, and optionally provides the following final criteria documentation:

- Table of contents
- The declaration block, which consists of a list of user-declared variables, RX names and codes, previous DX names and codes, and modifiers
- Diagnostic criteria by category
- A list of statements with text formatted as it would appear in the diagnostic reports
- Cross-reference information

ECL has been used extensively by HP 5600C ECG Management System users to optimize the HP Analysis Program for their particular ECG reading style. In addition, our research-oriented users have made use of ECL to communicate new criteria to other users for evaluation and comment.

main modules (Fig. 3). The Quality Monitor examines the incoming data for various types of noise contamination and passes this information back to the operator for corrective action and into the Data Conditioning module. This module applies a 50- or 60-Hz notch filter to the data if any ac noise is detected. All ac noise is thus eliminated before the data is sent to the Pattern Recognition module, which adaptively filters each P, QRS, ST, and T region to produce a comprehensive set of measurements for each region. These measurements are then polled and passed to the Criteria Module. This module contains the medical criteria—such as those found in electrocardiography textbooks—used for making the diagnosis. On an HP 5600C ECG Management System, additional capabilities include a diagnosis that includes a comparison with the previous ECG and modification of the criteria by the user.

Waveform Boundary Indicator

The most crucial decisions for the Pattern Recognition module are to identify all the QRS waveforms correctly and to allocate an accurate search region for all remaining component waveforms, (i.e., P and ST-T waveforms). A search window should be as narrow as possible and yet wide enough to contain the earliest onset and latest portion of a particular simultaneous three-channel waveform. Enhancing the features of the signal from those of noise is critical to reduce the possibilities of mislabeling waveform components in the early stages. When searching for a QRS waveform, a tall T wave is just as undesirable as noise contamination. Therefore, the goal is to transform the ECG signal into different forms so that there is maximum separation between the P, QRS, and T waves and noise. The transformed signal is called the waveform boundary indicator (WBI) and it is derived from the simultaneous three-channel ECG signal. The WBI is based on the combined magnitudes of the first and second derivatives as follows:

$$\mathrm{WBI}(k) = C_1 \sum_{i=1}^{3} |f_i'(k)| + C_2 \sum_{i=1}^{3} |f_i''(k)| \qquad (1)$$

where k refers to the kth sampled data point, and C_1 and C_2 are constants. The subscript i refers to the ECG signal from channel i. In other words, the WBI is a weighted average of the speed and acceleration of the ECG signal over the three channels. The equations for the first and second derivatives are:

$$f'(k) = f(k+1) - f(k-1) \qquad (2)$$

$$f''(k) = f(k+2) - 2f(k) + f(k-2) \qquad (3)$$

The WBI works well in detecting QRS waveforms and in discriminating against T and P waves. However, it tends to be narrower than the simultaneous three-channel QRS complex. Furthermore, it does not significantly enhance the T and P waves over the noise. To correct both of these problems, another version of the WBI, called WBIF, is created from the same data after digital filtering. The filtered data $\overline{f(n)}$ is obtained by convolving the raw ECG data f(n) with the impulse response h(n).

$$\overline{f(n)} = h*f = \sum_{k=-(N-1)}^{N-1} h(k)f(n-k) \qquad (4)$$

The impulse response h(n) is given by the triangular waveform shown in Fig. 4.

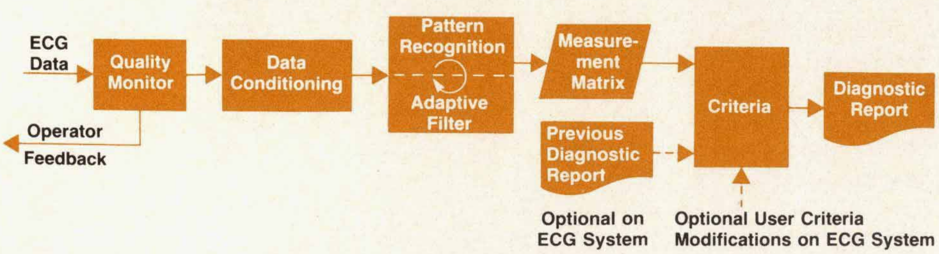


Fig. 3. *Analysis program structure.*

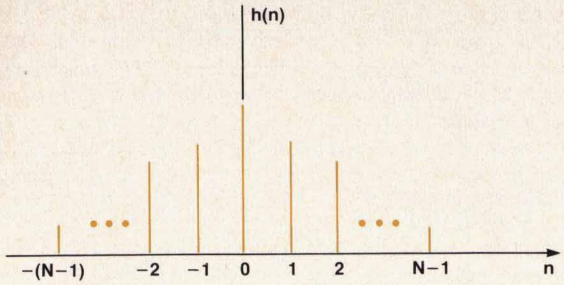


Fig. 4. *Impulse response to be convolved with raw ECG data.*

Fig. 5 shows a typical set of three-channel ECG waveforms and the associated WBI and WBIF. Note that the QRS complex can be easily extracted from a "high" condition in both the WBI and the WBIF. After identifying and then excluding the QRS regions, the remaining "high" regions are the T and P waves. While the WBIs identify the initial search regions, all the measurement parameters are derived from the original raw data.

Morphology Measurements

Within the search regions established by the WBIs, each waveform component is processed to establish its:

- Onset and endpoints
- Amplitudes, durations, shapes, areas, etc.
- Delta waves, notches, slurs, and pacemaker spikes.

The QRS complexes are analyzed first, followed by the ST-T waves, and finally the P waves. Since the QRS waveforms are the most prominent and therefore the ones that can be most confidently measured, they are analyzed first and their exact onsets and endpoints are used to improve the performance of the P and ST-T measurements.

The exact onset and endpoints of P and T waves are located through the analysis of four sets of data within the predetermined WBI region. These data sets include (see Fig. 6):

- The raw ECG data f(x)
- The data smoothed by adaptive filtering
- The first difference function f′(x)
- The second difference function f″(x).

By using the properties of the maxima and minima of f′(x) and f″(x), approximate departure points of the ECG waveform from the baseline can be located. These are then used in conjunction with the empirical threshold to determine the exact onset and endpoints.

Adaptive Filtering. Each waveform component such as a P, QRS, ST, or T wave is examined and filtered independently for noise contamination before it is measured. The measured amount of noise is the waveform's "signature." A signature is defined in terms of the critical points such as maxima, minima, and zero crossings of the data, f′(x), and f″(x). The data is iteratively smoothed until the signature is within an acceptable level before analysis is performed. The filter uses a quadratic polynomial least-squares technique to calculate the smoothed data, smoothed f′(x), and smoothed f″(x). The amount of smoothing is specified by the number of data points used to fit the curve.

Rhythm Analysis. After all beats are analyzed and measured, they are classified into groups based on RR interval, QRS duration, PR interval, and pacemaker spike (present or absent). For each group, mean values are calculated for the RR, PR, QRS, and QT regions, etc. If more than one group exists, a selection process chooses a group that represents the intrinsic rhythm. Beats in this group will be used for contour measurements while beats in all other groups originating from rhythm disturbances are excluded.

Measurement Matrix. At this point in the analysis, each of the many P-QRS-T complexes in each of the twelve leads has been measured in detail. To apply clinical diagnostic criteria, twelve representative subsets of measurements, one for each of the twelve leads, are polled from the larger set. The measurements for the many complexes are reduced to a subset by means of a series of confidence checks and weighted averages. Complexes originating from rhythm abnormalities are not included in these averages.

Diagnostic Criteria

The last module of the analysis program contains all the

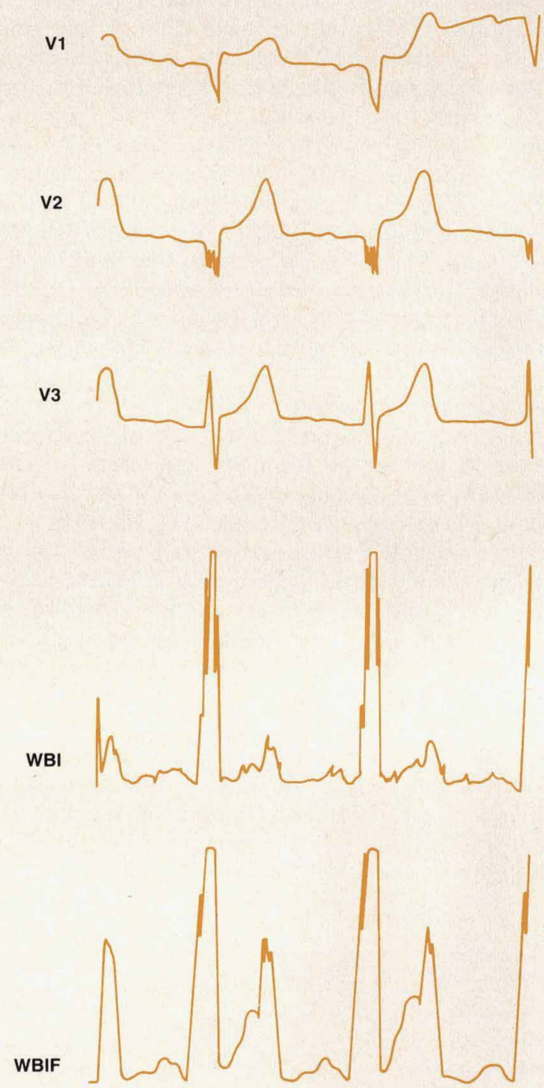


Fig. 5. *Typical set of three-channel ECG waveforms and their associated WBI and WBIF.*

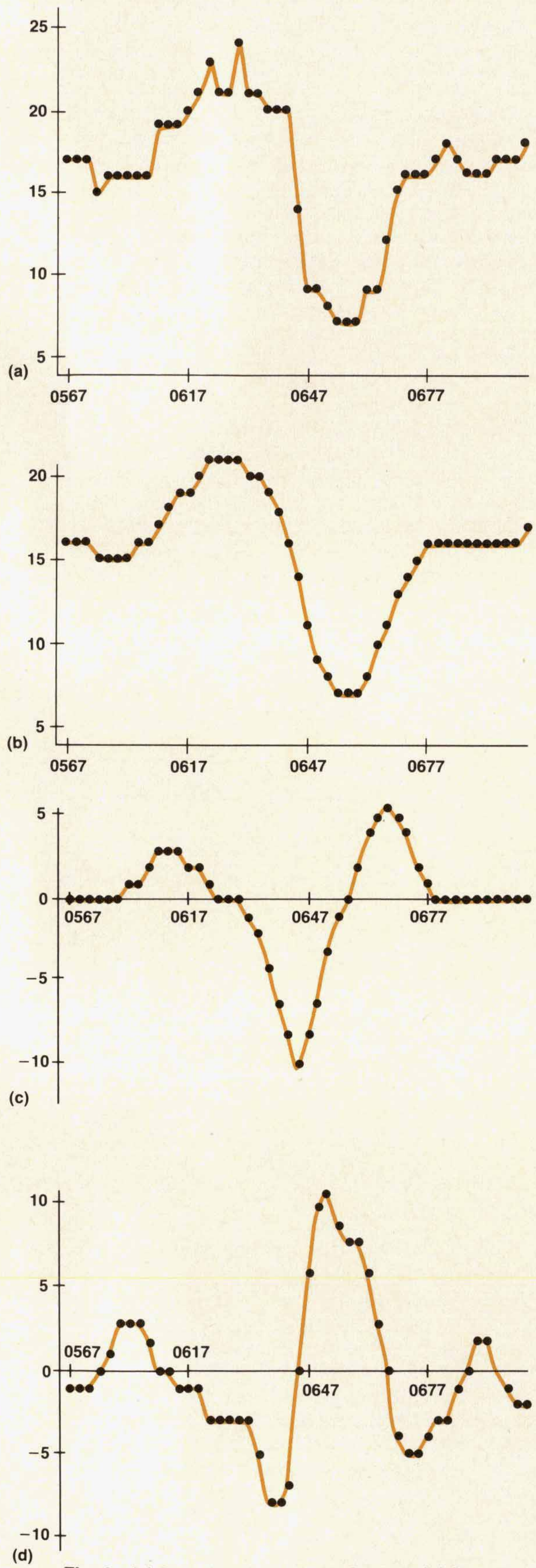


Fig. 6. *(a) Raw data for biphase P wave. (b) Data smoothed by adaptive filtering. (c) First difference function of (b). (d) Second difference function of (b).*

medical criteria for making the interpretation. It is the medical knowledge base. It "sees" the ECG by means of all the previous analysis modules. What it sees is the measurement matrix. The criteria module is written in a cardiologist-readable language called ECG Criteria Language, or ECL (see box on page 30 and reference 6). More than just a language, ECL is an entire programming environment that allows the clinical user to modify, enhance, and optimize the criteria on an ECG management system. The criteria program is similar to an electrocardiography textbook. Each major abnormality category is a separate category in the criteria. For example, there is a category for left ventricular hypertrophy, a separate one for right ventricular hypertrophy, and yet another one for inferior infarct. Within each category are the rules for diagnosing the various gradations of a particular abnormality. Similar to medical texts, there are also rules that specify the relationships between diagnoses in different categories. In short, the criteria written in ECL form a powerful tool to carry out clinical criteria research or simply to use and understand the analysis program's medical logic.

References

1. H.V. Pipberger, "Use of Computers in Electrocardiogram Interpretation," *Circulation Research*, Vol. 25, 1962, p. 555.
2. C.A. Caceres, et al, "Computer Extraction of Electrocardiographic Parameters," *ibid.*, p. 256.
3. M. Hodges, "A Clinical Evaluation of the HP ECG Analysis Program: Program Accuracy and Value of Adjustable Criteria," *Computers in Cardiology*, September 1979.
4. Balda, et al, "The HP Analysis Program," *Proceedings, IFIP Conference on Trends in Computer-Processed Electrocardiograms*, 1977.
5. J.C. Doue, "The T Wave Algorithm of the HP Program," *Proceedings of Engineering Foundation: Computer Interpretation of ECGs*, 1982.
6. *ECL Programmers Reference Manual*, Hewlett-Packard Company Medical Products Group Publication #05600-91819.

Pediatric Criteria

Pediatric ECG interpretation is a special challenge for several reasons:

- The availability of pediatric cardiologists is much more limited and is restricted mostly to major urban centers.
- Pediatric criteria contain a large number of age-dependent tables, which causes them to be significantly more difficult to remember than adult criteria.

These difficulties provided the stimulus to extend computer interpretation to pediatric ECGs. Coupled with the availability of a standard criteria development tool, ECL (see box on page 30), the addition of a pediatric criteria module to the HP 4760A Cardiograph family's ECG analysis program became a reality.

Rapidly Changing ECG Morphology

Pediatric ECG criteria are age-dependent, and are therefore extremely complex. An example of this complexity is the right ventricular dominance at birth, which changes to left ventricular dominance with age. This differs significantly from a diagnosis of right ventricular hypertrophy in an adult. Since the QRS shift in the early weeks of life is frequently measured in days, pediatric electrocardiographers must memorize, or have available, rather detailed tables of QRS voltages, axes, intervals, and other parameters versus age for many leads. An example of these frequently consulted charts is shown in Fig. 1. The chart shows the complexity of pediatric criteria. Here, the heart rate versus age is shown. Tables such as these are incorporated into the HP 4760A's Pediatric Program Module.

ECL and User Development

Recognizing the special challenges of pediatric interpretation, Dr. Laks and his colleagues at Harbor-UCLA Medical Center set out in the mid-1970s to develop a pediatric criteria package for the HP ECG analysis program. This effort was greatly facilitated by the availability of the ECG Criteria Language, called ECL, as a standard development tool on an HP 5600C ECG Management System. ECL is a high-level computer language that bridges the gap between the programmer and the cardiologist. ECL, unlike other computer languages such as Fortran, allows the physician to read the computer criteria directly, facilitating criteria development and enhancement. As a result, the translation of the large tables of pediatric ECG values and the development of unique pediatric terminology were vastly simplified.

After the pediatric criteria were successfully developed and extensively tested at Harbor-UCLA, the data was submitted to HP for evaluation. In addition to examining the criteria from a technical viewpoint for incorporating into the ECG analysis program, HP also enlisted the help of Dr. Walter Gamble at Children's Hospital of Boston to carry out an independent clinical evaluation.

All the evaluations are now completed and the pediatric analysis program is receiving a high degree of acceptance by pediatricians. Nevertheless, continuing improvements to the pediatric criteria and the adult criteria is an ongoing process, which is being propelled by these evaluation results, other user critiques, and ECL as a development tool.

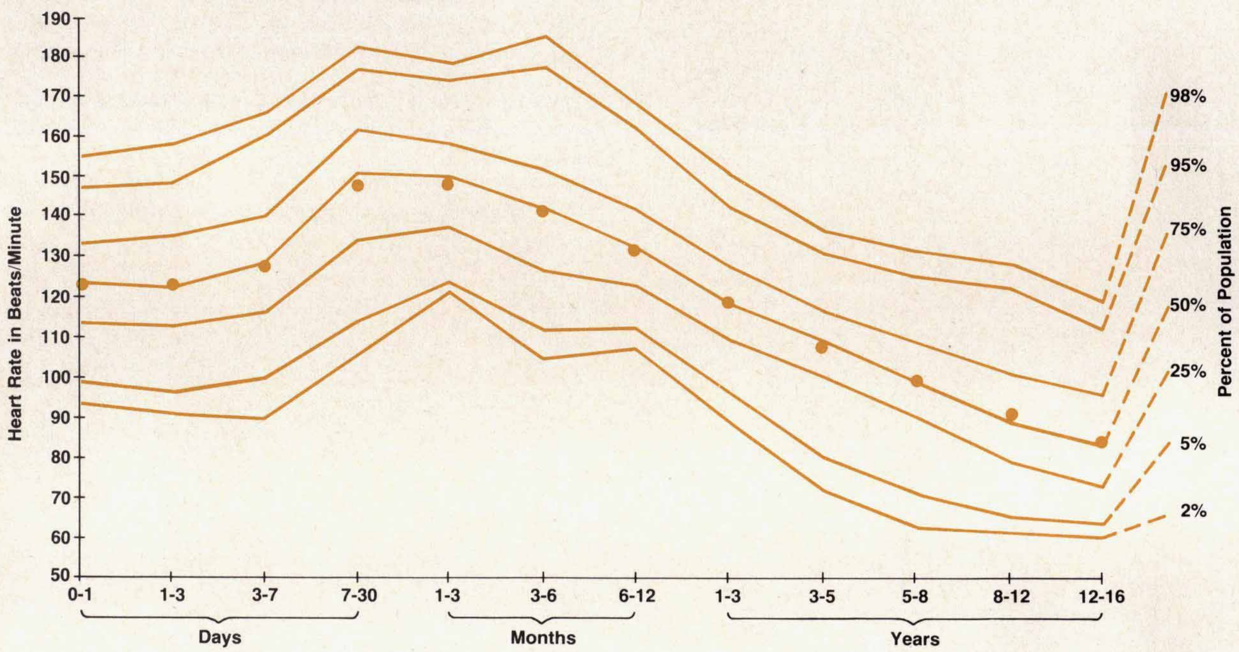


Fig. 1. *Example of age-dependent pediatric ECG chart. The dots represent the average heart rate versus age. (Derived from percentile charts by A. Davignon, et al, Pediatric Cardiology, Vol. 1, 1979/1980, pp. 133-152.)*

Authors

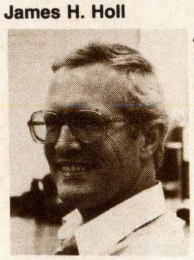
September 1985

4 — HP 3000 Computer System

Frank E. La Fetra, Jr.

Skip La Fetra was born in Los Angeles, California and studied electrical engineering at Stanford University, from which he received his BSEE and MSEE degrees in 1976 and 1977. After joining HP in 1977, his first assignments as a development engineer involved circuit design and analysis, reliability, burn-in, and automated test equipment. Later he worked on the HP 3000 Series 68 and Series 37 Computers and is now an R&D project manager. Skip is a registered professional engineer and a member of the IEEE, and is interested in small, multiuser computer systems. He and his wife live in Sunnyvale, California. He is an avid bicyclist and likes to tinker with personal computers.

James H. Holl



A native Californian, Jim Holl was born in Palo Alto, studied electrical engineering at the University of California at Berkeley (BSEE 1966) and the University of Santa Clara (MSEE 1971), and came to HP in 1969. He is an R&D section manager and was the section manager responsible for the HP 3000 Series 37 Computer. He was also a member of the original R&D team that developed the HP 3000. Jim lives in Cupertino, California with his wife and two sons, is a youth soccer referee, and has been a YMCA Indian Guides leader. He loves sports, particularly ultimate frisbee, managing to keep up with HP teammates who are often 10 to 20 years younger than he is.

7 — Computer Architecture

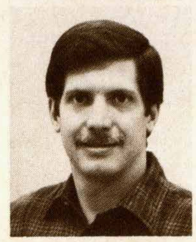
Frederic C. Amerson

An R&D section manager at HP's Data Systems Division, Rick Amerson has contributed to the development of both the HP 3000 and HP 1000 Computers. He worked on the plotter interface for the original HP 3000, was a project manager for the HP 3000 Series 64, and is a section manager for HP 1000 hardware development. Rick received a BSEE degree from the Georgia Institute of Technology in 1972 and came to HP the same year. He now lives in Santa Clara, California, is a church pianist, and enjoys downhill skiing. He is also a commercial pilot with instrument and multiengine ratings.

Patria G. Alvarez

A native of San Jose, California, Pat Alvarez earned a BSCS degree from San Jose State University in 1983 before coming to HP the same year. She currently works on software for the HP 3000 operating system and has also developed and maintained hardware design tools for the HP 3000 Series 37. This fall she will be working on an MS degree in computer science at Stanford University. Outside of work, Pat enjoys tennis and making handicrafts and clothing.

John R. Obermeyer



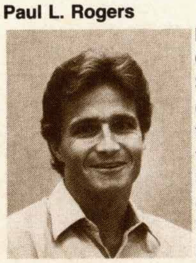
John Obermeyer studied electrical engineering at Northwestern University and came to HP in 1981, the same year he received his BS degree. He also completed an MS degree in computer science at Stanford University in 1984. He contributed to the design of the terminal interface controller for the HP 3000 Series 37 Computer and also worked on the diagnostic and utility systems. Currently, he is working on VLSI chip design. John was born in Cincinnati, Ohio, lives in San Jose, California with his wife, and is an advisor and choir director for youth groups in his church. His outside interests include painting, drawing, woodcarving, and volleyball. He also collects fossils, mostly from the Ordovician period.

Greg L. Gilliom

Currently an R&D project manager for HP 1000 Computer products, Greg Gilliom has been with HP since 1979. He worked as a production engineer on a number of models of HP 3000 Computers, and later developed diagnostics and microcode for the HP 3000 as an R&D engineer and project manager. He was the project manager for the microcode on the HP 3000 Series 37. Greg was born in St. Charles, Missouri and graduated from the University of Missouri with a bachelor's degree in electrical engineering in 1979. He lives in Campbell, California, is single, and has many athletic interests, including sailing, windsurfing, waterskiing, scuba diving, skiing, and ultimate frisbee.

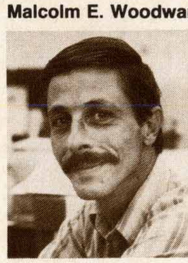
13 — Simulation

Paul L. Rogers



Paul Rogers designed the CPU board for the HP 3000 Series 37 Computer and contributed to the design of the terminal interface controller. At HP since 1981, his other experience includes work on the hardware cache for the HP 3000 Series 64 Computer. He is a graduate of the University of California at Berkeley (BSEE 1981) and recently completed an MSEE degree at San Jose State University through the HP fellowship program. Paul lives in Santa Clara, California and has a variety of outside interests. He plays water polo, ultimate frisbee, or basketball with other HP employees at lunchtime and also enjoys cooking, woodworking, scuba diving, water skiing, and sailing.

Malcolm E. Woodward



Born in Ontario, Oregon, Woody Woodward served in the U.S. Marine Corps before coming to HP in 1972. While he was in the Marine Corps he supervised the maintenance of tactical data systems, and in his first HP job he worked as a technician on system I/O and on the HP 2100 Computer. Later, as an R&D staff member, he contributed to the development of the HP 3000 Series 64 and designed the peripheral interface channel for the HP 3000 Series 37. He is currently an R&D project manager. Woody lives in Sunnyvale, California with his son and has one other son and two daughters. He is an amateur radio operator (W6PLT) and also likes fishing and tinkering with cars.

17 — Debug Tools

Edwin G. Wong

With HP since 1979, Ed Wong wrote the diagnostic microcode for the HP 3000 Series 37 and is currently working on CMOS VLSI chip design. He also designed an I/O card for the HP 1000 L-Series Computer and a memory controller for another product. A California native, Ed was born in San Francisco and earned a BS degree from the University of California at Santa Barbara in 1978. He expects to receive an MS degree from the University of Santa Clara in 1986. He is a resident of Sunnyvale, supports the Big Brothers youth organization, and is active in his church. He enjoys windsurfing, running marathons, and participating in triathlons.

William M. Parrish

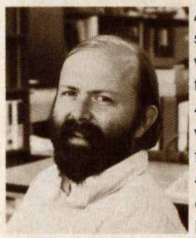
Bill Parrish was born in Dallas, Texas and is a graduate of the University of California at Santa Barbara (BS 1973). At HP since 1974, he has contributed to the development of both the Series 64 and Series 37 HP 3000 Computers and is presently investigating the field supportability of future products. He is a member of both the IEEE and the ACM. Bill and his wife live in Meadow Vista, California and enjoy taking ballet lessons together. His other interests include photography, travel, and playing the piano and organ.

Eric B. Decker

At HP since 1980, Eric Decker has written microcode for the HP 3000 Series 37, 64, and 68 Computers. He also contributed to the development of the terminal controller for the Series 64 and 68 and to the development of the HP 75C Handheld Computer. He is interested in distributed systems, computer architecture, and the societal impact of computers. He has attended Case Institute of Technology, Iowa State University, Stanford University, and California State University at Chico. Eric lives with his companion and two children in Scotts Valley, California. He says he likes t'ai chi ch'uan, intellectual pursuits, and "yard destruction."

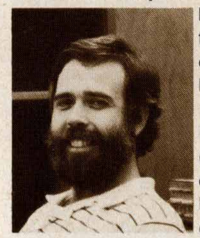
23 Cardiograph Family

Peter H. Dorward



At HP's Andover Division since 1975, Peter Dorward was project manager for the HP 4750A and HP 4760A cardiographs. He was also a project leader for electronics on the HP 4700A Cardiograph and developed software for the HP 5600C ECG Management System. Peter was born in Lancaster, Pennsylvania and received an AB degree from Dartmouth College in 1973 and a Master of Engineering degree from the Thayer School of Engineering in 1975. He lives in Harvard, Massachusetts with his wife and daughter and enjoys softball and skiing. He is also renovating a 100-year-old Victorian farmhouse, raises chickens, and grows fruit trees and Christmas trees.

Steven A. Scampini



Born in Bristol, Connecticut, Steve Scampini was educated at Rensselaer Polytechnic Institute (BSEE 1972) and at the California Institute of Technology (MSEE 1973). He worked on undersea electronics at Bell Laboratories, then came to HP in 1976. At HP he has contributed to the development of the HP 4700A, the HP 4750A, and the HP 4760A cardiographs. He was also the author of an HP Journal article on the HP 4700A. Steve lives in Reading, Massachusetts and likes photography, running, and cross-country skiing.

Robert H. Banta, Jr.

At HP since 1980, Bob Banta was responsible for the integrated tape backup and HP-IB interface for the HP 7908, HP 7911, and HP 7912 disc products. He was one of the developers of the HP 4750A Cardiograph and was the software project leader on the HP 4760A Cardiograph. Bob was born in Neptune, New Jersey and received his BS degree from Duke University in 1980. Now a resident of North Andover, Massachusetts, he enjoys bicycling, hiking, and soaring.

29 ECG Analysis

Anthony G. Vallance

Tony Vallance was born in Amersham, England and studied at Woolwich Polytechnic (BS 1963) and Northeastern University (MSEE 1972). At HP since 1974, he is a section manager at the Waltham Division and was also a section manager at the Andover Division. In his earlier assignments he was project manager for the HP 5600C ECG Management System and project manager for the system and test software for the HP 77020A ultrasound imaging system. He has published several technical papers and is a member of the IEE and ACM. Tony lives with his wife and two sons in Westford, Massachusetts and is interested in sailing and astronomy.

John C. Doue

John Doue was born in China and educated in the U.S. He attended the University of California at Berkeley, receiving a BSEE degree in 1967 and an MSCS degree in 1968. After working as a software engineer at two other electronics companies, he joined HP in 1972. He has made a number of contributions to the development of cardiograph products and is presently a project leader for the analysis program for the products. He has published papers in conference proceedings, is a member of the American Heart Association, and is interested in the application of artificial intelligence to electrocardiograph analysis. John lives with his wife in Manchester, Massachusetts. They designed and built an A-frame cabin, all with hand tools, in the Maine woods.

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

HEWLETT-PACKARD JOURNAL

September 1985 Volume 36 • Number 9

Technical Information from the Laboratories of Hewlett-Packard Company

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

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

02000352166&&COLL&RHOO
MR R H COLLINS
2732 CHEROKEE DR
BIRMINGHAM AL 35216

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

5953-8539