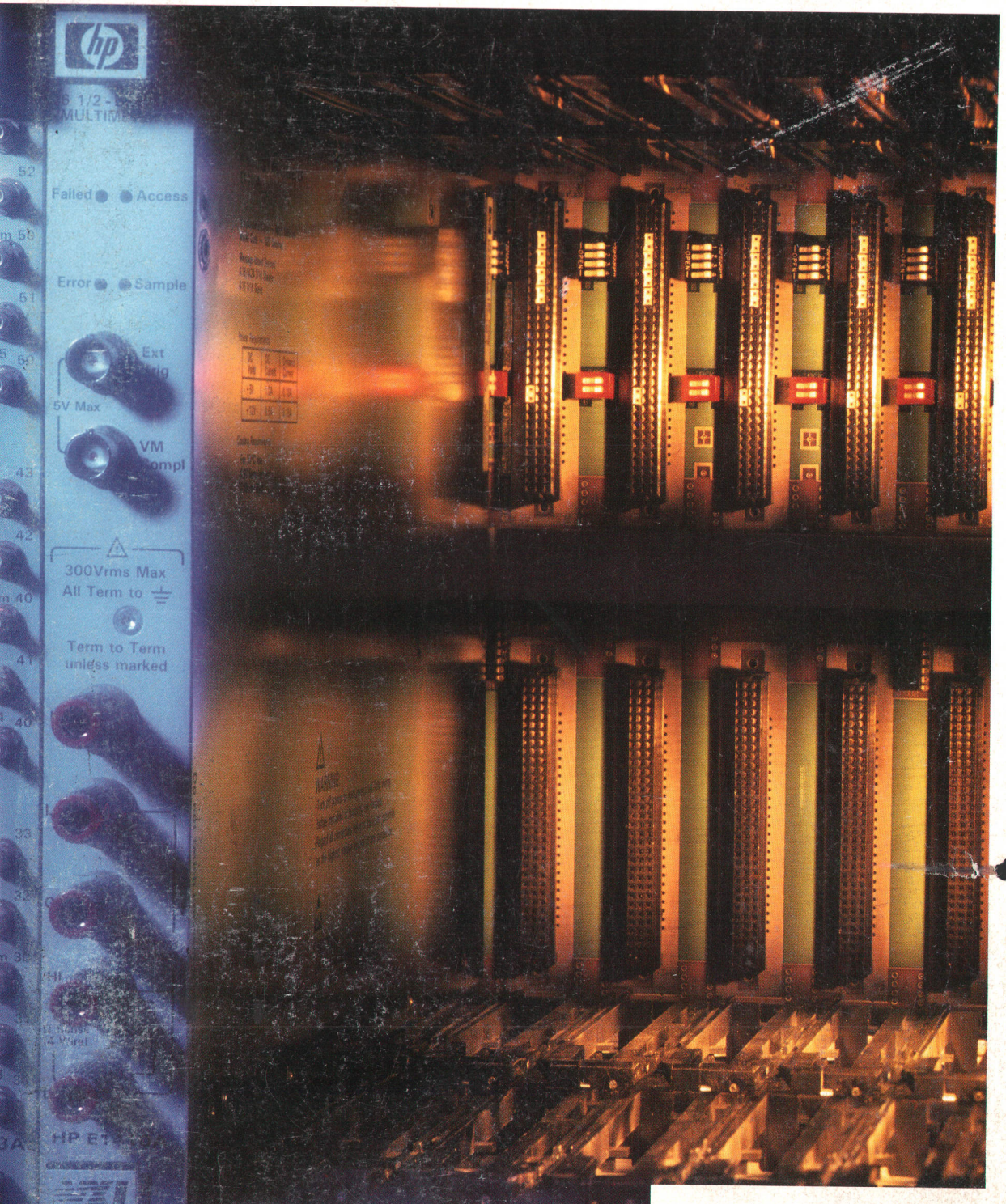
HEWLETT-PACKARD
JOURNAL
April 1992



HEWLETT PACKARD

HEWLETT-PACKARD
JOURNAL

April 1992 Volume 43 • Number 2

Articles

Editor, Richard P. Dolan • Associate Editor, Charles L. Leath • Publication Production Manager, Susan E. Wright
Illustration, Renée D. Pighini • Typography/Layout, Rita C. Smith

©Hewlett-Packard Company 1992 Printed in U.S.A.

Departments

The Hewlett-Packard Journal is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

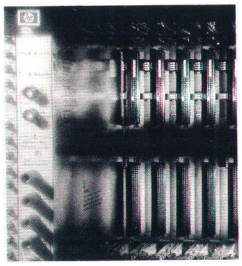
Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP address on the back cover that is closest to you. When submitting a change of address, please include your zip or postal code and a copy of your old label. Free subscriptions may not be available in all countries.

Submissions: Although articles in the Hewlett-Packard Journal are primarily authored by HP employees, articles from non-HP authors dealing with HP-related research or solutions to technical problems made possible by using HP equipment are also considered for publication. Please contact the Editor before submitting such articles. Also, the Hewlett-Packard Journal encourages technical discussions of the topics presented in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

Copyright © 1992 Hewlett-Packard Company. All rights reserved. Permission to copy without fee all or part of this publication is hereby granted provided that 1) the copies are not made, used, displayed, or distributed for commercial advantage; 2) the Hewlett-Packard Company copyright notice and the title of the publication and date appear on the copies; and 3) a notice stating that the copying is by permission of the Hewlett-Packard Company.

Please address inquiries, submissions, and requests to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304 U.S.A.

In this Issue



VXIbus is a new interconnection standard for modular instruments. It defines an architecture, communication protocols, and electromagnetic compatibility requirements for a mainframe cardcage and for instrument modules that plug into the mainframe and work together as a system. The architecture is open to all manufacturers, so users can mix instruments from different manufacturers in the same system. The history of the VXIbus begins in 1979, when Motorola Semiconductor Products Corporation published a description of what came to be known as the VMEbus. Three years later, the International Electrotechnical Commission proposed that it become an international standard. In 1987, under pressure from both military and commercial customers for an open architecture for modular instruments, a group of instrument manufacturers, including HP, agreed to develop a modular instrument standard based on the VMEbus. The VXIbus is the result. VXI stands for "VMEbus extensions for instrumentation." The VXIbus doesn't replace the widely used HP-IB standard (IEEE 488, IEC 625). In fact, it's very common for VXIbus systems to communicate with a computer or other controller over the HP-IB, and for HP-IB instruments and controllers, VXIbus mainframes, and MMS mainframes (MMS is another open modular architecture optimized for microwave applications) to coexist in the same measurement system. HP VXIbus products include mainframes, existing HP instruments redesigned to fit on VXIbus cards, and new instruments designed specifically for the VXIbus. The article on page 6 describes the VXIbus standard and gives several examples of VXIbus measurement systems. Details of HP's implementation are covered in the articles on pages 15, 24, and 29, which deal with internal HP standards that supplement the VXIbus standard, the design of HP's mainframe firmware, and the operating system for HP VXIbus command modules. A library of C functions that helps test programmers create VXIbus applications using the high-level C language is the subject of the article on page 35. HP VXIbus switching and interconnect products are described in the articles on pages 41 and 52, and the hardware and software designs of a VXIbus functional tester for digital electronic assemblies are presented on pages 59 and 69. HP's own use of VXIbus instrumentation for manufacturing test is discussed in the article on page 75.

In radar, telemetry, navigation, and communications, microwave signals are typically pulsed rather than continuous. Engineers designing and testing such systems need to make many measurements on the envelope of the microwave pulses, such as peak and average power, rise and fall times, pulse width and repetition rate, overshoot, and others. Until now, the best way to make these measurements was a microwave detector connected to an oscilloscope. Like many homemade solutions, this arrangement is difficult to calibrate for accurate measurements. Among the reasons are the nonlinearity and temperature sensitivity of the diode detector, the frequency response of the detector output, mismatch error, and harmonics. This solution also suffers from slow response and limited oscilloscope sensitivity. Overcoming these problems was the motivation for the development of a new type of microwave instrument, the HP 8990A peak power analyzer. Taking a fresh look at the challenges of diode detection, the designers created special power sensors using gallium arsenide detector diodes. In the analyzer design, they incorporated switched amplification and processing of the pulse envelope signals, leveraged modern digital oscilloscope technology, and used microprocessor power to implement a new calibration approach that makes calibration mostly automatic. The article on page 81 describes the problems of microwave pulsed power measurements and tells how the design of the peak power analyzer addresses them. The contributions of gallium arsenide technology to the sensor and analyzer designs are discussed in the article on page 90. The calibration approach and firmware design are covered in the article on page 95.

A more familiar instrument, the network analyzer, measures the response of some device to a stimulus signal and displays the response as a function of the frequency of the input signal. The device under test might be a filter, a resonator such as a quartz crystal, a circuit, an integrated function block, or a complex discrete device. The HP 8751A network analyzer is designed to deliver state-of-the-art performance in its frequency range of 5 hertz to 500 megahertz, both for testing such devices in production and for evaluating them in the development laboratory. Its three-processor design achieves a very fast measurement speed of 400 microseconds per point, which not only improves production throughput but also gives the analyzer a real-time response that displays the results of adjustments as they are made. The HP 8751A's accuracy, resolution, sensitivity, and dynamic range match or exceed those of other comparable HP network analyzers. It has the powerful built-in analysis functions that users have come to expect, and it offers some new capabilities, such as simulating impedance matching networks for the device under test, making simultaneous high-speed and high-accuracy measurements in separate frequency ranges, and simultaneously displaying the three key parameters for filter applications. The design of this analyzer is described in the article on page 101.

In the early 1980s, when the HP 9000 Series 200 computer was HP's primary instrument controller, HP BASIC became a widely used instrument control programming language. A few years later, many HP customers were using personal computers for instrument control but wanted to continue to program in HP BASIC, and HP responded by offering a plug-in BASIC language processor card for the PC. The third generation of this card, the HP 82324A measurement coprocessor, is described in the article on page 110. Its contributions are higher calculation speed, faster HP-IB input/output performance, and data transfer by direct memory access for better overall system performance. The card also provides more efficient communication between its own processor and the PC's and makes it possible to develop complex applications involving up to three measurement coprocessors interacting with PC applications such as a spreadsheet.

R.P. Dolan
Editor

Cover

A view of a VXIbus module and the backplane of a VXIbus mainframe. Because the backplane is the embodiment of an industry standard, modules from different manufacturers can communicate over it. Restrictions on the VXIbus mainframe and modules ensure electromagnetic compatibility.

What's Ahead

The June issue will feature recent developments in system software for HP 9000 PA-RISC workstations, including the latest optimizing compilers, shared libraries, and the HP-UX operating system for the recently introduced, very fast HP 9000 Series 700 workstations. There will also be three research reports that were presented at the 1991 HP Technical Women's Conference. One is on a parallel computing architecture for ray-traced image generation, one is on the evaluation of printed image quality using spatial frequency methods, and one is on integration of an electronic dictionary into a natural language processing system.

VXIbus: A Standard for Test and Measurement System Architecture

The VXIbus standard defines an open architecture that allows instrumentation and processors from various manufacturers to operate together within a single chassis or mainframe.

by Lawrence A. DesJardin

In July 1987, Hewlett-Packard and four other major electronic instrument manufacturers jointly announced their support for a new instrumentation standard called VXIbus.[1] An abbreviation for VMEbus Extensions for Instrumentation, VXIbus is an open architecture that allows instrumentation and processor modules from various manufacturers to operate together within a single chassis or mainframe. The VXIbus Consortium was formed shortly thereafter to develop and maintain the VXIbus specification, the primary technical document that describes the mechanical, electrical, and communication interface requirements for VXIbus products. Since that time, more than 50 manufacturers have announced hundreds of VXIbus products encompassing mainframes, instrument modules, computer modules, interfaces, fixturing, and software. VXIbus applications range from research of high-energy physics to computer-aided test systems for production test of electronic assemblies, and VXIbus is rapidly becoming a mainstream test and measurement architecture. This article presents an overview of the VXIbus standard, and describes how the VXIbus architectural features are used at a system level to create a VXIbus measurement system.

What is VXIbus?

The VXIbus is based on the VMEbus computer backplane standard. While VMEbus has two module sizes, A and B, VXIbus offers four sizes by adding C and D sizes (see Fig. 1). The C and D-size modules are wider, allowing two printed circuit boards and enveloping shields to be used. Mainframes also come in these four sizes and include methods for accepting smaller modules as well (see "The HP VXIbus Mainframes," on page 9). To keep the architecture flexible for a wide range of applications, the VXIbus does not specify the amount of power or cooling a given mainframe must supply, but it does specify the way in which module and mainframe power requirements and capabilities must be documented. This allows a user to select modules and mainframes for a given application, knowing a priori that a particular product set will work together. VXIbus defines three backplane connectors, labeled P1, P2, and P3. Only the P1 connector is required of all modules and mainframes; the others are optional. P1 contains the required VMEbus data transfer bus capability, and can address the A16 (64K-byte) and A24 (16M-byte) address spaces using 8-bit or 16-bit data transfers. The VXIbus autoconfiguration registers and the protocol registers for message-based devices are located in the A16 address space, allowing all required communication to occur using only P1. The P2 connector, optional on B, C, and D sizes, extends the data transfer bus to include 32-bit transfers and the A32 (4G-byte) address space, as in the VMEbus standard. However, while VMEbus left 64 pins of P2 undefined, VXIbus defines these remaining pins to be a set of clocks, identification lines, power supplies, and trigger buses.[2] It also defines a local bus that connects adjacent slots using these pins. The optional P3 is only found on D-size modules, and is fully defined as additional clocks, power, trigger buses and local bus lines.

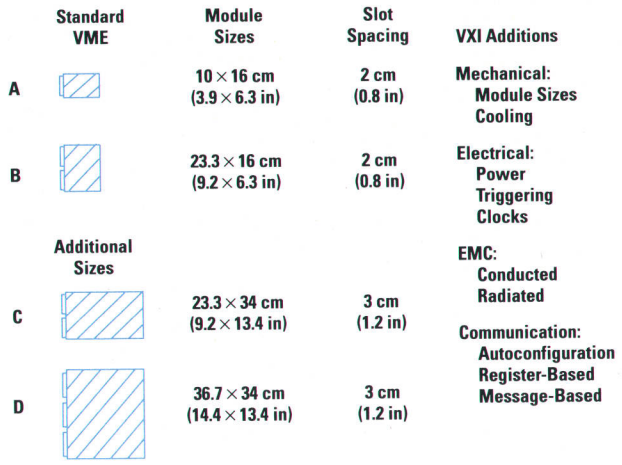


Fig. 1. The VXIbus standard (VMEbus Extensions for Instrumentation) includes the two original VMEbus module sizes, A and B, plus two new modules, C and D. Smaller modules can be inserted into mainframes designed for the larger sizes.

Electromagnetic compatibility is guaranteed in the VXIbus architecture by stringent requirements on how much any module can radiate into the adjacent modules and how much ac and RF current it creates on each of the power buses. Likewise, each module is required to meet its measurement specifications when subjected to radiation from other modules within the specified VXIbus limits, or powered within a specified voltage ripple range. If the sum of the dynamic module currents, which must be documented for each module in a data sheet, is less than the specified mainframe dynamic current capacity, then

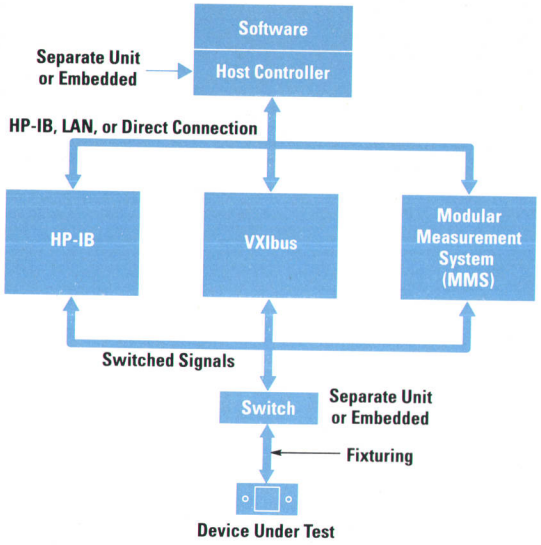


Fig. 2. The measurement system architecture for a typical computer aided test system consisting of HP-IB, VXIbus, and MMS instruments.

the system is guaranteed to be compatible. The mainframe is required to limit power supply voltage ripple on any of its power buses to be within the specified ranges, even when all modules are producing their maximum ac and RF currents. This combination of requirements ensures electromagnetic compatibility.

Finally, VXIbus adds autoconfiguration and communication protocols. All devices must have a set of registers available for control and status to perform self-test and initialization functions. Register-based devices have additional registers that access the device's specific functions. Message-based devices communicate using the VXIbus word-serial protocol (described later) to transfer ASCII commands and data between modules. Some devices include both modes of communication, using SCPI-based* ASCII commands for easy development and direct register access for higher speed. SCPI, register-based instruments, and message-based instruments are described in detail in the article on page 15.

This range of VXIbus functions provides a very scalable architecture in terms of size, power, performance, and cost for addressing a wide range of applications. The VXIbus standard is also upward compatible, that is, products that are designed to work with less mainframe resources in terms of size, connectors, or power will also work in mainframes that deliver more resources. This gives the user a breadth of products to choose from, making it possible to select only the performance needed for the present applications while preserving an upgrade path if future applications require more performance.

Measurement System Architectures

One of the most common applications of the VXIbus is testing an electronic product such as a printed circuit board. Fig. 2 shows the measurement system architecture of a typical functional test system. It consists of a computer, various instruments, and a switching system connected to the device under test (DUT). The computer can be any computer with an HP-IB or other interface, and is commonly called the host controller. The instruments can be any combination of three formats: HP-IB, MMS, and VXIbus. HP-IB instruments communicate through the HP-IB (IEEE-488.1, IEC 625) interface developed by Hewlett-Packard in the early 1970s. These are often bench instruments that supply an HP-IB interface port for command and control, and can be rack-mounted into a standard EIA rack cabinet. Many products, such as power supplies, will continue to be controlled primarily via the HP-IB. MMS, the Modular Measurement System, is another open architecture modular instrument system developed during the same time as VXIbus. Also maintained by a multimanufacturer consortium, MMS is a modular architecture optimized for RF and microwave applications, and works well in unison with HP-IB and VXIbus instrumentation and control.

The controller, though shown logically as a separate unit in Fig. 2, may actually be embedded in the VXIbus mainframe or in one of the other instrument architectures, allowing a direct connection between the instruments and the controller. Likewise, the switching unit may exist as an HP-IB unit or a group of VXIbus or MMS modules. Typically, HP-IB cables are used to interconnect the various instruments and mainframes, though some applications have used RS-232, LANs, and MXIbus** cables.

Fig. 3 shows a typical VXIbus system that is controlled by an external HP-IB controller. Each VXIbus instrument or instrument set (e.g., a set of switches) acts as an independent HP-IB instrument when controlled over an IEEE-488.1-to-VXIbus interface device installed as a module within the VXIbus mainframe. The VXIbus specification refers to this interface functionality as a 488-VXIbus interface device. In Fig. 3 this functionality is included on the command module. The command module owns one primary HP-IB device address, and accesses each VXIbus instrument or group of instruments through a unique HP-IB secondary address. HP-IB cables are used to connect to other external instruments and mainframes. Signals to the DUT are routed from the various instruments through electronic switches to an interface connector assembly (ICA) mounted on the rack cabinet. Typically, for each product to be tested by this system there will be a removable mating connector assembly that connects the interface connector assembly and the device under test. This fixture assembly is often called an interface test adapter (ITA) and is unique for each electronic product type tested on the particular tester. This allows one tester to test different products by changing only the interconnect test adapter and the software program that controls the instruments. ICAs and ITAs are described in detail in the article on page 52.

* SCPI means Standard Commands for Programmable Instruments.

** MXIbus is a flexible interconnect cable like the HP-IB, but represents the VXIbus data transfer bus on its conductors with only a small loss of performance. MXIbus has also been used as a high-speed link from VXIbus to an external computer and an extender link between VXIbus and the instrument mainframes.

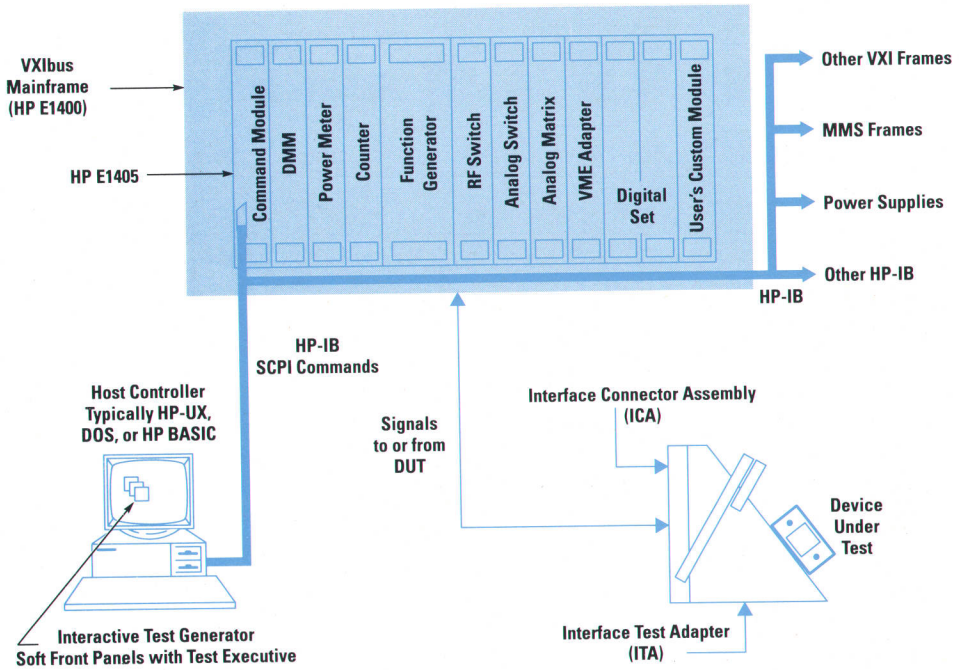


Fig. 3. A block diagram of a typical VXIbus test system using the HP-IB as the interconnection from the controller.

The test system in Fig. 3 is controlled by test executive software running on the external controller. Often, the developer will use the HP interactive test generator or a similar software package that displays soft front panels for faceless VXIbus instruments and the other HP-IB and MMS instruments. This software can be used to generate the ASCII commands for the instruments automatically, or the user may develop these I/O commands directly. The industry-standard SCPI language provides commands that a developer can use to control all instruments, regardless of the particular instrument architecture. In this example all VXIbus, MMS, and HP-IB instruments are programmed exactly as HP-IB instruments.

Fig. 4 shows an example of a system in which the controller is a VXIbus module embedded within the VXIbus mainframe. Since this configuration has an embedded controller that allows direct access to the VXIbus instrument modules, higher speeds are attainable via direct VXIbus communication. Also, because the controller is embedded, rack space is reduced. The register interfaces to these instruments can be directly mapped into the memory space of the controller, allowing extremely high-speed command and control. Often, an embedded controller will be operated in a diskless environment by interfacing through a LAN to a server. As in Fig. 3, a developer may choose to integrate VMEbus

(continued on page 10)

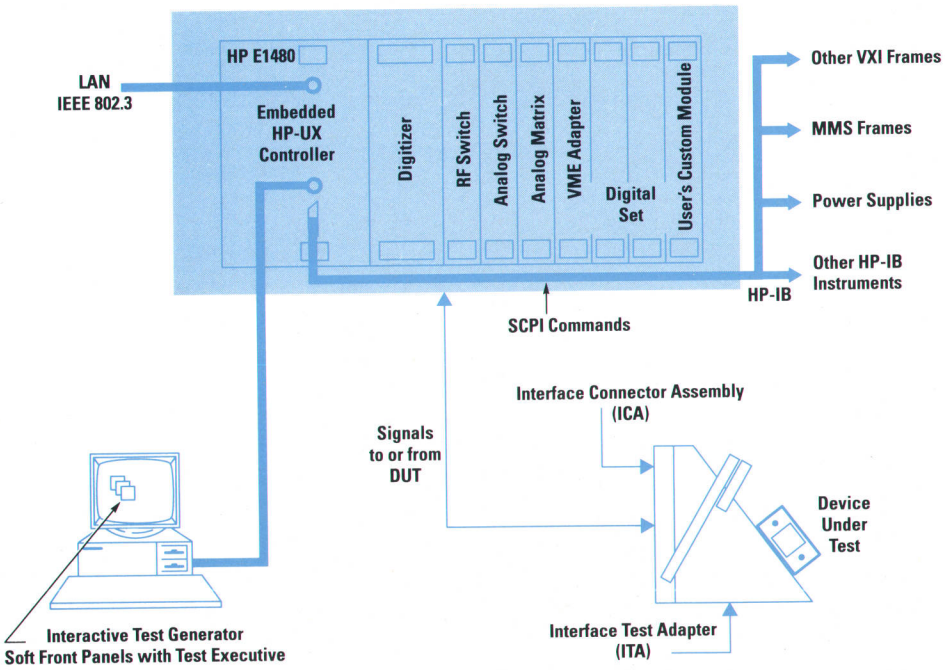


Fig. 4. A VXIbus system controlled by an embedded HP-UX controller.

The HP VXIbus Mainframes

The HP 75000 family of VXIbus products delivers a scalable offering of instruments, switches, and embedded controllers. The HP 75000 Series B mainframe (HP E1300) provides three A-size and nine B-size (two internal) slots, and a two-line display and keyboard (see Fig. 1). The mainframe can also be configured to include embedded hard and flexible disk drives. Command module functionality (HP-IB and resource manager) is permanently embedded in the mainframe along with two internal slots. The command module in the HP E1300 mainframe can be upgraded to an embedded controller with the IBASIC (Instrument BASIC) option. Series B products deliver a very cost-effective solution for cases in which moderate performance is needed.

(continued on page 10)

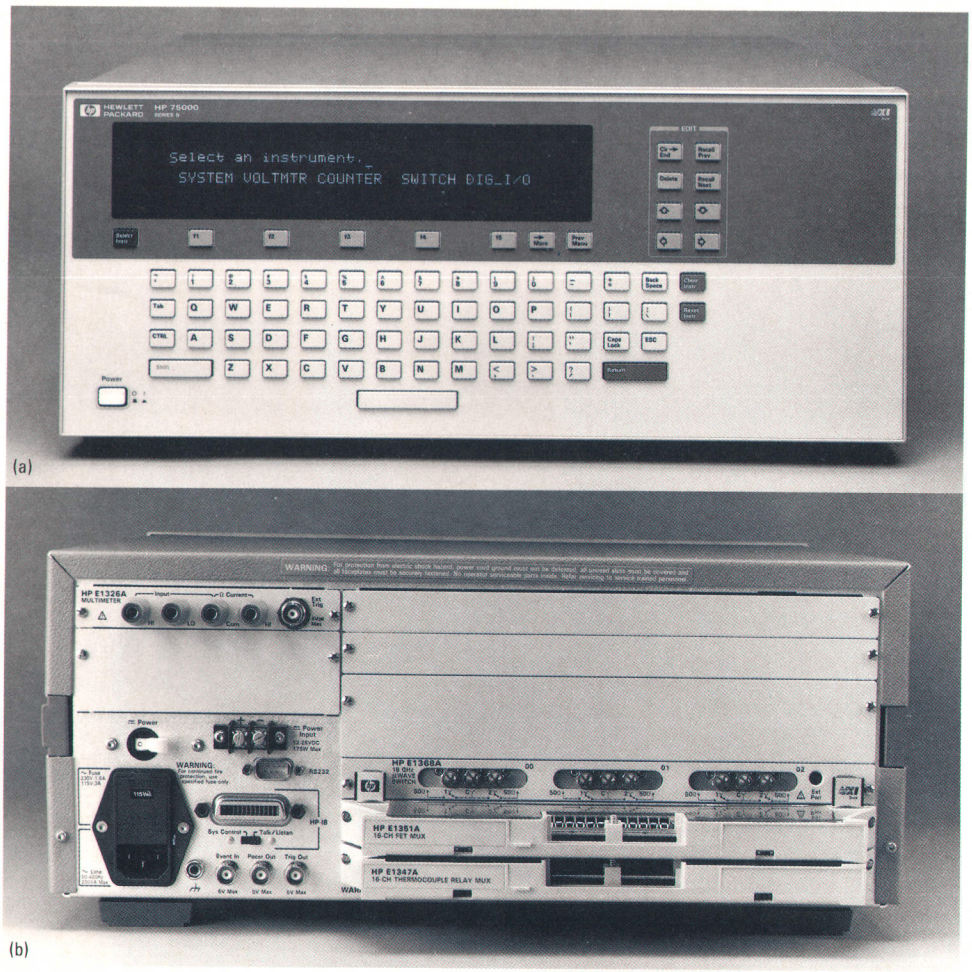


Fig. 1. HP 75000 Series B mainframe (HP1300 (a) Front view. (b) Rear view showing the location of the A-size and B-size slots.

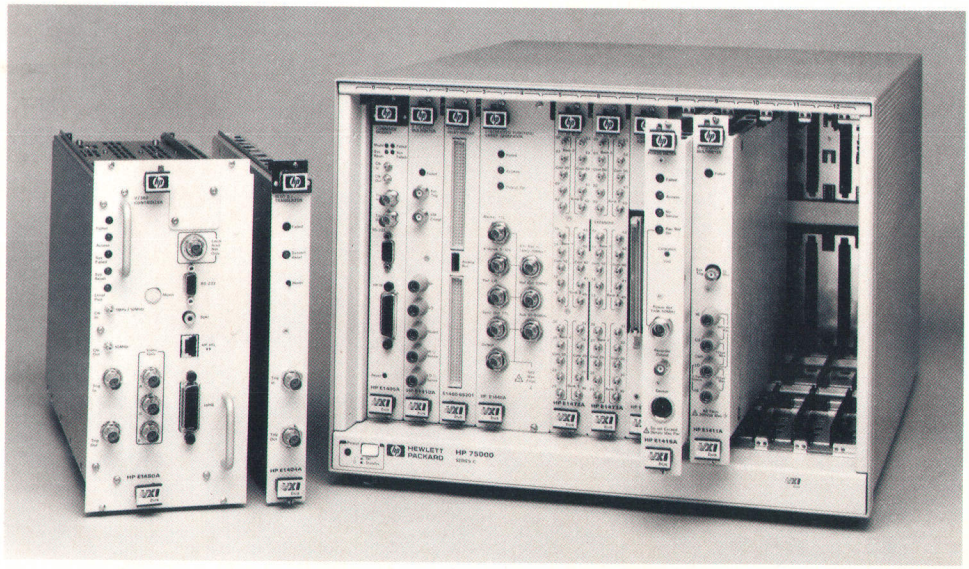


Fig. 2. HP 75000 Series C mainframe (HP E1400) and some of the C-size products

(continued from page 9)
The HP E1400 Series C mainframe (Fig. 2) provides a 13-slot C-size mainframe, an embedded HP-UX* workstation controller, a single-slot command module, and numerous high-performance instruments and switches. C-size modules are typically constructed within shielded enclosures and mount vertically within the mainframe. Module adapters are available that allow convenient insertion of A-size and B-size modules into the C-size mainframe. Series C products offer the highest-performance capabilities, yet can be intermixed with Series B products to attain the most economical system solution.

(continued from page 8)
modules into the system or develop custom VXIbus modules to create unique functionality that cannot be purchased as a standard product. If high-speed register access to additional VXIbus mainframes is desired, MXIbus cables can be used to interconnect to additional mainframes directly.

VXIbus Electrical Architecture

It is important to understand the basic internal VXIbus structures so that a system can be configured to take advantage of the VXIbus architecture. The VXIbus is a backplane bus composed of the VMEbus data transfer bus and various power, trigger, and clocking lines added by the VXIbus specification. For the most part, these lines are implemented as standard parallel buses, that is, conductors connect the same pin of each connector in every slot. For example, pin 1a of P1 in slot 0 is connected to the same pin in slot 1, slot 2, and so on. Two other topological structures exist in the VXIbus: local buses and star buses (see Fig. 5a). The widely-used VMEbus parallel bus structure has a memory-mapped architecture that can support data transfer rates of up to 40 Mbytes/s between any two modules. It also includes the interrupt bus structure and various system control and bus arbitration functions. The VMEbus data transfer bus can function using only the required P1 connector, although the P2 connector is required to use any of the 32-bit modes. VXIbus defines trigger lines on the P2 and P3 connectors that have parallel connections to all installed modules. This allows tight synchronization of instruments without any external cabling.

The second bus topology, known as the local bus, only connects adjacent slots of the backplane. On a B-size or C-size system, 12 very short lines connect from the right row of the P2 connector to the left row of the P2 connector in the adjacent slot. Thus, except for the slots on the end of the backplane, all slots have 12 local bus lines coming in on the left and going out on the right (see Fig. 5b). These bus lines can be used by manufacturers to connect signals or other private communication paths in one set of modules without interfering with or degrading the communications occurring in another set. Thus, the local bus can supply extremely tight coupling between modules where required, and complete isolation otherwise. Since the VXIbus allows a wide range of analog and digital signals on the local bus, there was a potential for electrical damage caused by accidentally plugging two incompatible modules into adjacent slots. This was solved in the VXIbus specification by requiring a unique local bus mechanical keying scheme on a module's faceplate

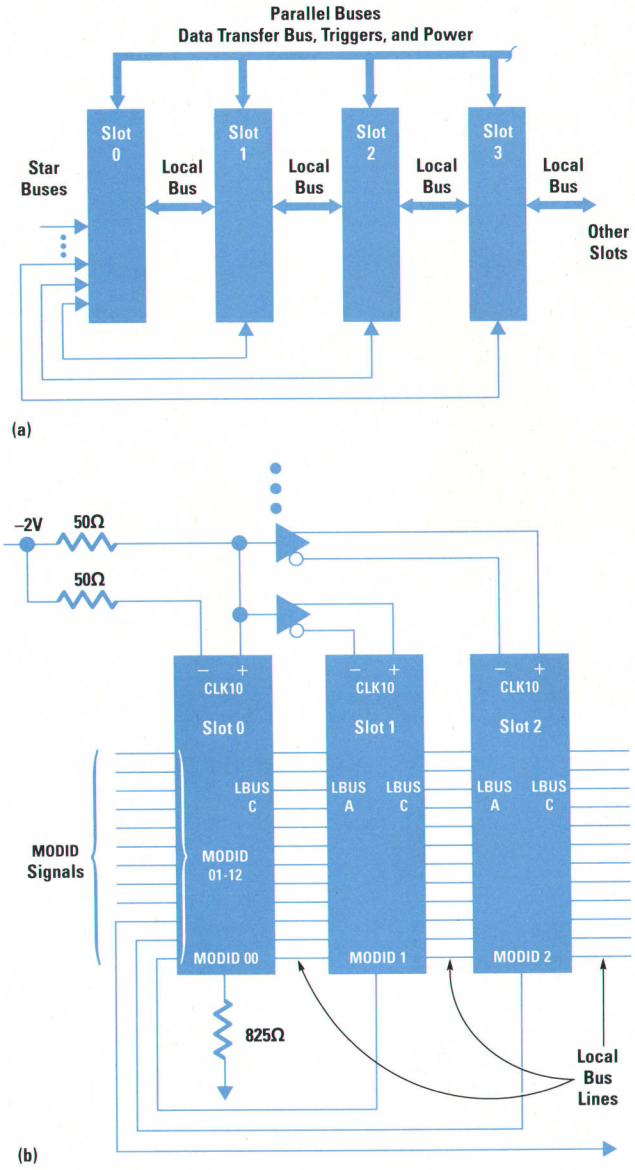


Fig. 5. (a) The parallel, local, and star bus structures used on the VXIbus backplane. (b) A detailed look at the signals available on the P2 connector.

that prevents modules with incompatible local bus signal types from being inserted next to each other.

The third bus topology used in VXIbus is known as a star bus. Here, signals are routed from one slot in the backplane to each of the other slots. A star bus structure is used on the VXIbus P2 connector to route a 10-MHz clock signal known as CLK10 from the leftmost slot to all of the other slots on the backplane (see Fig. 5b). The CLK10 signal is independently buffered to each slot on the backplane. Thus the loading characteristics of any slot have a negligible effect on the signal received at each of the other slots, allowing a very stable reference frequency with minimal phase jitter to be distributed to all modules. A derivative of the star bus structure is also used on the VXIbus P2 connector by routing a total of twelve different signals from the leftmost slot to each of the twelve slots to its right, with one signal going to each slot. These are known as MODID (module identification) signals and allow

the system to detect the absence or presence of a module in each slot, even if it is a failed or nonoperating module. MODID lines are also used to match each VXIbus logical device and address with its slot number. In the VXIbus, the leftmost slot is always known as *slot 0*. The modules that receive the CLK10 and MODID signals are sequentially numbered starting at slot 1 to the immediate right of slot 0, and may number up to slot 12. All slots with common CLK10 and MODID signals, and the slot 0 module they connect to, are collectively known as a VXIbus subsystem.

The above description highlights the uniqueness of slot 0 and the module inserted into it. Is slot 0 the only unique slot? Not necessarily. To support the data transfer bus, VMEbus requires that the leftmost module contain a system arbiter, generate the 16-MHz clock SYSCLK and drive the interrupt acknowledge chain. This functionality is known as the VMEbus system controller, and exists on the P1 connector. Since a VXIbus subsystem typically starts from the leftmost slot also, slot 0 and VMEbus system controller functionality are almost always combined on the same module and referred to collectively as slot 0. Technically speaking, this is not always the case. Since P2 and P3 are optional connectors, and only P1 is required by VXIbus, a system may be defined that has a VMEbus system controller but not slot 0. This may be implemented in low-cost systems such as the HP E1300A B-size VXIbus mainframe. In an uncommon but allowed architecture, a VXIbus subsystem begins with slot 0 to the right of the VMEbus system controller, allowing a few slots to be pure VMEbus slots to the left of slot 0. Here, slot 0 and the VMEbus system controller would be unique modules. However, in most cases, slot 0 functions and VMEbus system controller functions are combined in the leftmost slot.

VXIbus Logical Architecture

The VXIbus is based extensively on the VMEbus memory map. VMEbus defines 16-bit, 24-bit, and 32-bit address spaces that exist independently and simultaneously. They contain 64K bytes, 16M bytes, and 4G bytes of address space respectively. VMEbus also has 8-bit, 16-bit, and 32-bit wide accesses to these memory spaces. Each module presents a block of addresses that allows another device to access that module's functionality. A VMEbus master is a device that has control of the bus, and can read from or write to any address. A slave is a device that never has control of the bus, but has registers located within its address space that are accessed by one or more bus masters. There can be multiple bus masters in a system but only one can control the bus at any time. The bus arbiter on the VMEbus system controller module decides which device is granted control of the bus.

In a VMEbus system, a user typically has to create a memory map that shows the address blocks granted to each device so that one device's address space does not overlap another's. Next the user notes the starting addresses for each address block and configures jumpers on each module to set it up. Finally, the user finds the location and definition of any control and status registers and writes a driver so that after power is applied, one of the bus masters can access these registers on each device to check for proper system configuration, perform self-test, and then initialize the system. After initialization the user can proceed with the application.

In VXIbus, this is much simpler. VXIbus requires a standard definition of the control and status registers for all devices. Furthermore, the locations of all control and status registers are standardized. To implement this, the upper quarter (16K bytes) of the 16-bit address space is reserved for this information. It is split into 256 blocks of 64 bytes apiece. The lowest eight bytes of a block contain the control and status information, including the manufacturer identification and model code. The 256 blocks are numbered from 0 to 255. Each block number represents the logical address of a VXIbus device. Typically, a VXIbus device has switches that allow a user to set the logical address to a unique number, although a totally switchless configuration mode also exists. Thus, setting the address of a VXIbus module is very similar to setting the address of an HP-IB device, and the user can remain virtually unaware of how devices are actually using the VXIbus address space. A device is free to use the remainder of the 64-byte block for operational registers that access the device's functions. The article on page 41 shows how these 64-byte blocks are used for matrix relay addressing.

Of course, some device on the bus must still access each of the control and status register locations to see if a device is present, check the self-test results, and perform a proper system initialization. This functionality, typically a software or firmware program, is called the *resource manager*. The resource manager is defined to be at logical address 0, and is the only device allowed to access the bus immediately after power-up and system reset. It checks for the presence of a device at each logical address and will synchronize the operation of any slot 0 functions to identify the slot number of each device. If a device's function requires more than the 64 bytes of address space given to it in the 16-bit address space, the device will list how much additional address space is needed in its configuration registers, and the resource manager will assign the base address for this memory by writing to another configuration register. All this is done automatically without intervention from the user. Once the resource manager finishes its tasks, it tells other bus masters that they may now request use of the bus and operation can begin.

VXIbus Communication Protocols

VMEbus devices are typically controlled by accessing registers in the module's address space. This allows very high-speed, interactive control. However, HP-IB instruments are typically controlled by sending and receiving ASCII commands and numbers. Because of the finite time required by the instrument to interpret ASCII commands and translate between ASCII and binary numbers, this mode of operation tends to be slower than direct register access. However, the speed of the measurement may limit system speed, so ASCII communications would not present a speed bottleneck, but could deliver a significant improvement in ease of use. This is particularly true if the ASCII commands conform to industry standard SCPI so that there is a single set of ASCII commands to

control instruments, and a single message exchange protocol to exchange data and status.

VXIbus supports both of the above communication methods. If a device only supports register accesses, it is known as a register-based device. If it supports ASCII communication using the VXIbus-defined word-serial protocol, it is called a message-based device. Message-based devices may also support direct register access for cases in which increased speed is important.

Word-Serial Protocol

The word-serial protocol used by message-based VXIbus devices provides a standard way to send and receive ASCII messages between devices of different manufacturers. This is implemented by defining additional communication registers next to the configuration registers in the 16-bit address space and standardizing their operation. Essentially, 16 bits can be sent at a time, with data represented in the lower 8 bits, and additional control information in the upper 8 bits.

There is a one-to-one correspondence between any HP-IB operation and its equivalent word-serial operation. This is a key feature of the VXIbus. Because of this equivalence, it is possible to build a device that translates from the HP-IB at one end to VXIbus word-serial protocol at the other without any knowledge of the function of the ASCII commands being sent. This allows a user to program VXIbus instruments from any computer that has an HP-IB interface as if they were HP-IB instruments. A translation module used to translate between the two protocols is known as a 488-VXIbus interface device. A typical method of translating between HP-IB addresses (32 primary addresses with 32 secondary addresses each) and VXIbus logical addresses (256 total) is to use the HP-IB secondary addressing feature. This can be done by assigning each mainframe's 488-VXIbus interface device an HP-IB primary address and each embedded VXIbus instrument a unique secondary address. The five most-significant bits of the VXIbus logical address are used to determine the HP-IB secondary address. In fact, assuming the three least-significant bits are already set to zero, setting the five most-significant bits can be exactly the same procedure as setting the five address bits for an HP-IB instrument. Since the three least-significant bits are set to zero, a single-module instrument always has a logical address that is divisible by eight. For a single instrument composed of several modules, the modules can be set to consecutive logical addresses, but controlled from a single secondary HP-IB address. This allows vendors to deliver naturally modular instruments, such as switches or scanning voltmeters, that operate as a single instrument. If a resource manager is included on the 488-VXIbus interface module, it will be found at secondary address zero since, by definition, it must be set to logical address zero.

HP-IB translation works well for communicating with message-based devices, but something else is needed for register-based devices. A common method for communicating with register-based instruments using ASCII commands is for a manufacturer to include the instrument's ASCII interpreter firmware on a 488-VXIbus interface device. An instrument's interpretation firmware is often referred to as its driver. At power-up, the HP-IB interface device would recognize which modules are register-based and which modules it has drivers for, and invoke the driver whenever that instrument's equivalent HP-IB secondary address is accessed. Using this method, register-based modules behave as message-based modules, and they both appear as independent HP-IB instruments when accessed from a 488-VXIbus device. If no firmware driver exists for the module, it can still be accessed by commanding the resource manager to perform the proper register reads and writes.

Since VXIbus is a multiple-master architecture, there must be some method of determining which slaves belong to which masters. For this reason, VXIbus has created a commander-servant hierarchy. A commander is a device that controls other devices. The devices it controls are called its servants. A VXIbus device can only have one commander, but it can have any number of servants. Since only bus masters can control the bus, a commander must be a master, and slaves can only be servants. A commander can also be a servant to another commander, forming a hierarchial command system. VXIbus has the additional requirement that commanders must be message-based devices. This allows the resource manager to configure the commander-servant hierarchy with commanders from multiple vendors using standard word-serial commands.

The commander-servant hierarchy is determined by the resource manager after power-up. Each commander or potential commander has an attribute called its servant area. This is a number, which the user typically sets by a switch, that determines how many logical addresses a device may own as servants. For instance, if a device is at logical address 24, and has a servant area of three, then it has exclusive control over devices at addresses 25, 26, and 27. If there are other commanders at these addresses, the commander at address 24 can control those also, but not those commanders' servants. This allows hierarchial command structures.

This feature can be used to make register-based devices appear message-based by configuring a commander to perform the ASCII command translation and register access for a set of register-based instruments. A user can control the set of modules by communicating with the commander using ASCII messages, while the commander performs the register accesses required. If a particular commander doesn't translate for all the modules in a given system, and there is another commander in the system that translates for the remaining modules, both commanders can be configured in the same system without conflict by setting up the commander-servant hierarchy correctly.

VXIbus Terminology

The following definitions should be helpful in understanding the terminology used here and in other VXIbus articles in this issue.

Binding. The process of associating a firmware function (like a VXIbus device driver) with a particular piece of hardware.

Command Module. This module contains the intelligence (processor) that controls register-based devices, the human interface, and RS-232 resources. In addition, this module may contain either the VXIbus slot 0 or datacomm functionality.

Console. The human interface port that is used by the system to report power-on and diagnostic information to the user.

Datacomm. This is the port through which an external host (computer) can control the mainframe. An example of this type of port is the HP-IB connection, which is used by the host to send commands to and read data from the instruments in the mainframe.

Device. A component of a VXIbus system. Normally a device will consist of one VXIbus module that occupies one slot. However, multiple-slot devices and multiple-device modules are permitted. Some examples of devices are computers, multimeters, switches, operator interfaces, and counters.

Driver. A firmware or software program that controls a device.

Host. This is typically a computer that contains the test system executive software and has an HP-IB connection to the test system. However, the host can be an embedded controller in the instrument mainframe (e.g., the HP E1480A V/360A embedded controller).

Human Interface. This is the path through which a user interacts with the mainframe. This path can be implemented via different hardware interfaces such as the front panel of the HP E1300 B-size mainframe, or a terminal connected to the mainframe via an RS-232 port.

Instrument. An instrument is a collection of hardware and firmware functions that are addressable by the host as a complete unit. An instrument has its own HP-IB address and input and output buffers, and the ability to share system resources such as the human interface with other instruments. Some instruments have on-board intelligence (message-based instruments) and some require support from a command module for intelligence (register-based instruments).

HP Instrument BASIC (IBASIC). A subset of HP BASIC that can reside inside the mainframe and has the ability to control instruments and other resources in the system such as the display and keyboard.

Mainframe or Cardcage. The VXIbus cage that contains VXIbus modules.

Module. This is typically a printed circuit board assembly and its associated mechanical parts, front panel, optional shields, and so on.

Peripheral Control. When the mainframe or one of its instruments is controlling an external device such as a disk, plotter, printer, or another instrument, it is being used to do peripheral control. Note that the HP-IB, RS-232, or other physical communication paths can be used to control peripherals, but this functionality should not be confused with datacomm. One other important note is that peripheral control is only supported via certain IBASIC commands.

Rack-and-Stack Instrument. This is a traditional stand-alone instrument that consists of a cabinet, instrument functionality, power supply, user interface, and computer interface.

Resource Manager. This is a VXIbus device located at logical address 0 (sometimes the command module or embedded controller), which provides configuration management services such as A24 and A32 address space configuration, commander/servant configuration, power-up self-test, and diagnostic test management.

VXI-OS. A set of software services that reside on top of the pSOS™ operating system. These services consist of interrupt handling, shared data space management, and operation of specific hardware features of the command module that enable instrument drivers to implement the two-task model for handling instrument and user interface I/O (see article on page 29).

pSOS is a U.S. trademark of Software Components Group, Inc.

Since the slot 0 module is a special device to begin with, it is convenient to place all system resources on this module. This includes slot 0, the VMEbus system controller, the resource manager, the 488-VXIbus interface device, and the instrument drivers for any register-based instruments made by the same manufacturer. This combination device will be referred to as a *command module* in this series of articles. There is no industry standard term for this common module, and it is sometimes referred to by one of its functions such as slot 0 or resource manager.

Fig. 6 shows the alternate communication paths allowed in a VXIbus system and the role of the command module in this system. If the controller is an external computer, it will typically interface through HP-IB cables since no direct VXIbus path will be available. The instruments will be controlled using ASCII commands, typically complying with the SCPI standard. If the instrument is a message-based device, instrument commands will be routed through the command module directly to the message-based instrument for interpretation and execution. If the instrument is register-based, the command module will intercept and interpret the command and then perform the proper binary register accesses to the device to execute the function. From the controller, both device types connected to the command module appear to be SCPI message-based devices.

If an embedded controller is used, it can still communicate over its HP-IB control port to the command module, or it can control any or all devices directly from its VXIbus interface. Contrary to many preconceptions, controlling a message-based device directly from an embedded computer will deliver very little or no increase in speed. This is because ASCII command interpretation and number building result in delays much greater than bus handshake time and thus speeds are kept comparable to HP-IB communication. However, accessing the hardware registers directly on a register-based device can

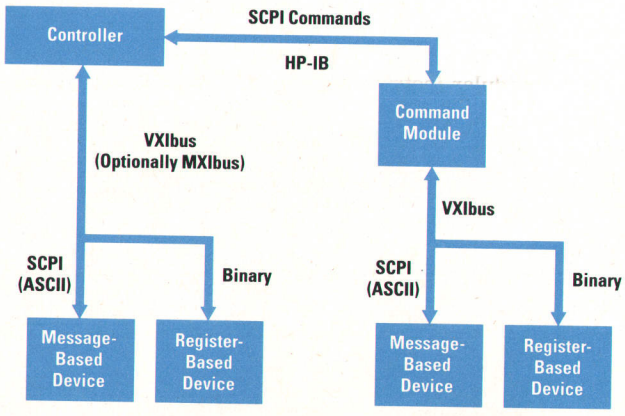


Fig. 6. Alternate communication paths allowed in a VXIbus system.

provide dramatic increases in continuous and interactive throughput, sometimes approaching three orders of magnitude. Therefore, the highest-speed path in a VXIbus system is direct access to the internal registers of a device, and is the only path that can consistently exceed HP-IB speeds.

The recently introduced MXIbus interface allows external controllers access to VXIbus registers by mapping the VXIbus data transfer bus onto a flexible interconnect cable. This allows external controllers to access the high-speed VXIbus architecture with only a modest degradation of total system performance compared to an embedded controller. This modest throughput degradation is still much higher than can be expected from the HP-IB.

Many users choose SCPI ASCII commands to develop their test systems initially, and then fine tune the identified bottlenecks by accessing registers directly where needed. Some message-based devices also support register access for functions in which speed is important. These may be registers in shared memory space that hold blocks of binary data for fast continuous data transfer, or registers that access the instrument's configuration hardware whenever high, interactive throughput is desired.

Summary

The VXIbus offers a high degree of flexibility in module size, bus structure, and communication protocols for high-performance instrument systems. Instrument vendors, system integrators, and end users have unprecedented opportunities to address a wide variety of applications using the VXIbus architectural features. By using the HP-IB as an interface, users can mix and match VXIbus products with modular measurement system (MMS) modules and HP-IB instruments to perform an even wider range of measurements, while retaining a single model of interfacing with the instruments. This migration path from the HP-IB, the compatibility with MMS and HP-IB architectures, and the high-performance capabilities of the VXIbus architecture promise to make VXIbus a widely used architecture.

References

1. K. Jessen, "VXIbus: A New Interconnection Standard for Modular Instruments," *Hewlett-Packard Journal*, Vol. 40, no. 2, April 1989, pp. 91-95.
2. *Ibid.*, p. 92.

HP-UX is based on and is compatible with UNIX System Laboratories' UNIX* operating system. It also complies with X/Open's* XPG3, POSIX 1003.1 and SVID2 interface specifications.

UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

The VXIbus From an Instrument Designer's Perspective

HP has defined a set of internal standards to compensate for some missing aspects of the VXIbus standard that are critical to instrument design.

by Steven J. Narciso and Gregory A. Hill

The VXIbus specification defines a technically sound modular instrument standard that addresses electrical, mechanical, EMC/power, and communication requirements for instrument modules. Although extensive, the VXIbus standard does not cover certain information that is critical to an instrument's design. For this reason, it becomes necessary for a VXIbus instrument manufacturer to define an internal standard to supplement the VXIbus specification.

The internal standards developed by HP enable instrument designers to provide customers with a consistent "look and feel" and functional interoperability. These standards also provide designers with common hardware interface and firmware design guidelines so that they can focus on specific instrument features and not on system design.

HP has concentrated its internal standardization efforts in four areas: a common instrument language, hardware interface, soft front-panel design, and industrial and mechanical design. This paper will cover these areas and other supplemental HP standards.

SCPI: A Common Instrument Language

Until recently a test engineer had to deal with a variety of instrument programming languages and formats. Data was transferred in almost every format imaginable, from ASCII to device dependent compacted binary. When a new instrument was introduced, the engineer could look forward to rewriting the majority of the instrument driver code, even if the new instrument was a direct replacement from the same manufacturer.

In 1987, IEEE standard 488.2 was approved. This standard provides a level of standardization in message and data formats, message exchange protocols, and common commands to be shared among a wide range of devices. This standard allows computer languages to speak in a consistent manner to instruments and enables higher-level instrument test executives to be written.

IEEE 488.2 defines a few common commands that address the housekeeping functions of the instrument. The standard intentionally does not define commands that are necessary for measurement or signal configuration. This is the area that the industry's Standard Commands for Programmable Instruments (SCPI) addresses. SCPI is used by all of HP's VXIbus modules.

Originally HP's Test and Measurement System Language, SCPI builds upon existing standards wherever possible, including IEEE standard 488.2 and IEEE Standard 754. The result is a language that is orthogonal, powerful, extensible, and yet familiar and friendly to users. The language never refers to instrument types, only to instrument functions. Therefore, SCPI applies to a wide variety of instruments, from power supplies to network analyzers. We refer to this consistent coverage of all instruments as horizontal compatibility. The language also offers vertical compatibility, or consistent coverage of succeeding generations within an instrument family.

Mnemonics

SCPI uses a formal set of truncation rules to generate mnemonics. The rules are stated as follows:

- If a keyword is four characters or fewer, the keyword itself is the mnemonic (e.g., AUTO, ON, OFF).
- If the keyword is longer than four characters the mnemonic is the first four characters of the keyword (e.g., OUTPut).
- If the resulting mnemonic ends with a vowel, the mnemonic is truncated to three characters, dropping the trailing vowel (e.g., ATTenuator).
- If a phrase is used instead of a single word, the keyword uses the first character of the first words, followed by the entire last word (e.g., Kaiser BESsel).*

Hierarchical Structure

Keywords are joined to form a compound header element consistent with IEEE 488.2. This produces a hierarchical language. Thus, to set the input attenuator the command would be INPut:ATTenuator. There are several reasons for choosing a hierarchical structure. First, a four-character mnemonic becomes quickly overloaded, even within a single instrument. Collisions often occur that must be resolved by violating the mnemonic generation rules. By making the language hierarchical, keywords can have meaning within the context of their tree position, and collisions effectively disappear. For example, to set an output attenuator, the command would be OUTPut:ATTenuator. Both the input and output subsystems

* The four character rule restricts the keyword to KBES.

use the keyword ATTenuator without conflict because they appear in different places in the tree.

Default Node

Some functions are executed more often than others, and the most common functions need to have short and easy-to-use keywords. SCPI accommodates this by placing default nodes at critical points in the tree. Default nodes do not need to be sent as part of the command. For example, there are several commands associated with output conditioning such as attenuator, filtering, offsets, and output enabling. We determined that output enabling was the most commonly used function. Therefore, we added the command OUTPut[:STATe], which enables the instrument's output. The application can send either OUTPut:STATe ON or simply OUTPut ON to enable an output.

Default nodes play an equally important role in allowing the language to be extensible. For example, if we had a command in the output section called FILTer, which enabled a low-pass filter, and we wished to add a command that enabled a high-pass filter, a conflict would exist. The new command creates a further qualification of the keyword FILTer. To add the new command, one would add a default node to the existing command under filter, describing the additional qualification. Thus, the existing command would become OUTPut:FILTer[:LPASs]. We could then add the new command as OUTPut:FILTer:HPASs. It is important to extend by adding default nodes because the default node allows applications written for the old instrument to continue to work with the new extended capability instrument. The application can still send OUTPut:FILTer and expect it to function the same as it did before the extensions were made.

Standard Parameter Forms

SCPI provides several standardized parameter forms and discourages the use of forms that do not conform to the standard. These standardized forms include:

Numeric Parameter. In addition to accepting ASCII numeric input, the numeric parameter also accepts several data approximations. For example, INFinity is represented in SCPI as 9.9 E37, and the command NAN, or "not a number," is represented as 9.91 E37. All numeric parameters must also accept MAX and MIN which represent the maximum and minimum values that the function can be set to, while giving consideration to other settings within the instrument.

MIN and MAX are also required as parameters to the query form of a command. This form allows the application to determine the maximum and minimum legal values that can be sent without error. For example, if an input attenuator can be set to values of 0 to 70 dB, then for the query INPut: ATTenuator?, MAX would return the value 70. Attempting to set a value greater than 70 would result in an error. Sending any value within the range of 0 to 70 dB would cause the attenuator to be set to the closest available value. For example, if the attenuator had steps of 10 dB, receipt of the command INPut:ATTenuator 38 would cause the attenuator to be set to 40 dB, with no error generated.

SCPI requires the use of numeric parameters whenever a parameter can be expressed as a number. Qualitative terms such as FILTer:CUToff LOW | HIGH are not allowed because one instrument's LOW is another's HIGH.

Boolean Parameter. A Boolean parameter accepts ON, OFF, 1, or 0 as values. ON and 1 are aliases as are OFF and 0. The query form returns 1 and 0 to be compatible with the values returned by the IEEE 488.2 common commands.

Discrete Switch Parameter. The discrete switch parameter allows for selection among several different options. These selections are specified as character data. An example would be

```
TRIGger:SOURce INTernal | EXTernal | BUS | IMMediate
```

where | indicates alternate selections.

Parameter Couplings. When a function can be coupled to another function, SCPI controls the coupling through the AUTO keyword. Turning AUTO on allows the function's value to be automatically determined by other settings in the instrument. AUTO has the parameters ON | OFF | ONCE. ONCE is an event that has the effect of turning AUTO ON and then OFF, thus allowing the value to be automatically determined and then frozen at that value. The AUTO node is a subnode of the function, producing an unbalanced tree like that in the following example:

```
SENSE
  :ATTenuator <value>
  :AUTO <Boolean> | ONCE
```

Minimization of Parameters

Most commands that control instrument settings have a single parameter associated with them. When more than one parameter is specified, the function tends to become overloaded and confusing to use. When the two parameters are broken into separate commands, the use becomes more obvious and easier to learn. An example is a proposed command that sets a power level and controls the output port. The proposed form is POWer:LEVel 5 W, ON. This is broken into two commands, one that sets power level (POWer:LEVel 5 W), and one that controls the output switch (OUTPut:STATe ON).

Signal-Oriented Measurements

When making measurements, the particular type of hardware has traditionally controlled the specific measurement technique. This meant that the application had to know how the measurement was being performed in terms of the hardware. The U.S. Air Force's MATE/CIIL program was the first to recognize that measurements should be performed in terms of the signal being measured and that the instrument should perform the hardware setup necessary to make that measurement.

A signal measurement is performed through a set of layered commands. At the highest level the MEASure command is used. This command configures the instrument, triggers acquisition of data, and then processes the data into final form. Thus, sending the command MEASure:VOLTage:PERiod? would return the period of the voltage signal.

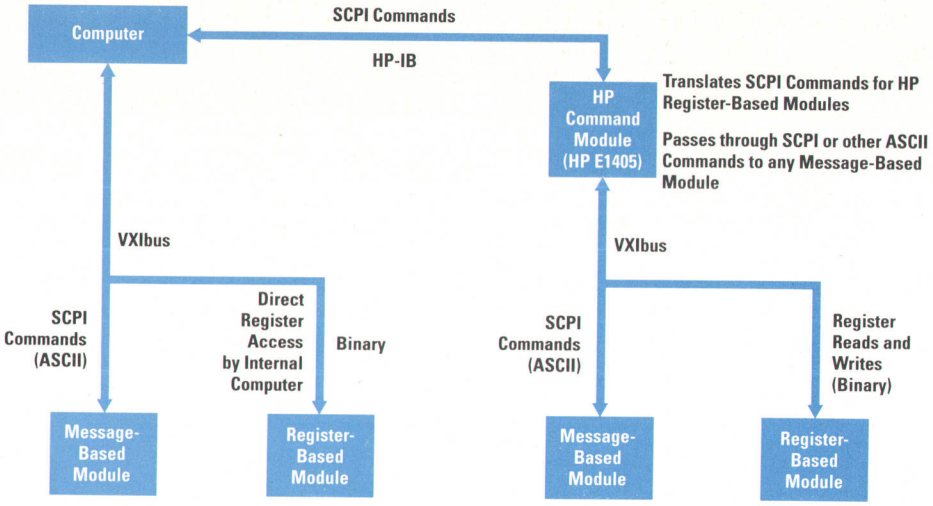


Fig. 1. Alternate communication paths for VXIbus instruments.

There is a need to be able to fine tune the measurement beyond what the instrument can provide. Therefore, the MEASure command has two additional associated commands, CONFigure and READ. The purpose of CONFigure is to perform the static setup of the measurement. The READ command performs the data acquisition and postprocessing functions. The justification is that the application is able to configure the measurement, make small device-dependent adjustments to the setup, and then complete the high-level measurement process through the READ.

The READ is further broken down into INITiate, which causes acquisitions to occur, and FETCh, which performs the postprocessing. This allows several postprocessing operations to occur on the same acquired data. This feature is especially valuable when the acquired data is nonperiodic.

Parameters to all of these commands are described in terms of the desired measurement. Ranges are avoided, and replaced with expected value parameters. Accuracy parameters are expressed in absolute values as opposed to a percentage of range. For example, MEAS:VOLT:DC? 5, .001 configures the instrument to a range capable of 5 Vdc with an accuracy of 1 mV.

A Common Hardware Interface

Hewlett-Packard bases its VXIbus instruments on two different hardware interfaces: message-based and register-based. Whenever space and cost allow, the message-based interface is used because it allows the ultimate freedom in interoperability and independence of commanders* since the instrument's language is contained within the module. For other instruments the register-based design provides small size and low cost. Both interfaces are VXIbus revision 1.3 compatible.

The HP E1405 command module (which also provides slot 0 functions) contains the SCPI language processor for all HP register-based modules and allows transparent IEEE 488.2 or SCPI control of register-based and message-based modules. Addressing is identical to that used for rack-and-stack instruments in that there is a primary address for each VXIbus mainframe and there are secondary addresses for each instrument. Module combinations, such as HP's VXIbus switches, can be configured together as a single instrument. Embedded computers may speak directly on the VXIbus using SCPI for message-based modules and register-level instructions for register-based modules. They may also speak to register-based modules with SCPI via the HP-IB and the command module. Fig. 1 shows these communication paths for VXIbus instruments.

The firmware for both message-based and register-based instruments is based on a generic command parser design which is IEEE 488.2 compatible. For the VXIbus interface code in a message-based instrument, the parser and the execution routines all reside in the instrument module. The register-based instruments have their corresponding code in the HP E1405 command module or the HP E1300/1A mainframe's built-in commander. This multitasking design allocates each HP register-based instrument its own task, allowing transparent programming of each as an individual instrument, while substantially reducing overall cost because the CPU is shared by all register-based modules. The article on page 29 describes the multitasking scheme for VXIbus command modules.

The Message-Based Interface

The purpose of the generic HP message-based interface is to control the operation of the instrument's hardware through commands received over the VXIbus directly from the controller. These commands generally take the form of IEEE 488.2 ASCII strings using VXIbus-defined word-serial protocol (see page 12), but can also be simple register reads and writes. The ASCII strings allow the user to program the instrument module in the same manner as its rack-and-stack equivalent. The simple register reads and writes allow higher-speed communication for tasks such as moving large blocks of data between the instrument and a computer.

One of the promises of VXIbus is that by designing to a common architecture, instrument designers can focus development efforts on the instrumentation functions instead of on the digital interface, and therefore be able to release new products with a lower up-front investment

* A commander is a device that controls other devices on the VXIbus (see page 12).

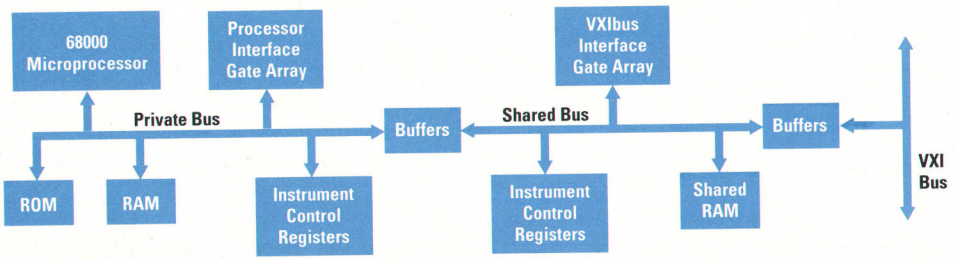


Fig. 2. HP message-based instrument block diagram.

and shorter development time. Achieving this goal is not an automatic by-product of choosing VXIbus as a common platform. It requires commitment to a single instrument architecture from top to bottom. HP's family of message-based instruments is based on such an architecture. Starting with the HP E1410A digital multimeter, every message-based instrument has been built around a common core. This core includes a high-performance SCPI parser, a flexible real-time operating system, and common VXI driver code, all running in a common hardware environment.

At the foundation of this architecture is a 68000 microprocessor and two gate arrays, which guarantee that the common functions are identical within the various instruments (see Fig. 2). With this hardware core, designers can easily concentrate design efforts on the instrumentation functions, instead of on the microprocessor, VXIbus interface, and common firmware.

The Processor Interface. One of the gate arrays provides resources common to most microprocessor-based instruments. In a single 124-pin package, it performs the work of more than 250 SSI and MSI integrated circuits. Its functions include address decoding, interrupt conditioning, various timing functions, and single-bit I/O.

This gate array defines a standard memory map, as shown in Fig. 3. It has separate output pins for selecting ROM, private RAM, shared RAM, the VXIbus, several local peripheral devices, and a 32K-byte address block. It generates the appropriate VXIbus signals for accessing the A16 (64K-byte) and A24 (16M-byte) addressing modes as well as executing VXIbus interrupt acknowledge cycles.

There is signal conditioning for 16 different interrupt sources, including three internal timing sources. Each interrupt can be programmed to one of seven priority levels. When an interrupt is asserted, the gate array drives the appropriate priority code onto the microprocessor's three IRQ lines. The microprocessor then executes an interrupt cycle, in which it fetches a vector from the gate array. This vector indicates which of the various sources is interrupting. To provide complete flexibility, each of the interrupt inputs can be programmed to select the signal polarity, and whether it is edge or level sensitive (see Fig. 4).

The processor interface gate array also includes a pacer function. When triggered either by software or an external signal, the pacer generates a series of pulses. The number of pulses and the pulse frequency are both programmable as 24-bit values.

VXIbus Interface. The other gate array provides the interface to the VXIbus. This gate array contains the VXIbus configuration and communication registers, VXIbus access control, and support of shared RAM. It resides on a shared bus between the microprocessor and the VXIbus, as shown in Fig. 2. The shared bus is accessible to both the onboard microprocessor and external VXIbus devices.

The configuration registers are the key to identifying a device within a VXIbus system. In addition, they provide a standard mechanism for reporting device self-test status and assigning device addresses. The communication registers provide the mechanism for SCPI communication between a commander and its servants.

The VXIbus interface gate array supports the full set of message-based capabilities defined by the VXIbus specification including the various word-serial protocols, the shared-memory protocol, static and dynamic configuration, shared A24 or A32 RAM, and VXIbus master capability.

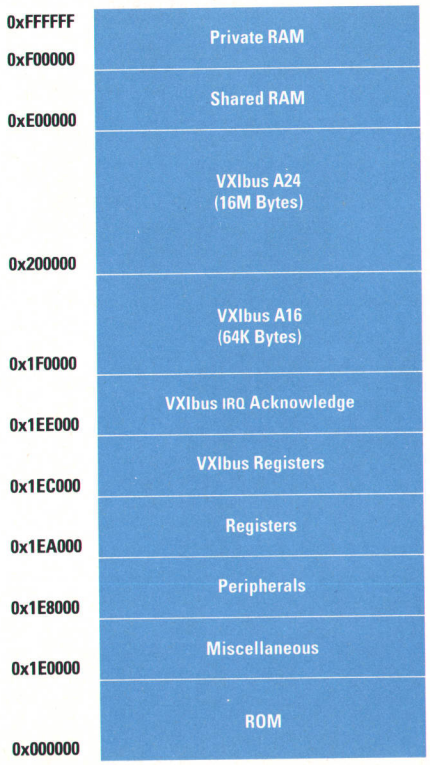


Fig. 3. Standard memory map for HP message-based devices.

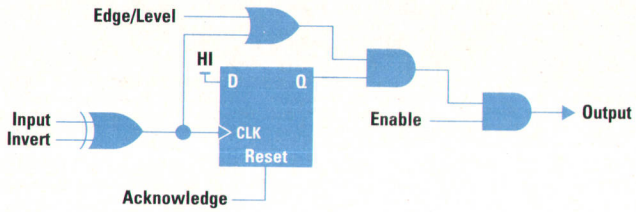


Fig. 4. Interrupt signal conditioning.

The bus access control capability supports eight different access modes. The onboard microprocessor can directly access the gate array registers or the shared RAM. Any VXIbus master can access the gate array's registers or the shared RAM. The onboard microprocessor can access other VXIbus slaves,* and it can access the gate array or shared RAM as VXIbus slaves. In addition, the gate array itself can initiate VXIbus transactions.

To reduce bus traffic, the gate array can be programmed to send a signal to the device's commander to indicate certain events. This mode allows the commander to attend to tasks other than polling, which is otherwise required. Similarly, the gate array can interrupt the local microprocessor to notify it of certain events, including the arrival of a word-serial command.

The gate array also implements hardware decoding of common word-serial commands. This feature increases command processing speed by eliminating the need to interpret word-serial commands in firmware.

Direct Register Access. HP's message-based devices are normally controlled via SCPI commands passed through the VXIbus interface. The onboard microprocessor interprets these messages and then executes a series of reads and writes to and from the instrument's internal control registers shown in Fig. 2. Some of HP's message-based instruments provide a more efficient mode for critical high-speed operations. In these instruments, the internal control registers are accessible from the VMEbus (they are interfaced to the shared bus much like the shared A24 or A32 RAM). This architecture allows a VXIbus controller to control the instrument's operation directly without the overhead of SCPI command transmission and parsing. In effect, these instruments can be operated as register-based devices when the application demands the highest possible speed.

The Complete Family. This common architecture has been implemented in more than 15 different HP message-based instruments. Some of these instruments include:

HP E1405 Command Module
HP E1410A 6-1/2 Digit Multimeter
HP E1445A Arbitrary Function Generator
HP E1416A Power Meter**
HP E1440A Synthesized Function/Sweep Generator**
HP E1420B Universal Counter
HP E1426A Digitizing Oscilloscope**
HP E1428A Digitizing Oscilloscope
HP 75000 Series 90 Modular SONET/SDH Analyzer

* Since the processor can access and be interrupted by other VXIbus modules, it can function as a commander and as a servant of a commander.

** See "Examples of Message-Based VXI Instruments" on the next page.

Register-Based Interface

The generic HP register-based interface is controlled by simple register reads and writes, which allow high-speed communication to the instrument. ASCII strings can also be used to program all HP register-based modules through the SCPI language processor in the HP E1405 command module or the HP E1300/1 mainframe.

HP's register-based devices all have a simple, consistent interface to the VXIbus backplane. Fig. 5 shows a block diagram of a typical register-based interface. The interface is ideally suited for any low-cost instrument application having straightforward control requirements. To simplify the firmware development tasks, these interfaces have been designed to have a consistent "look and feel" among similar types of instrument modules. Each register-based instrument includes the required VXIbus configuration registers as well as the device dependent registers needed for instrument control. The key to both the low cost and high speed of these devices is their simplicity. Many functions, such as closing a relay, can be executed by just a single register write from the VXIbus.

The interface hardware consists of some address and data buffers, an address decoder, a small gate array, and the various data registers. The gate array takes care of all the timing details for the various handshake and control signals between the VXIbus and the device's registers. The interface functions as a VXIbus servant, and provides the device with the capability of generating interrupts on the VXIbus. This interface is used in more than 30 assorted HP Series B and Series C modules including low-cost multimeters, counters, digital-to-analog converters, and switches. It is also offered in both Series B and Series C breadboard modules.

The Model D20 digital functional test system described on page 59 uses register-based interface hardware to achieve the high I/O speed required to move large blocks of digital patterns to and from the computer.

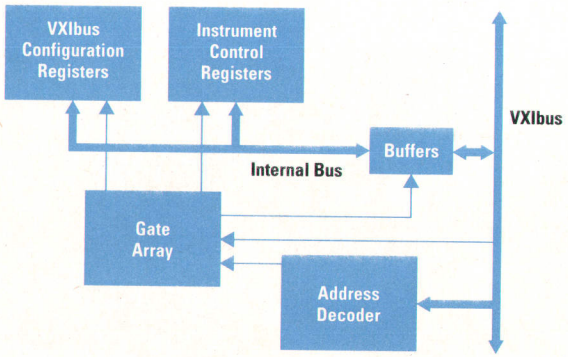


Fig. 5. Register-based interface.

Examples of Message-Based VXI Instruments

To incorporate the capabilities of existing non-VXIbus instruments into modules that could fit into VXIbus mainframes, the designs and features of some instruments were reengineered to make them into VXIbus message-based instruments. The following sections describe the design considerations that went into creating these instruments and the benefits derived from their VXIbus implementation.

HP E1426A 500-MHz Digitizing Oscilloscope

To create the HP E1426A, engineers leveraged technology developed for the HP 54503A 500-MHz digitizing oscilloscope. This technology provided an excellent platform to implement a four-channel, 500-MHz, high-accuracy acquisition system in a two-slot VXIbus module. A high level of integration created all the elements necessary to implement the acquisition system on one board and the control and interface circuitry on the second board. The scope functions were integrated using the following HP semiconductor processes:

- Preamp: HP-5 integrated circuit process
- Track and hold: LTCMOS
- High-speed trigger: HP-5
- Logic trigger: NMOS IIIB
- Timebase: HQMOS
- 16-channel DAC: LTCMOS.

The CPU system was designed to take maximum advantage of the existing firmware allowing a high level of compatibility between the HP E1426A and the HP 54503A. The HP E1426A offers the user many advantages over its counterpart, the HP 54503A. Along with the benefits of standardization of the VXIbus, the user can configure a single VXIbus frame with up to 24 500-MHz acquisition channels, all synchronized via the ECL/TTL backplane trigger. The E1426A firmware was customized to take advantage of the removal of the display from the traditional oscilloscope architecture to optimize the throughput to the VXIbus. This, coupled with shared memory, has resulted in a significant improvement in throughput compared to the IEEE 488 (HP-IB) based HP 54503A. In addition to SCPI, the E1426A also accepts HP 54503A IEEE 488.2 commands.

HP E1440A 21-MHz Synthesizer and Function Generator

The HP E1440A synthesizer circuits are derived from the HP 3324 synthesized function/sweep generator and the HP 3325 synthesizer/function generator. With this leverage it was easy to provide a full-featured instrument. All features associated with the HP 3325 are available, including AM and PM, plus extra benefits like SCPI programming.

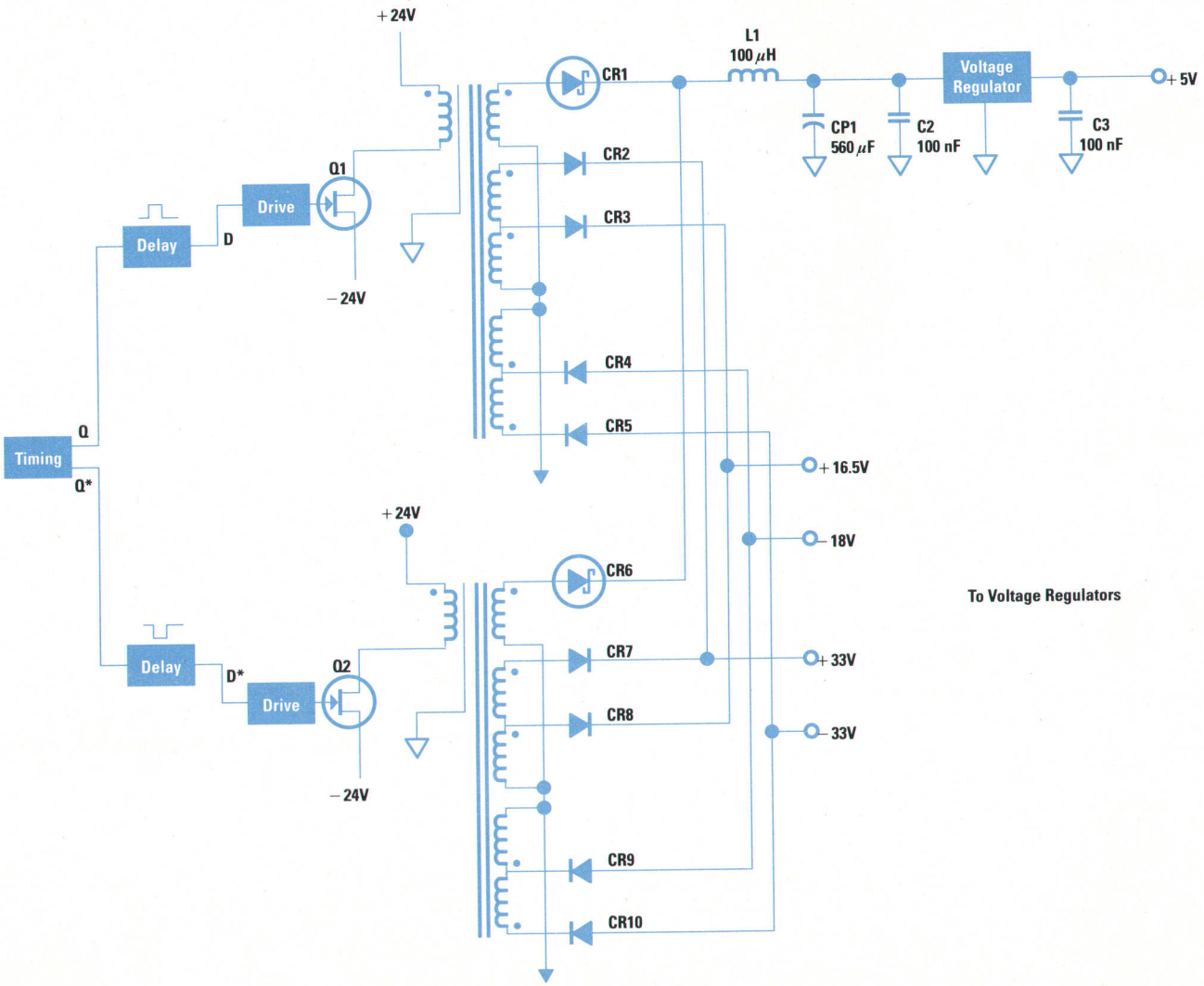


Fig. 1. The dc-to-dc converter used in the HP E1440A synthesizer and function generator.

The HP E1440A plug-in module occupies two C-size mainframe slots and has a standard front panel providing BNC type electrical connectors. SMT (surface mount technology) was necessary for the signal board to achieve the high-package density of the module.

All critical sections of the synthesizer, such as the VCO, mixer, preamp, and reference oscillator are shielded, each with a two-sided metal cover to reduce any induced noise, especially from digital frequency dividers and other switching circuits. Additional shielding of the power supply and microprocessor board area was necessary to improve the purity of the output signal. To avoid crosstalk from the switcher of the module power supply to the output amplifier, the whole dc-to-dc converter with the output filters had to be shielded with a steel box. The printed circuit board has feedthroughs around the dc-to-dc converter area so that this box totally encloses all switching circuits. This shield is also necessary to satisfy VXIbus specifications for magnetic field radiation on the module surface.

Both boards are mounted with screws (every 3 inches) to the module cover. The chassis ground and floating ground are coupled with 1-nF capacitors via these screws. For the same reason, BNC connectors are mounted with ring capacitors (like a washer). The P1 and P2 connectors are shielded with grounding brackets to reduce the radiation throughout the connectors and to make a good, low-inductive digital ground connection from the microprocessor board to the mainframe. With gaskets on nearly all slots, this module meets the VDE Class-B RFI regulations.

Possible ground loops caused by interconnected instruments are reduced by isolating signal ground from the VXIbus and chassis ground. All microprocessor-related control signals are coupled via optocouplers while the analog signals to and from the VXIbus are coupled by small transformers. The number of optocouplers is reduced by serializing the connection between the microprocessor and the signal board. Standard components are used, such as an MC68661 USART on the control board and standard shift registers on the signal board for serial-to-parallel conversions and vice versa. With a transfer rate of 800 kilobaud, a data rate of about 40 kbytes/s is achieved for random access to the device bus, which has seven address, one R/W, and eight data bits.

For generating floating voltages of +5V, 15V, and 30V with very low ripple (< 1 mV), a new concept for the dc-to-dc converter was needed. It consists of two forward converters with a duty cycle greater than 50%, running 180 degrees out of phase (see Fig. 1). The rectified voltages of both converters are connected together resulting in a voltage with 100% duty cycle. For this reason the LC filter does not need to filter the high-voltage ripple from a single forward converter. It only has to reduce the switching spikes caused by the rectifier diodes.

HP 1416A RF Power Meter

The E1416A VXIbus RF power meter evolved from the HP 70100A power meter used in the HP 70000 modular measurement system (MMS) and the VXIbus HP E1410A voltmeter. The E1416A has a high level of reuse and leverage from the designs of these two products, which reduced the cost of R&D and the time to market.

The measurement of power is often made in locations that have high levels of RF noise, so there is a need for good shielding from RF fields and immunity from noise conducted into the unit via cables. The VXIbus specification dictates a high level of immunity to both conducted and close-field radiated interference from within the VXIbus mainframe. VXIbus also specifies the level of close-field radiated interference generated by the instrument.

To handle the RF noise problem in the HP E1416A a decision was made to use two circuit boards, an analog board for the low-level power meter circuits and a digital board for the VXIbus interface and instrument controller. This is important for both performance and manufacturing considerations. There was a potential problem with the coupling of high-frequency noise generated by the digital hardware into the measurement signal. This is eased considerably by separating the circuits and their grounds. In addition, using two boards improves utilization of the printed circuit panels and allows the analog section to use a less-expensive two-layer construction.

The board layout and grounding are significant factors in meeting the VXIbus performance and electromagnetic compatibility specifications. The digital board uses a ground-plane layer to reduce the inductance of the ground traces. The analog board is constructed on a two-layer printed circuit board with a gridded ground system for the digital interface and separate ground systems for noisy and quiet analog circuits. These grounds are connected together on the digital board at a star point.

The sensitive analog input circuits have a ground reference that floats at the same potential as the sensor, which may differ from that of the rest of the instrument because of long sensor cable lengths. This ground reference is a separate single point ground system driven by a broadband opamp, supplying all of the ac section of the power meter up to the synchronous detector. To ensure trouble-free operation in environments with high levels of RF radiation, the HP E1416A circuit boards fit into an aluminum alloy case of high shielding integrity. The number of apertures and their size were reduced as much as possible and each seam around the case is either firmly clamped or sealed with a conductive gasket. The case also functions as runners that slide into the mainframe guides.

500-MHz Digitizing Oscilloscope
Don Smith
Project Manager
Colorado Springs Division

21-MHz Synthesizer and Function Generator
Harald Mattes and Helmut Sennewald
Project Engineers
Böblingen Instrument Division

RF Power Meter
Tony Lymer
Project Engineer
Queensferry Microwave Division

Front-Panel Design

Since VXIbus instruments lack sufficient front-panel space for displays and controls, the instrument designer must turn to a computer for a virtual front panel. HP's interactive test generator (HP ITG) software allows the programmer to control VXIbus and rack-and-stack instruments interactively and speeds development of test programs.

HP ITG is a program that runs on HP BASIC workstations, HP BASIC/UX, and MS-DOS. Therefore, it offers the power and performance of HP BASIC in the development or execution environment, as well as compatibility with industry standard operating systems.

The software has two different environments. The first is the development environment. This is the windows or panels environment in which instrument panels are used to generate code that is displayed in the Editor window of this environment and then stored to a file for execution. A mouse is used to select from pop-up windows and pull-down menus (see Fig. 6, page 23). The second environment is the execution environment. In this runtime environment, the application is executing, not HP ITG. No front panels are seen, and the application executes rapidly.

Instrument States

When the instrument is in a desired configuration, or setup, the user can save this setup as an instrument state.

Small, Low-Cost Mainframe with a Register-Based Interface

The HP E1300A/1A VXIbus mainframe is designed for high performance and low cost. For cost and size reasons, the VXIbus B-size or VMEbus 6U format was chosen. The desire to maximize the number of available card slots with field wiring brought about the idea of an interdigitated backplane with seven B-size and three A-size slots for external modules and three internal B-size slots for the built-in command module and the two-slot HP E1326 multimeter.

The mainframe with the built-in command module is stand-alone. This required the addition of other features not found in other VXIbus mainframes such as keyboard and display, dc power operation, and mass storage. Careful design was necessary to minimize the conducted and radiated noise environment that the shieldless B-size cards operate in. The power system is able to operate from either ac, or optionally, dc power with "bumpless" transfer between power sources. There are two power supplies, one OEM offline switcher providing +5 and ±12 Vdc and an optional dc-to-dc converter that is connected between the ac main and the first supply. This second power supply senses when the ac power becomes unusable and switches to dc power operation transparently.

The mass storage devices were leveraged from the HP 9153C disk drive. The controller board, ruggedized hard disk, and flexible disk drive were integrated into the mainframe around the internal VXIbus slots and power supplies. Communication with the disks is accomplished by piggybacking on the built-in command module's HP-IB port.

Designing the HP E13XX series of VXIbus cards provided many challenges. Size was the foremost challenge with only approximately one-half inch of vertical space and 50 square inches of horizontal space in which to design. Shields in this limited vertical space were out of the question. Therefore, an extra effort was made to minimize noise sources by using good ground gridding techniques, minimizing clock and power supply loop areas, and controlling signal rise times. The register-based interface was chosen for space and cost reasons. An ASIC was developed to do the handshaking and timing for the register-based interface.

The greatest space challenge was in the development of the field wiring terminal housing. This housing has to contain screw terminal blocks large enough to handle 250 Vac, support a 35-pound load, and be less than 0.8 inch wide. We did a lot of 2D and 3D modeling to create this design. This housing design has been scaled up for use in HP's C-size VXIbus offerings.

Cost was also a major challenge in designing the cards. Two-layer printed circuit boards were the rule rather than the exception, further increasing the noise reduction challenge. Most of the cards' backplane interfaces were common, many right down to the layout of the components. This leverage helped to decrease costs and time to market.

Von Campbell
Project Manager
Loveland Instrument Division

HP ITG will record this instrument setup in memory as a list of values for each instrument function or control. The user gives this instrument state a unique name. The user can then recall this instrument setup at any time by using the state name instead of having to set each individual value separately.

Another feature is called automatic incremental state programming. This simply means that the software will let the controller determine the fewest commands necessary to reconfigure the instrument to the state selected by the user. When changing an instrument from its current state to some other state, the list of values for the current state is compared with the list of values for the next state. The software then identifies the values that need updating and sends only the commands necessary to update those values. This can speed up the execution time substantially compared to traditional techniques.

Instrument Drivers

The drivers for HP's VXIbus instruments generate SCPI commands. These drivers are ASCII text files that contain the SCPI command information and HP ITG panel layout information for a particular instrument. These files are loaded into memory by the development environment.

Instrument drivers are written in an independent language developed by HP. This means that these drivers are not tied to a particular operating system. This allowed HP to bring HP ITG functionality to another operating system without having to rewrite any drivers.

HP ITG also allows the creation of application panels, a panel that represents a group of instruments, unique subprograms, or both. The task is similar to writing an instrument driver file. Once this file is created, it can be used by any person with a development system that has access to the application driver file.

Industrial and Mechanical Design

In addition to colors and graphic styles, HP has adopted several other standards for external ease of use. Series C instruments have at least three LED annunciators (Failed, Access, and Error) to assist in system configuration and troubleshooting. Failed indicates an interface failure, Access indicates VXIbus backplane activity, and Error indicates an SCPI user programming error. Additional LEDs indicate instrument-specific activities.

Series C instruments are normally positioned vertically causing cables to hang down. This can obscure the LEDs or connector labels. Hence, all LEDs are located on the top of the module and all labeling must be on the side of the connectors. In addition, triggering, clocks, and synchronizing signals are located centrally on the panel and analog signal input and output are on the bottom.

A dual three-wire analog bus standard has been defined which is suitable for microvolts but also has approximately a 10-MHz bandwidth. There are six HP modules presently using this standard including multimeters and scanners.

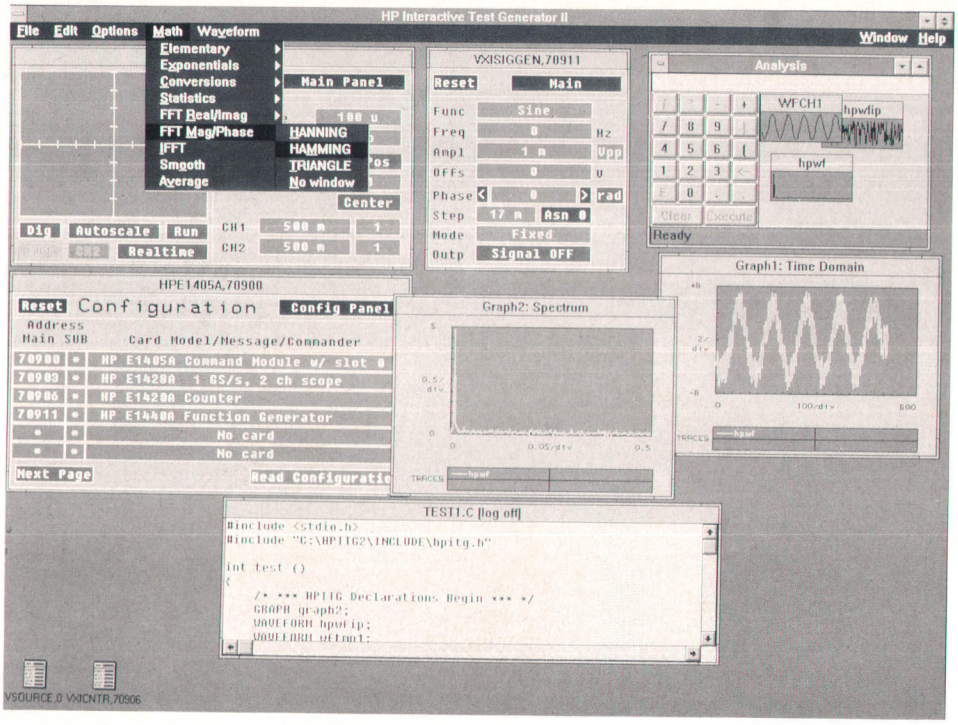


Fig. 6. HP ITG development environment, showing typical instrument soft front panels.

All calibration controls (e.g., potentiometers) on HP VXIbus instruments are either electronic or are accessible from the front panel so module removal is not necessary for adjustments. Switches for logical address and interrupt and bus request level are accessible without removing shields.

Acknowledgments

The authors would like to thank Jay Nemeth-Johannes, Don Smith, Helmut Sennewald, Harald Mattes, Tony Lymer, and Von Campbell as well as all of the others who helped make HP VXIbus a truly interdivisional effort.

Design of Mainframe Firmware in an Open Architecture Environment

Compatibility, portability, expandibility, usability, scalability, and compliance with SCPI are some of the attributes designed into HP's VXIbus mainframe firmware.

by Paul B. Worrell

The VXIbus common firmware architecture is designed to meet the needs of HP's instrument mainframe products that are based on the VXIbus (VMEbus Extensions for Instrumentation). The design and definition of this architecture was based on experience with other HP mainframes such as the HP 3497A, HP 3852A, and HP 3235A, and the needs of two newer products, the HP E1300 B-size VXIbus mainframe and cards and the HP E1400 C-size VXIbus mainframe and cards.

The design of the firmware for these mainframes was influenced by many external factors. Some of these factors included the IEEE 488.2 standard, the evolving VXIbus instrument standard, the emerging SCPI standard, and the need to support the architecture on multiple CPU platforms. This article will discuss some aspects of the design environment and the resulting impact on the firmware architecture, as well as product features included in the design.

This mainframe firmware architecture was designed with the following goals:

- Compatibility with the IEEE 488.2 instrument programming metaphor
- Compliance with SCPI and leverage of the HP SCPI language parser for all instrument development
- Expandability and openness to support for future enhancements or changes
- Better ease of use than our previous generation of products
- Support for the creation of a scalable family of VXIbus-compatible products from multiple HP divisions
- Support for a smooth migration between existing rack-and-stack instruments and modular VXIbus instruments.

IEEE 488.2 Compatibility

The IEEE 488.2 standard had just been ratified when the design effort for the VXIbus firmware architecture was started. IEEE 488.2 clarifies many aspects of instrument implementation that were previously left up to the designer's interpretation. This clarification inevitably removed some flexibility to promote interoperability between instruments.

For instance, under IEEE 488.2, an instrument is required to continue to respond to bus messages when in local mode, even when there is simultaneous front-panel activity. This requirement forces an IEEE 488.2 instrument to consist of at least two tasks, one to handle the IEEE 488 bus (HP-IB) and one to handle front-panel operations. IEEE 488.2 introduces the concept of sequential versus overlapped operation. A sequential command always finishes before the next command is executed. An overlapped command allows execution of subsequent commands while the device operations initiated by the first overlapped command are still in progress. Overlapped commands typically require one additional task per simultaneously operating overlapped command. The article on page 29 describes the two tasks associated with handling bus and front-panel activity.

IEEE 488.2 instruments aren't allowed to perform some operations that previous generations of instruments were capable of doing. An example of one of these illegal actions is a measurement operation that will never complete. The IEEE 488.2 message exchange protocol requires that all commands execute to a definite completion. Another example is the case of a classical voltmeter, which when it is put in internal-trigger mode will always have its latest reading ready to be output to the bus. This type of instrument behavior is also a violation of the message exchange protocol because the readings aren't the explicit result of the execution of a command, and the readings do not have the defined IEEE 488.2 response separators.

In exchange for these functional restrictions, the IEEE 488.2 standard specifies device behavior in the following areas:

- Standard message handling protocols that include error handling
- Unambiguous syntactic structures for program and response messages
- Common commands for instrument systems
- Standard status reporting structures and mechanisms.

The standard message exchange protocol describes how messages are received by an instrument and how responses to these messages are generated. This protocol also defines most of the error conditions that can arise between a controller and an instrument. One of the new error conditions that IEEE 488.2 defines is a query error. This error can result from either sending a new command to a device before reading all of the response of a previous command, or by requesting a response from a device that has not received a query.

The message exchange protocol also describes in detail the actions of an instrument in response to several asynchronous signals such as clear and trigger. It describes fully the operation of the input and output buffers of an instrument, even during error conditions such as a deadlock.

The IEEE 488.2 program and response message syntax allows for handling different types of data in one standard way. The program message syntax defines character, decimal numeric, nondecimal numeric, string, arbitrary block, and expression data that can be received by instruments. The response message syntax allows character, three types of decimal numeric, hexadecimal, octal, binary, string, and arbitrary ASCII data to be returned from instruments. It also allows for definite- and indefinite-length arbitrary blocks, which are useful for transmitting large quantities of data, 8-bit extended ASCII codes, or other data that is not displayable directly.

The common commands enable control and interrogation of instruments with a standard syntax. One example of a common command is the *IDN? query, which returns a unique instrument identifier that contains the instrument manufacturer, model specifier, and firmware revision number. Other commands configure the status subsystem, provide for message synchronization, reset the instrument and initiate internal instrument self-tests.

The common device status reporting model of IEEE 488.2 allows the user to generate a generic status handler that can operate with multiple instruments. The status subsystem is configured via common commands that specify masking options and SRQ (status request) generation. The status subsystem also provides a way of causing an SRQ on several common device errors such as an execution error, a command error, or a query error. It also allows SRQ generation for common events such as power-up or operation complete.

Because of the potential for design and implementation leverage between instruments, we determined that the common firmware architecture for the VXIbus would require that all instruments be IEEE 488.2 compliant. In addition to internal leverage, the commonality of a programming metaphor was thought to be an ease-of-use advantage for our customers. With the emergence of the SCPI industry standard instrument language, IEEE 488.2 compliance is now an accepted design practice.

SCPI Compatibility

Concurrently with the development of our VXIbus firmware architecture, HP's Measurement Systems Operation (MSO) was in the process of generating a common language for all of HP's test and measurement instruments. TMSL (test and measurement system language), as it was being called, was designed to provide both horizontal and vertical language portability between instruments. The management team saw the potential for standardization and implementation leverage with TMSL and decided that all VXIbus instruments would be TMSL compliant.

MSO decided that to support the new language more fully, it would be appropriate to provide software support. This support was delivered in the form of a high-performance TMSL parser that was written for the Motorola MC68000 microprocessor. Since our VXIbus command module hardware is based on the MC68000, this parser became an integral part of all our VXIbus instruments. The performance characteristics of this parser implementation enable our VXIbus instruments to deliver better throughput than our previous generation of instruments.

The need for a standard programming language across all test equipment was recognized by other test and measurement equipment manufacturers in 1989. These manufacturers formed a consortium to generate the SCPI (Standard Commands for Programmable Instrumentation) language specification. The initial core of the SCPI language was based on HP's TMSL. Since our instruments were designed to be TMSL compatible, we had to make only a few minor changes to be compliant with SCPI 1990.0.

Expandable Architecture

Early in the development of our common VXIbus firmware architecture, we determined that modularity and flexibility were the best ways to ensure a long life for the architecture. Modularity allows independent design and implementation of the various modules once their interfaces are specified. It also allows for upgrading modules to add new capability while keeping the old interfaces for backwards compatibility.

We were particularly careful to provide flexibility in the areas of instrument creation and binding because the design requirements specified that the architecture must have the ability to:

- Easily add drivers for new register-based modules
- Replace an existing device driver with a new driver with enhanced capabilities
- Supply replacement drivers for rapid turnaround of defect fixes to customers and to HP field service and support organizations
- Allow creation of custom instruments that could be easily integrated by customers
- Allow construction of virtual instruments, which are arbitrary combinations of message-based and register-based devices
- Allow construction of pseudoinstruments, which are code-only modules with an instrument-like interface.

Driver Replacement. Replacing existing drivers in HP's VXIbus firmware architecture simply involves downloading the driver from the host machine to the command module in the mainframe. The downloaded driver is placed in nonvolatile RAM so that it will remain in place after power cycling. During power-up the driver RAM area is first searched by the resource manager when it is binding devices to drivers. This guarantees that downloaded drivers will always be able to replace a ROM version that handles the same devices. New drivers can also be added by downloading. Downloadable drivers ensure rapid distribution of defect fixes since the customer only needs to be sent a disk with the driver software rather than a new ROM.

Fig. 1. First-level instrument selection menu for a system containing a voltmeter, a switch, and HP IBASIC.

Custom Instruments. Drivers for custom instruments are easily incorporated into the mainframe system just like any other downloadable driver. At configuration time, these drivers tell the resource manager about the devices and resources they need. The resource manager then starts them like any other driver, and since they are downloadable, custom drivers can replace any ROM driver in the system.

Virtual Instruments. A virtual instrument is a group of individual instrument drivers bound together in one driver. To become a virtual instrument, a driver must indicate to the resource manager during configuration that it needs a particular set of devices. The resource manager will bind those devices to the driver and start it. One of the first virtual instruments is the scanning voltmeter, which consists of a voltmeter device and one or more switch devices. The scanning voltmeter behaves like a voltmeter with many input terminals, which are selected via channel specifiers. Customers can download their own configuration tables that control the binding of devices to drivers.

Pseudoinstruments. Some programs running in the VXIbus mainframe do not communicate with devices directly, but they do require some support from the system. These programs are called pseudoinstruments, or code-only instruments. After all the instrument devices in the system have been bound to drivers, the resource manager looks for drivers that identify themselves as pseudoinstruments. It then determines the resource requirements for these drivers and starts them. HP IBASIC is an example of a pseudoinstrument driver. IBASIC has an SCPI programming language that is used to communicate with the IBASIC execution engine, but it doesn't require any VXIbus devices to function.

Better Usability

To provide better ease of use, we decided to perform extensive usability testing. In the design phase of this project, a user interface and front-panel prototype was constructed. This prototype was used to test design alternatives for the front-panel layout and menu system organization. A test script that consisted of a list of user tasks was generated and then people from various departments were asked to follow the list. Their questions, reactions, successes, failures, and comments were recorded. This input was used to refine the design of the VXIbus system display functionality. With user feedback it was possible to test for reactions to different types of menu presentation and organization.

Customers of our earlier products appreciated being able to use IEEE 488 bus syntax directly from the front panel.

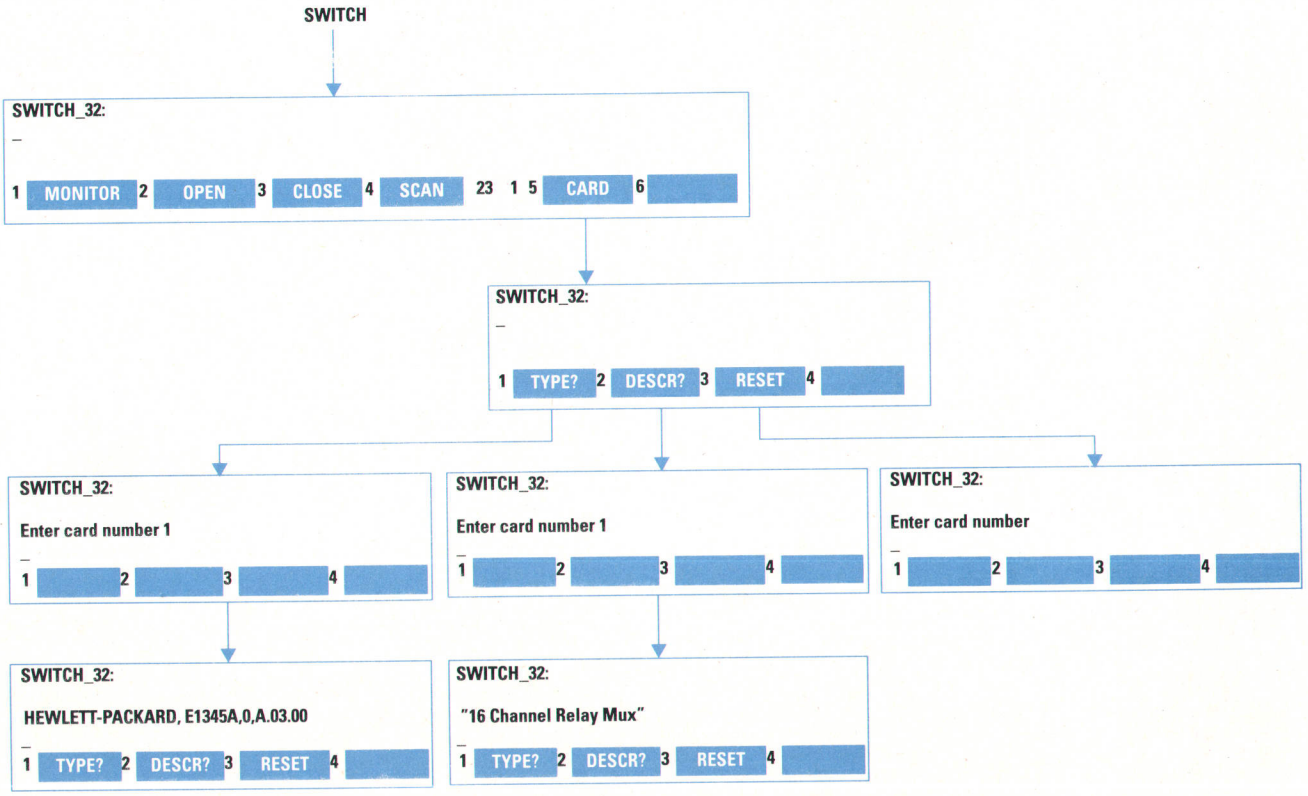


Fig. 2. The range of selections available after selecting SWITCH from the first-level menu in a system containing an HP 1345A relay multiplexer on card 1. The 1 that appears where it says "Enter card number," is entered by the user.

We felt that this was a key feature that we had to provide in the VXIbus common firmware architecture since it is easy to learn and debug instrument syntax by trying it on the front panel.

There were also many customer requests for single-key setups for common measurement functions. This need was addressed by providing a menu system for the front panel. Fig. 1 shows the first-level instrument selection menu for a system containing a voltmeter, a switch, and IBASIC. Fig. 2 shows some of the possible menu selections that appear once the SWITCH menu item is selected.

When a specific instrument is selected, an instrument-specific function menu appears. These function menus are nested with the most likely selections appearing first. Usability testing was again valuable here in tuning the menu choices. Once a menu key is pressed, the user is prompted for any additional required information, the measurement is executed, and the result is returned to the display. The user can at any time recall the SCPI command that corresponds to the last menu choice. This feature allows users to interrogate the instrument for SCPI strings to program specific functions, and has been viewed as enhancing SCPI ease of learning.

VXIbus Product Scalability and Leverage

One of the early design goals for the firmware was leverage between the B-size and C-size VXIbus products. This was accomplished by designing a system that includes all the features required by both the HP E1300 and HP E1400 mainframe projects and then subsetting the implementation as necessary. The overall architecture was first put together with HP Teamwork SA, a tool that supports structured analysis and structured design. Structured analysis helped the design team verify the completeness and correctness of the architecture, and helped the management team with effort estimation and job partitioning.

Fig. 3 shows the first-level structured analysis diagram for the VXIbus firmware architecture. This diagram shows the interfaces to IBASIC, external terminals, built-in front panels, message-based devices, and so on. This graphical depiction of the system was very useful in explaining functionality to management and communicating architecture trade-offs within the design team.

The RS-232 terminal drivers are essentially the same for both products. The front-panel display driver is unique to the B-size product since the current C-size product has no built-in front panel. The resource manager functionality is also common except for message-based device handling (which is unsupported because of hardware restrictions in the B-size product) and user-defined system configuration tables that were added to the C-size product.

The operating system for both products, called VXI-OS, is generated from common source code which is retargeted for the specific hardware features of the various platforms. The operating system presents a consistent interface to the rest of the system despite certain hardware dependent features present on specific platforms. VXI-OS is described in the article on page 29.

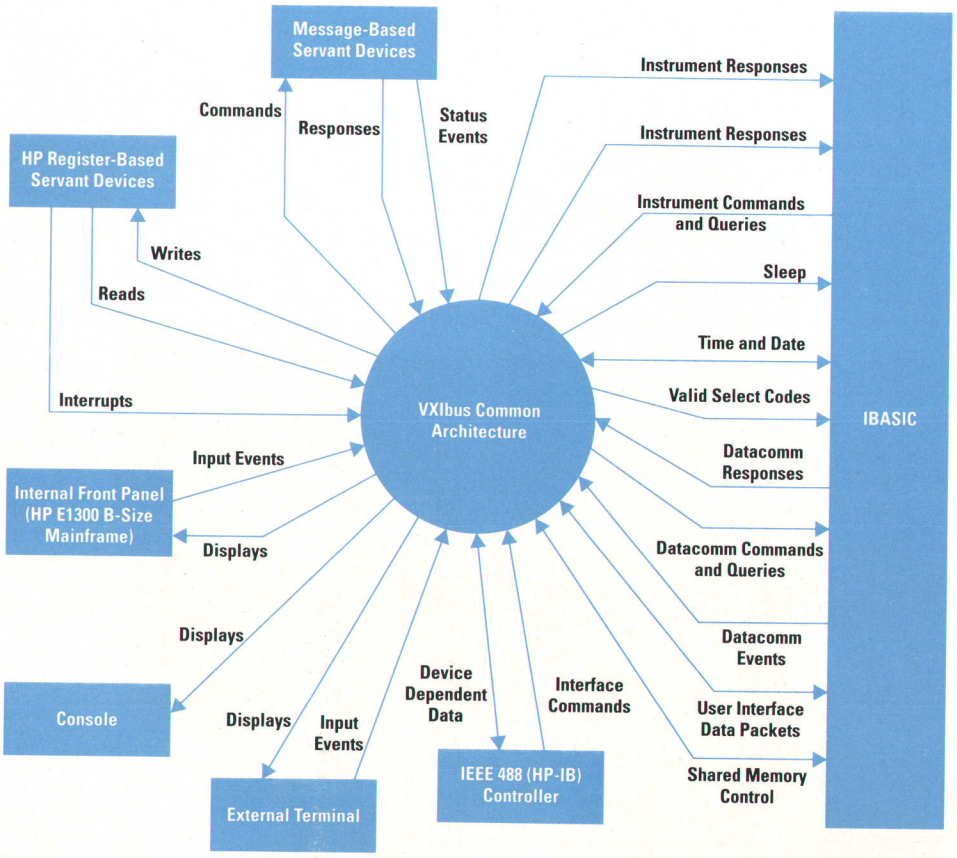


Fig. 3. Structured analysis diagram for the VXIbus firmware architecture.

Most of the instrument drivers are identical in both products. The only exceptions are cases in which additional capability is provided by the C-size product and the driver takes advantage of it. Even in those cases, there is only one copy of the source code, which is conditionally compiled.

In addition to the VXIbus common firmware architecture, we developed a common message-based kernel for use in all message-based instruments developed at HP. This message-based kernel includes the real-time, multitasking pSOS operating system, a common SCPI parser, and a recommended hardware layout.

Migrating Rack-and-Stack Test Systems to the VXIbus

Since application portability from rack-and-stack systems to the VXIbus was one of our goals, we had to develop a scheme that ensures that very little or no change has to be made to applications written for rack-and-stack architectures when they are ported to the VXIbus environment.

A rack-and-stack system consists of separate instruments which all have their own IEEE 488 addresses. To allow maximum portability, this type of programming metaphor should be preserved. Early in the design phase, the idea of embedding addressing information into the data stream was considered. Several problems with this design emerged.

For example, consider the case in which the command SELECT:DVM is used to specify that succeeding commands go to the voltmeter, and SELECT:CNTR is used to specify that succeeding commands go to to counter. In the simple case, if the developer sends SELECT:DVM, then sends a measurement command, and then reads the response, the response will be from the voltmeter that received the measurement command. In a more complicated scenario, if the developer sends SELECT:DVM and a voltmeter measurement command, then sends SELECT:CNTR and a counter measurement command, and then reads the response, the response that is read is dependent on the timing of the voltmeter and the counter measurements.

In this type of setup, the system software developer views the test system as one large instrument with parts that can be selected via commands. There is only one input buffer, one output buffer, and one copy of the status information. With this configuration, the software developer has to keep track of which instrument is currently processing information from the input buffer. Also, any of the instruments could be putting their results into the output buffer. The software developer must know which instruments have put what data in what order in the output buffer. If the timing between the initiation of a measurement and the result appearing in the output buffer changes, then it is possible for results to be placed in the output buffer with time dependent ordering. This task is even more complicated because errors could have occurred on any or all of the instruments, creating unexpected results.

Device status is another area in which information sharing is problematic. There is normally only one service request bit in the device status register, which is shared among all the instruments capable of requesting service. When a service request is asserted, the software developer must try to unravel which entity in the system is really requesting service.

To solve these problems HP's VXIbus system provides a programming metaphor in which each instrument has its own input buffer, output buffer, and device status information. These VXIbus instruments behave exactly like separate IEEE 488.2 instruments. The software developer is able to use the same programming metaphor used with separate instruments in the rack-and-stack implementation. Porting the code to the VXIbus system can be as simple as substituting the IEEE 488.2 secondary address of the VXIbus instrument for the IEEE 488 address of the instrument in the rack-and-stack system.

Acknowledgments

When it became evident that with the available resources neither the HP E1300 or the HP E1400 mainframe projects could be implemented on schedule as separate projects, our R&D manager combined the two design teams and used those resources to design a common architecture for both products. Therefore, Joe Marriott deserves much of the credit for the success of the implementation of the firmware architecture, since he was able to provide an environment in which it was possible to share the architecture design. The team that constructed the VXIbus common architecture consisted of Chuck Platz who provided the resource manager, Chris Kelly who provided the operating system, Rick Hester who provided the terminal independent display system, and Karen Moy who provided the front-panel and menu systems. Martin Meyer, Dilip Muranjan, Jerry Metz, Bryan Sutula, and Dave Rustici were responsible for the design and construction of individual instrument drivers. Additional thanks go to Art Boyne and Darren Kwock who wrote hardware drivers. Ron Firooz provided initial project guidance and was almost single-handedly responsible for the integration of the two project teams.

Real-Time Multitasking of Instruments in the VXIbus Command Modules

The operating system in HP's command modules uses two reentrant processes to handle communication between the user and instruments on the VXIbus.

by Christopher P. Kelly

The ROM-based program (firmware) in the HP VXIbus controllers provides the "personality" that users interact with when operating the instruments in the VXIbus cardcage. The factors that affected the design of the firmware in these products include:

- The VXIbus standard, which was evolving during firmware development, describes certain required functions.
- The SCPI instrument language was chosen as the standard language for all HP VXIbus products. The IEEE 488.2 language standard (upon which SCPI is based) defines certain instrument behavior and some language constructs.
- Support had to be provided for both message-based and register-based VXIbus instruments with no programming differences visible from the HP-IB interface.
- The firmware was targeted to run on the on the CPUs for both the B-size HP E1300 and the C-size HP E1400 product families.
- Since ongoing development of instruments for the VXIbus is expected, the firmware had to allow the addition of new instrument drivers to the existing products with minimum development and upgrade effort.
- Software development for many register-based instruments from different divisions had to be coordinated.

This article explores the design of HP's VXIbus multitasking real-time operating system, or VXI-OS, and the system configuration firmware called the resource manager. A significant portion of the VXI-OS provides support for register-based instruments. To use the high-level SCPI language, register-based instruments with less on-card intelligence require more support from the host CPU than message-based instruments. Because of this fact most of the discussion in this article will be specific to VXI-OS support for register-based instruments.

The VXIbus Instrument Model

The VXIbus instrument model can be described as an "instrument on a card," meaning that each card acts as an independent instrument. This contrasts with some previous generations of HP card-based instruments. The older products use languages in which all the instruments in the cardcage operate as a single instrument. Also, the older architecture provides the integrated services of many different cards with a single, massive command language. The new model treats each card as a single instrument, but also allows combining several cards into a more complex virtual instrument.*

By adopting the new model, VXIbus instruments allow the use of a standard command language shared by both card and rack-and-stack instruments. For example, a VXIbus DMM and a rack-and-stack DMM use the same SCPI language in the same way. This provides some hardware independence to the user, who can now choose between performance, cost, and space without requiring changes in the software driving the instruments. The SCPI language standard protects the sizeable software investment the typical customer has made in the software for an instrument system.

The two major classes of VXIbus instruments are message-based devices and register-based devices. Message-based devices usually include all language processing and a large well-defined register set on the instrument card. Protocols for communications between a commander and a message-based instrument are well-defined and aimed at multivendor compatibility. By contrast, register-based devices have a smaller required register set and a simpler programming interface. These devices are usually smaller and less expensive than equivalent message-based instruments, but may require more support from the command module. When programmed at the binary register level, register-based devices can be orders of magnitude faster than equivalent message-based devices that use a higher-level language.

One goal of the VXI-OS firmware team was to provide the same language and programming interface for both register-based and message-based instruments. This means that both types of instrument interfaces should answer the HP-IB communications bus in exactly the same way, and should operate using exactly the same SCPI language regardless of the instrument type.

Register-based instruments, with little or no intelligence on each card, require the services of a microprocessor to use the SCPI language. This microprocessor can be shared by several instruments, since in most cases an instrument does not require full and continuous use of the microprocessor's power. This sharing of expensive,

* See page 26 for more about virtual instruments.

intelligent hardware can result in lower system cost than in alternative designs. In the HP VXIbus systems, the command module provides this shared microprocessor, which processes commands and data for many register-based instrument cards.

The HP Series 75000 models E1300 and E1405 VXIbus command modules are based on the Motorola 68000 microprocessor. The B-size E1300 and the C-size E1405 provide an HP-IB interface to many HP register-based devices. The E1405 additionally provides full VXIbus slot 0 command module and resource manager functions for message-based devices (see the article on page 6 for more about slot 0 functions). The design of the firmware in these modules permits power-on system configuration based on the instrument card mix. It also supports independent operation of multiple instruments in the cardcage, and the addition of new instrument drivers. Both modules are also able to use HP Instrument BASIC (IBASIC) to control the instruments.

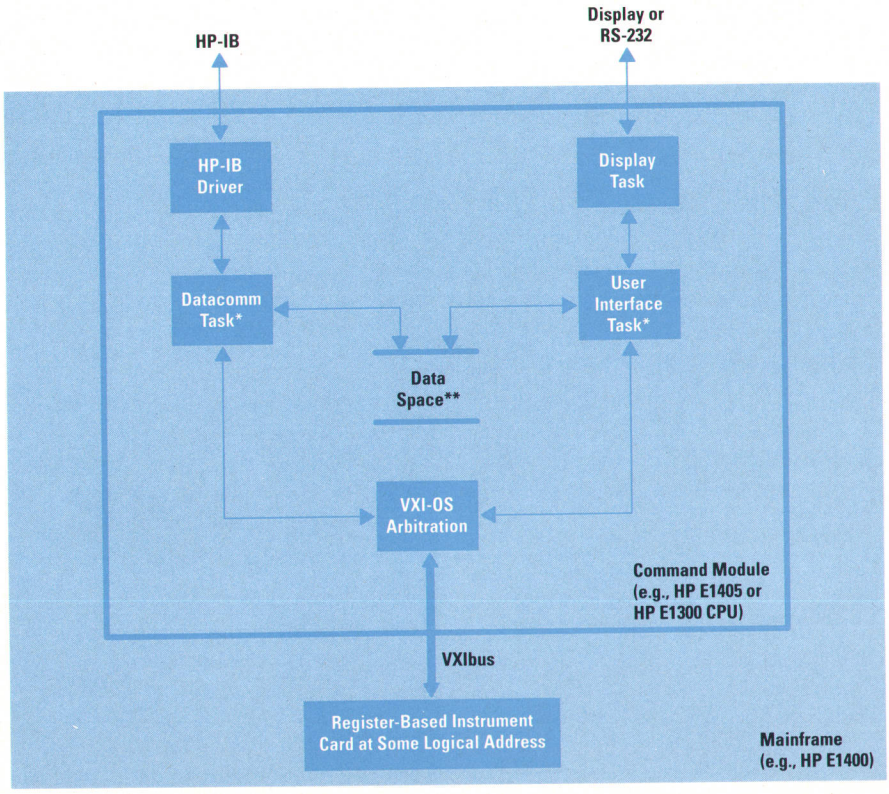
The microprocessor must do the following tasks:

- Receive commands from the HP-IB interface for one or several instruments
- Receive commands from the instrument front panel and RS-232 interface
- Process the SCPI language for register-based instruments
- Execute driver functions for register-based instruments
- Pass through commands from the HP-IB to message-based instruments
- Operate register-based and message-based instruments independently.

In addition to these product-oriented needs, several requirements were added that were driven by the development environment. One of these requirements was the need to ensure productivity for developers writing instrument driver software. During development, several developers were simultaneously writing instrument driver software for many different instruments. This software was essentially hardware drivers, which typically means real-time programming techniques and some assembly language programming for peak performance and maximum control of the instrument. In addition, these developers were required to solve many similar real-time programming problems for each driver. However, to be most productive, these developers needed to be able to use a high-level programming language such as C whenever possible.

To address these real-time programming problems in a general way, a real-time operating system was developed that provides a set of function calls for the instrument developer. This system has a C language programming interface that permits the instrument driver designers to use C and gain the high programming productivity levels associated with high-level languages and at the same time satisfy the needs of real-time programming.

Using a single general model for instrument operation, the resulting operating system, called VXI-OS, also frees the instrument designer from many of the problems of real-time software design. For example, simple rules regarding timing of instrument functions and use of



* Contains SCPI Parser and Execution Routines
** Contains VXI-OS Tables, Task-Specific Data, and Instrument-Specific Data

Fig. 1. Two tightly coupled (twin) tasks operate the instrument hardware in VXIbus register-based instruments.

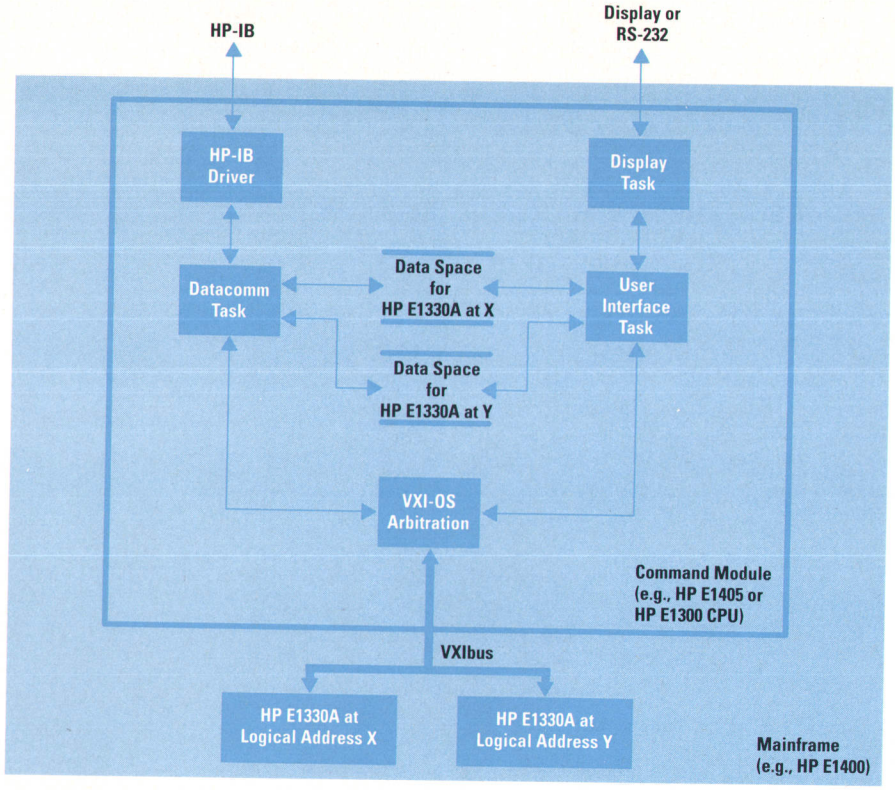


Fig. 2. An illustration of reentrancy. Two register-based counters are driven from one copy of the code for the datacomm task and one copy of the code for the user interface task.

interrupts allow cooperative interoperation of many instruments in the same CPU.

A calling mechanism that is address independent allows addition of RAM-loaded instrument drivers to the ROM-based system. This scheme supports a strategy of instrument driver upgrade and addition of new instruments to existing ROM-based VXIbus systems. At power-on, the system configures to the instrument cards found on the VXIbus, and is capable of finding and starting instrument drivers that have been RAM-loaded. This allows driver upgrade or addition of new instrument drivers long after the initial system firmware is completed. See the article on page 24 for more about downloading instrument drivers.

The VXI-OS Instrument Model

To meet the requirements of IEEE 488.2, instruments should be able to respond to front-panel commands while processing HP-IB commands. This approach differs from some instrument models used in the past, in which all instrument measurement hardware was operated by only one interface at any time. For many VXIbus instruments this is not an issue, since most message-based instruments do not have a front panel in the same sense as rack-and-stack instruments. However, for HP VXIbus register-based instruments the instrument model does include a front panel.

Since this model requires active command reception from both interfaces, in the VXIbus instrument model each instrument is implemented as two closely coupled tasks operating the same instrument hardware. These two tasks are called the *user interface* task and the *datacomm* task (see Fig. 1).

The user interface task receives commands from the display system, which is a task that operates the front panel of the instrument. This front panel may be a keyboard and a display that are integral to the chassis (as in the HP E1300 series), or it may be a terminal connected to the controller by an RS-232 or RS-422 interface. The datacomm task receives commands from the HP-IB interface.

As indicated in Fig. 1, the twin tasks of any given instrument are partners in operating the instrument hardware. These tasks are separate processes in the operating system with separate stack space and other memory allocations, but share the same global variables (data space in Fig. 1).

Together with these shared global variables, a number of resource locking functions in the operating system provide the tools necessary for cooperative use of the instrument hardware. These functions are represented by the block labeled VXI-OS arbitration in Fig. 1. The two tasks of a given instrument are nearly identical with the exception of the communications interface, which depends upon whether the task is serving the HP-IB or the front panel.

These tasks are written as reentrant firmware, so several instruments of the same type can be present in the VXIbus system simultaneously. These instrument tasks execute the same program at the same ROM address, but use separate data spaces to provide independent operation. This reentrancy allows the customer to operate any combination of register-based instruments simply by plugging in the cards required to perform the desired measurement.

Fig. 2 shows the case in which more than one register-based counter of the same type are plugged into the VXIbus. Because of reentrancy, the twin tasks execute one copy of the datacomm and user interface ROM code specific to the counters. However, because the counters are at different addresses and may have different setups, the tasks use separate data spaces to distinguish one counter from the other. Fig. 3 shows the software organization in the command module to handle three different register-based instruments plugged into the VXIbus mainframe.

Resource Management

When power is applied to the HP VXIbus system, the VXIbus controller configures itself to operate the particular set of instruments present on the VXIbus. The firmware that causes this action is the *resource manager*.

Following the power-on reset and the CPU self-test, the resource manager scans all VXIbus logical addresses for instrument identification registers. These registers can appear at any VXIbus logical address from 0 through 255. (In the A16 address space the identification registers appear at regular intervals.) The resulting list of instruments is used to generate a configuration table that includes the card manufacturer identifier, model number, logical address, and other data. The resource manager also identifies failed devices and places them into safe states.

The resource manager also configures the memory map of the VXIbus A24 and A32 memory space, and the seven VXIbus interrupt lines are assigned to VXIbus commanders. Since the VXIbus allows hierarchical instrument ownership, the commander/servant hierarchy is established at this time. This action assigns instruments to commanders based on the logical address of each.

After the instrument card set is identified, the resource manager assigns instrument drivers to the register-based cards. This is done by asking each driver in the library whether it recognizes a particular set of cards. When a match is found, the driver is flagged for later startup as an instrument task, mapping to a particular set of cards.

This scheme allows multiple copies of any instrument type—one for each card or card set matching the driver. For example, three digital multimeter cards on the bus will trigger the startup of three digital multimeter instruments, one for each of the three cards. This is possible because, as described above, the drivers (which become datacomm and user interface tasks when started) are written as reentrant code.

Once this phase of resource management is completed, the resource manager calls each flagged driver to determine the driver's needs for memory and other system resources. A cumulative list of resources is compiled and passed to the VXI-OS operating system. This information is compared to available resources to see whether the

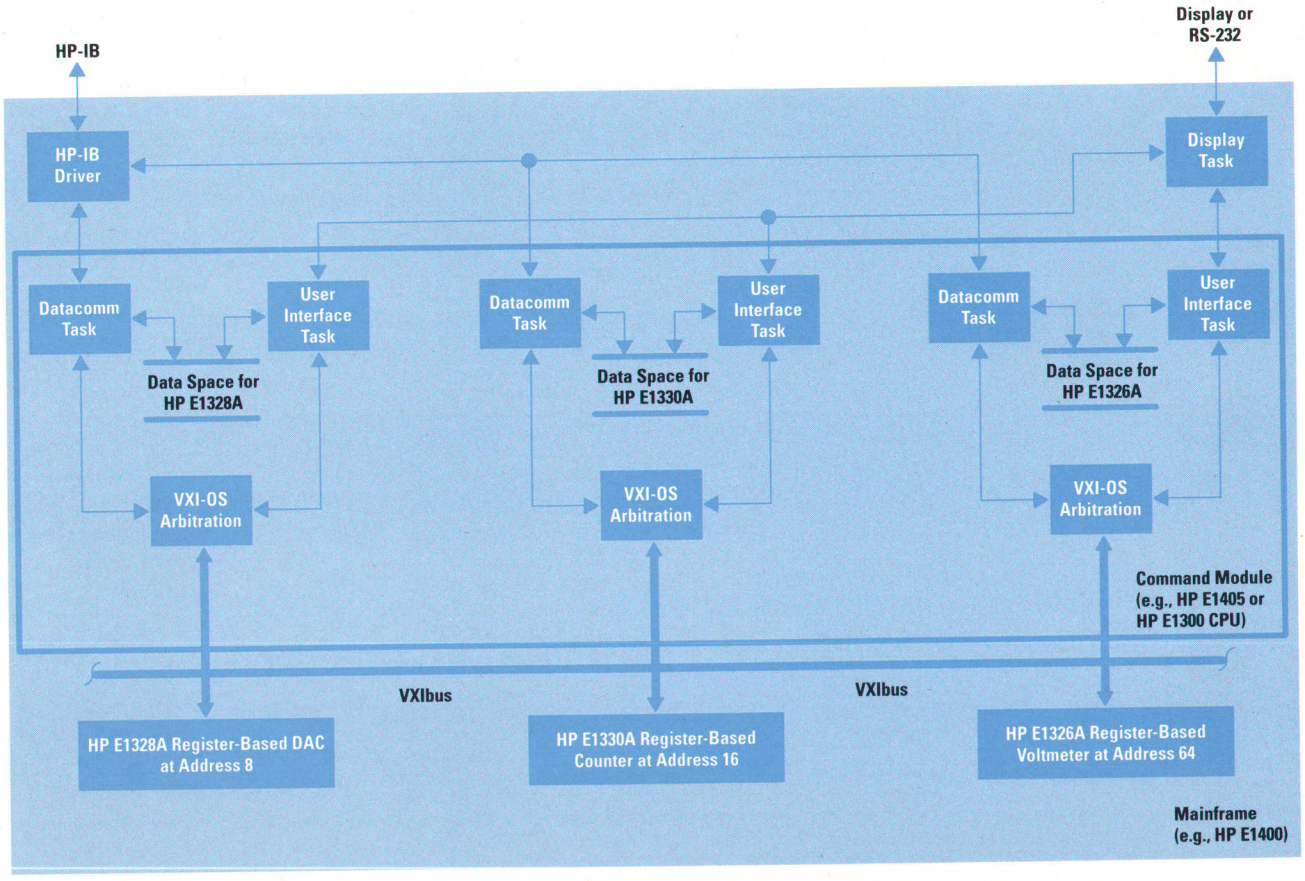


Fig. 3. Command module software organization for handling three different types of register-based instruments.

system can be successfully started with the present card set. Later, the information is again used to assign the required resources to each instrument task as the task is started.

The next phase of startup reserves the required memory and other resources, and configures the operating system to match the present card set. The operating system is started and in turn it starts the resource manager `pass 2` process as its first and highest-priority task. The resource manager `pass 2` process uses the card configuration table mentioned earlier to spawn and activate the instrument tasks that are associated with the VXIbus instruments found earlier. These tasks are assigned to instruments at particular logical addresses, and are also assigned HP-IB secondary addresses.

The resource manager `pass 2` process will spawn and activate only the first of each pair of instrument tasks. The first task of each pair started is the user interface task. This action is performed by a special operating system call. Each of these initial instrument tasks is responsible for starting its twin task, the datacomm task, using another operating system call. These calls establish the shared global data areas and other operating system data structures that provide the tight coupling between the tasks.

An additional special instrument called the *system instrument* is also started. This system instrument allows certain system-wide operations to be controlled. These operations include mapping of triggers on the VXIbus and the setup of communication interfaces and system time, date, and diagnostic functions.

When the resource manager completes the startup of all instruments, it starts a task to control each display interface. These display tasks each control a front panel or RS-232 interface. The resource manager then pauses (becomes dormant) until later operations require its services. As soon as the resource manager pauses, the lower-priority instrument tasks begin to run. Each instrument task pair (user interface and datacomm) calls the operating system to identify the addresses of any instrument hardware allocated to that instrument. Also at this time, resource sharing locks and other data structures are initialized, a self-test is performed on the hardware, and the connections to the HP-IB and the display system are established. Finally, each task pauses until commands are received for that instrument.

Communication with VXIbus Instruments

Since there are usually several instruments served by the two communication interfaces (HP-IB driver and display task), the interfaces must select which instrument is to receive the incoming command strings. In the HP-IB interface, this selection is performed using HP-IB secondary addressing. In the display system, this selection is performed by ASCII commands to the display system, or by softkey escape sequences.

In the HP-IB interface, most communication is handled by a TMS 9914 interface IC. This chip handles primary address actions satisfactorily, but secondary addressing presents additional challenges to the interface. Support

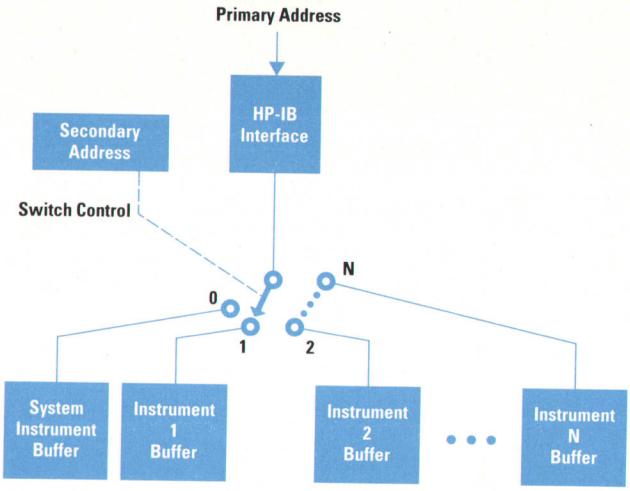


Fig. 4. An illustration showing how the HP-IB secondary address is used to select an instrument in a VXIbus system.

for secondary addressing is provided by a custom HP-IB assistant gate array IC. In the secondary addressing scheme, the VXIbus controller listens to a primary address, for example, address 9. When a primary address sequence is received, it is followed by a secondary address sequence. This secondary address, between 0 and 30, is used by the HP-IB interface to select one of the instruments connected to the controller at the specified primary address. If for example, a voltmeter has been assigned to secondary address 6, the HP-IB address for this device would be 906.

If the device were being addressed by an HP 9000 Series 300 controller using workstation HP BASIC, the full address used by the programmer would be 70906, where the 7 represents the HP-IB card address in the workstation. Another device in the same VXIbus system might answer at 70904, meaning secondary address 4.

To make addressing of devices easier, a relationship has been established between the VXIbus logical address and the HP-IB secondary address assignments. The relationship is that an HP-IB secondary address equals 1/8 of the VXIbus logical address. So a voltmeter set at VXIbus logical address 48 would be assigned HP-IB secondary address 48/8 = 6.

When the HP-IB interface has received the full addressing sequence, it uses the secondary address to select which instrument buffer to use for the incoming communications strings. This routes command strings to the appropriate instrument task, where the strings are processed and the commands are executed. Fig. 4 uses the analogy of a switch to illustrate choosing the destination of a command string. The switch is controlled by the secondary address received from the HP-IB addressing sequence.

In the case of commands received through a display system interface, the instrument selection is made through more human-readable means. ASCII strings and labeled softkeys are used to select which instrument is to be addressed. After an instrument is selected, the display system is connected to that instrument task and the display format is controlled by the instrument. In this

manner, instruments with differing human interface needs can each present the user with appropriate prompts and control displays. Softkeys allow the user to perform measurements, change instrument setup, view measurement data, and perform other typical instrument front-panel functions.

Instrument Operation

When a register-based instrument receives a command string, the SCPI parser interprets the string and calls an execution routine to perform the action that corresponds to that string. Before a task can operate upon its instrument hardware, it must be sure the hardware is not already in use by the other twin task. To prevent collisions between the HP-IB task and the display system task, each will request control of the instrument hardware through a call to a VXI-OS function. This function call executes in less than 50 microseconds in the 8-MHz MC68000. The caller is blocked if the hardware is already in use, and is again made runnable whenever the hardware becomes available.

When an instrument function requires an extended time to complete, the instrument task may decide to wait for a hardware interrupt indicating completion of the measurement function. A call to the operating system allows the caller to connect the instrument's interrupt service routine with the interrupts from an instrument card and then pause the task until the interrupt arrives. Upon interrupt assertion, the instrument service routine begins execution. The instrument service routine has access to the shared global data area of the two parent tasks of the instrument, and also can be passed pointers to the data buffers or other variables. When the instrument service routine has completed its function, it can signal the parent task to awaken from the pause condition.

This form of operation can result in high efficiency in the use of the shared CPU, since many instruments such as switches (e.g., relay multiplexers) have relatively short command sequences and relatively long relay closure times. Without this interrupt technique, the CPU would waste time waiting for relay closure to complete. This allows other instrument driver tasks to use the CPU time for other purposes, effectively overlapping the operations of several instruments. Guidelines based on the speed of the pause, interrupt, and awaken functions tell instrument driver writers the timing circumstances in which these tools are appropriate.

The VXI-OS has a similar function to perform the required operations resulting from the HP-IB Device Clear command. Device Clear is an asynchronous command used by the HP-IB interface to force an instrument into a condition in which it can accept commands. It can be used to abort ongoing measurements, and when properly implemented, requires an instrument to be interruptable in any of its possible states.

To awaken an instrument task that is paused waiting for resources or waiting for an interrupt, the VXI-OS contains a "clear instrument" function. This function sends wakeup signals to an instrument task that awaken the task from any of the pause conditions. If the task is awakened in this manner, the return value from the "wait for interrupt" function will indicate that Device Clear was received, not the expected interrupt.

Conclusion

The VXI-OS is a real-time multitasking operating system developed for HP VXIbus command modules. VXI-OS contains many functions that support instrument drivers for VXIbus register-based cards. The instrument model for these cards includes features allowing multiple independent instruments to share one CPU. Solving these problems once for all instruments, VXI-OS provides the instrument firmware with a high level of stylistic similarity and increases the productivity of instrument firmware writers.

Acknowledgments

Many thanks to Chuck Platz, who wrote the resource manager for both the HP E1405 and the HP E1300 Series commanders, for his assistance and ideas. Thanks also to Martin Meyer and the other instrument driver writers at HP's Loveland Instrument Division who provided helpful suggestions regarding enhancements to the original VXI-OS definitions. Their patience and skill during the parallel development of the operating system and instruments enhanced the results of both efforts.

VXIbus Programming in C

A library of C functions provides functionality that makes it easier for test program developers to create applications that communicate with HP-IB and VXIbus instruments.

by Lee Atchison

The VXIbus is a standard instrument control bus that allows high-speed access to test system instrumentation. To achieve these high speeds, controllers connected to the VXIbus mainframes require sophisticated test software. These controllers must also be powerful enough to take advantage of VXIbus's speed and versatility. This paper describes a modular instrument communications library that is designed to work with VXIbus interfaces. The library is designed to be extensible and applicable to several different computer architectures, operating systems, and instrument communication interfaces.

Library Features

The instrument library defines a core set of functionality that will work on all test-system instrument communication interfaces. This core includes read and write, addressing, timeout, locking, triggering, interrupt handling, and status reporting. When using just this core functionality, an application can be written to use HP-IB instruments (or VXIbus instruments through an HP-IB-to-VXIbus translator such as the HP E1405 command module). Later, the application can be ported with little or no change to talk to VXIbus instruments directly from a VXIbus-based controller (bypassing the HP-IB completely).

This core functionality is sufficient for most measurement application needs. However, if an application requires additional functionality, features can be added that apply only to that application and its associated interface to the test instruments. VXIbus adds extended triggering capabilities (TTL and ECL trigger line control), mapping (allowing access to register-based and memory-mapped devices), and enhanced interrupt handling. When using the additional VXIbus functionality, a test program (while no longer portable to the HP-IB) can still be ported to other VXIbus controller environments. The same test program should work on VXIbus embedded controllers and external controllers using a mainframe expander such as MXIbus.

Improved Productivity

The library contains features that help to improve the productivity of a test engineer developing a measurement application. These features include easy-to-use routines, high-level I/O commands, and a functional uniformity across applications.

Ease of Use. The library routines are designed to be easy to use and understand. The library incorporates the features specifically needed to communicate with instrumentation, especially instruments that look or act like IEEE 488.2 or SCPI instruments. This differs from other communication libraries that allow communication with devices that may or may not be instruments (such as printers, plotters, disk drives, etc.).

High-Level Commands. High-level formatted I/O commands are available to make test programs easier to write. These include routines that allow converting to and from a number of data formats used by most instruments. For example, number converters are available to convert data to and from six different numeric formats that are typically used in IEEE 488.2 and SCPI instruments. These formatted I/O routines, while based on the C language stdio routines printf and scanf, have extensions specifically for instrumentation, such as controlling EOI and END bits, IEEE 488.2 number and string formats, and so on.

Uniformity. Users need to learn only one set of routines despite the number of controllers and interfaces being used. This means that once a test programmer learns these library routines, there is no need to relearn a new set of routines when using a different controller.

The following C program takes a simple measurement from a SCPI-based voltmeter.

```
main(){
    INST dvm;
    double res;

  /* Print message and terminate on error */
    ionerror(I_ERROR_EXIT);
    /* Open the voltmeter */
    dvm=iopen("voltmeter");

    /* Take a measurement */
    iprintf(dvm,"MEAS:VOLT:DC?\n");
    /* Read the results */
    iscanf(dvm,"%f",&res);

    /* Print the results */
    printf("Result is %f\n",res);
}
```

This program shows several different things about the instrument library:

- It shows how to install an error handler, in this case, a standard error handler that prints an error message and then terminates.

- It shows how to address an instrument using a symbolic name. This isolates hardware dependencies in the structure of the address.
- It shows a simple example of the formatted I/O capability to send a command and read and parse the response.

Addressing an Instrument

Before communicating with an instrument, a link must be established between the test application and the instrument. This link is created with the iopen* routine. This routine takes as a parameter the address of an instrument. It then establishes a session with that instrument and returns a number that uniquely identifies the session with the instrument. This unique number, called an INST identifier, is used by all other routines in the library to communicate with the particular instrument set up in the iopen call. A number of other parameters, such as timeout values, can be set on a session-by-session basis.

The address of a VXIbus instrument can take one of two different forms:

vxi,<laddr>	(e.g., vxi,24 or vxi,128)
<symbolic_name>	(e.g., voltmeter, dmm, scope)

The first address format specifies a VXIbus instrument by its logical address <laddr>, which can be any number from 0 to 255. The second format allows an address to be a symbolic name for an instrument. These symbolic names must be assigned by some other system resource (such as the resource manager or a configuration file) to refer to a specific instrument at a specific address. For example, voltmeter may be the symbolic name of the instrument located at VXIbus logical address 24.

High-Level Interface

The high-level interface provides a formatted I/O mechanism that is similar to the C stdio mechanism, except that it is designed specifically for instrument communication and is optimized for IEEE 488.2 compatible instruments. Three main routines are available:

- iprintf. Send a formatted message to a given instrument.
- iscanf. Receive a formatted message from a given instrument.
- ipromptf. Send a formatted message to a given instrument and then immediately read a formatted response.

These formatted routines allow writing and reading of several different formats of data. The formats include the following standard stdio data types:

%d	–	Integer data
%f	–	Floating-point data
%c	–	A single character
%s	–	A string of characters

The following data types have been added specifically for VXIbus instruments:

- %b/%B. Defines binary-coded data that uses IEEE 488.2 definite-length and indefinite-length arbitrary block response data formats. Several options are available including byte-swapping values from the byte ordering of the instrument to the byte ordering of the controller. This allows fast reading and writing of large blocks of binary data in a manner consistent with IEEE 488.2 and SCPI.
- %e/%C. Defines characters and strings of characters that indicate end of data such as the EOI line for the HP-IB and the END bit for the VXIbus. This includes optionally setting the END indicator on a write call and waiting for an END indicator on a read call.
- %S. Defines IEEE 488.2 string response data. This is essentially a string enclosed in double quotes (with embedded double quotes escaped). The SCPI language also uses the string format.

Numeric data can be read and written in any of the following data formats:

@1	–	IEEE 488.2 NR1 format (integers such as 53)
@2	–	IEEE 488.2 NR2 format (real numbers without exponents such as 53.5)
@3	–	IEEE 488.2 NR3 format (real numbers with exponents such as 5.35E1)
@H	–	IEEE 488.2 standard hexadecimal number format (such as #H3f4e)
@Q	–	IEEE 488.2 standard octal number format (such as #Q377)
@B	–	IEEE 488.2 standard binary number format (such as #B01101100)

When writing to an instrument, any number can be converted into any of the above formats by specifying the format desired. When reading from an instrument, any of the above formats can be read and automatically deciphered (the data format does not have to be specified). Thus, if a number is read from an instrument, the controller does not have to know what format the number will be in because the library will determine the number's format and act appropriately.

An optional buffering mechanism is available that can dramatically improve the performance of instrument communication (especially to VXIbus instruments). When buffering is enabled on writes, characters sent to the instrument are buffered until an END indicator (or newline) is given and then the entire buffer is written to the device. The END indicator for output data corresponds to the end of a standard IEEE 488.2 program message. When buffering is enabled on reads, all characters are read from the device up to the END indicator and buffered in the controller. The END indicator for input data corresponds to a complete IEEE 488.2 response message. The controller then uses the buffer to satisfy data read requests from the application.

This buffering works fine with all IEEE 488.2 and SCPI-based instruments. For older instruments, buffering can be disabled if it interferes with the instrument's functions. The sizes of both read and write buffers can be set independently for each session so that the buffers can be tuned for the requirements of individual instruments. Finally, the read and write buffers are linked so that data is flushed to and from the instrument as appropriate to maintain the IEEE 488.2 Message Exchange Protocol synchronization.

* All of the instrument library routines are prefixed with "i."

These formatting capabilities make it easy to talk to VXIbus and non-VXIbus instruments.

Low-Level Interface

The low-level interface provides a nonformatted I/O mechanism that allows fast transmission of nonformatted binary strings. This mechanism, while it cannot be used simultaneously with the high-level interface, allows greater control of the instrument I/O than the high-level interface. The low-level library routines allow sending arbitrary data of arbitrary length while controlling the END indicator. They also allow data to be received and terminated by a predetermined maximum count, an END indicator, or an 8-bit pattern character such as a line feed.

These routines provide very tight control over I/O to a particular instrument, allowing activities to be performed that aren't possible with the high-level interface. The following low-level routines are available:

- iwrite. Send a block of data to an instrument. The END indicator can be optionally set on the last byte.
- iread. Read an arbitrary-length block of data from an instrument. The read can be terminated by the number of bytes read, the END indicator, or a pattern character such as a line feed. This routine will return the reason the read terminated.
- itermchr. Set the 8-bit pattern character that causes an iread to terminate.
- inbwrite and inbread. These are the nonblocking equivalents of iread and iwrite.

Memory Mapping and Register-Based Cards

The library provides routines to control register-based VXIbus cards and other memory-mapped devices. The imap routine maps an arbitrary section of VXIbus memory into an application's data space and the routine iunmap removes memory mapping. Once a section of VXIbus memory is mapped into an application's data space, the VXIbus memory can be accessed just like other parts of the application's data space. In particular, normal C pointer arithmetic can be used to read and write the registers of VXIbus instruments.

Using register-based instruments this way allows extremely fast access to instruments from a measurement application. This is because the operations needed to program a register-based instrument are simply register reads and writes—operations that take microseconds to perform. This is opposed to message-based instruments that require several milliseconds to parse and execute the requested ASCII commands. Using register-based devices in this way, while not as easy as using a message-based SCPI instrument, can allow measurements to complete 10 to 1000 times faster than message-based instruments.

The following program uses imap to talk to a VXIbus register-based voltmeter.

```
typedef unsigned short word;
struct dvm_data {
    word id;
    word devtype;
    word stat_ctrl;
    word offset;
    word hold[5];
    word range;
    word measure;
    double result;
};
main(){
    INST dvm;
    double res;
    dvm_data *dvmp;

  /*Print message and terminate on error */
    ionerror(I_ERROR_EXIT);
    /*Open the voltmeter */
    dvm=iopen("voltmeter");

    /* Get pointer to registers */
    dvmp=imap(dvm,I_MAP_VXIDEV,0,1,0);

    /* Set the range */
    dvmp->range=0x3d4e;

    /* Take a measurement */
    dvmp->measure=1;
    /* Wait for it to complete */
    while(dvmp->measure!=0);
    /* Read the result */
    res=dvmp->result;

  /* Print the results */
  printf("Result is %f\n",res);
}
```

The following mapping options are available for imap:

- I_MAP_A16. Maps in a section of VXIbus A16 address space.
- I_MAP_A24. Maps in a section of VXIbus A24 address space.
- I_MAP_A32. Maps in a section of VXIbus A32 address space.
- I_MAP_VXIDEV. Maps in the 64-byte device registers for the given instrument. These are the device configuration registers in A16 space (see article on page 41).
- I_MAP_EXTEND. If the given device has A24 or A32 address memory, then this routine maps in some portion of this device's A24 and A32 address memory. The location of this device's extended memory is determined by reading the identifier and offset registers in the device's 64-byte device registers. This can be very useful in locating a device's A24 or A32 address space without having to refer to configuration tables or resource manager files.

By using these routines, all or a portion of the given address space can be mapped into a process at any given time. However, hardware in the VXIbus controller can limit the number of simultaneous mappings available. This is because of the limited number of mapping windows available in a given VXIbus controller. If an application needs to work with several different controller types but still wants to take advantage of as many map windows as there are available on a given controller, it needs to modify its mapping requests based on the hardware resources available.

To get this information, the imapinfo routine is used. This routine returns, for a given address space, the number of map windows the hardware provides and the maximum size of these windows. This information can be used by an application to manage the mapping and unmapping of large chunks of VXIbus memory.

Interrupts

The VXIbus instrument library enables a process to be informed of asynchronous events happening on the VXIbus interface. There are core interrupt conditions that can occur on any type of interface, and there are interrupt conditions specific for the given type of interface such as VXIbus. The following is the list of core interrupt conditions and what they mean:

- STB. The commander just read a status byte.
- DEVCLR. The commander sent a device clear.
- DARRIV. The commander has sent data.
- DREQ. The commander has read some data.
- INTFACT. The interface has just become active. For VXIbus, this means normal operation, and for the HP-IB, this means that the instrument receives control of the bus.
- INTDEACT. The interface has just become inactive. For VXIbus, this means the instrument is no longer in normal operation because of either a soft or a hard reset. For the HP-IB, this means that the instrument has passed control to another instrument.
- TRIG. A trigger has occurred. For VXIbus, this could be any of the TTL or ECL trigger lines.

The following interrupt conditions are VXIbus-specific:

- LLOCK. A lock or clear lock word-serial command has arrived.
- SIGNAL. A write to the signal register or an interrupt occurred from an application's VXIbus servant device, and the value returned indicates an event signal that the library did not handle.
- VXI. An interrupt occurred from a VXIbus device that is not one of the application's servants.
- SYSRESET. A VXIbus system reset has occurred.

A program can be set up either to wait for an interrupt to occur, or to have a procedure executed when an interrupt arrives (or both). The ionintr, isetintr, and iwaithdlr routines work together to handle interrupts. Also, the ionsrq routine can be used to process service requests (SRQs) from a device. The following program handles interrupts.

```
#define on 1
#define off 0
void myhandler(INST id,long reason,long sec){
  printf("An interrupt occurred!\n");
  /* See what caused the interrupt */
  switch(reason){
      case I_INTR_TRIG:
        /* 'sec' contains the trig */
        /* line that fired. */
        break;
      case I_INTR_VXI_SIGNAL:
        /* 'sec' contains the value */
        /* written to the signal */
        /* register. */
        break;
      case I_INTR_VXI_VME:
        /* 'sec' contains the 'iack' */
        /* value that was read while */
        /* acknowledging the VME */
        /* interrupt. */
        break;
  }
}
main(){
  INST dvm;
  double res;

  /* Print message and terminate on error */
    ionerror(I_ERROR_EXIT);
  /* Open the voltmeter */
    dvm=iopen("voltmeter");

  /* Install the interrupt handler */
    ionintr(dvm,myhandler);

  /* Enable the appropriate interrupts */
    isetintr(dvm,I_INTR_TRIG,
          I_TRIG_TTL2|I_TRIG_TTL5);
    isetintr(dvm,I_INTR_VXI_SIGNAL,on);
      isetintr(dvm,I_INTR_VXI_VME,on);

  /* Wait for an interrupt to occur */
  /* with a 10 second timeout */
  iwaithdlr(10);

  /* Continue with other processing */
  ...
}
```

Besides the above capabilities, the iintron and iintroff routines are available to disable interrupts globally. This allows developers to create critical sections of code. When interrupts are disabled with iintroff they are queued until an iintron occurs. This prevents interrupts from being lost during the execution of critical sections. Also, the iwaithdlr routine will automatically reenable interrupts when it is called. Using iintroff or iintron with iwaithdlr allows interrupts to be queued until iwaithdlr is called, preventing iwaithdlr from missing an early interrupt.

Multiprocess Locking

The instrument library is designed to work on multiprocessing systems such as HP-UX.* In such a system, many programs can try to access a given device simultaneously, which can cause instrument contention problems to occur. To prevent such problems, a locking mechanism is available. This locking mechanism allows one process to grab control of an instrument and lock out other processes.

The ilock routine will lock the given instrument or interface to the calling process. The calling process can access the instrument or interface, but all other processes will be prevented from accessing the instrument or interface. The iunlock routine will remove a lock.

When a process attempts to access an instrument that is locked by another process, one of two things will happen. Either the call used to access the instrument will return an error, or the call will block until the instrument is no longer locked. The isetlockwait routine is used to set the action to be taken.

Error Handling

The instrument library provides a convenient mechanism for handling errors. This mechanism has substantial advantages over other I/O libraries because error handling code is located away from the heart of the test program. This makes reading and understanding the test program easier. Typically in test programs error handling code is intermixed with the test code as in the following test program.

```
main(){
  INST dvm;
  double resp;
  int res;

  if((dvm=iopen("voltmeter"))==NULL){
      printf("Error occurred in iopen\n");
      printf("Error:%s\n",ierrors[ierrno]);
      exit(1);
  }
    res=iprintf("MEAS:VOLT:DC?\n");
    if(res<0){
        printf("Error occurred in iprintf\n");
        printf("Error:%s\n",ierrors[ierrno]);
        exit(1);
    }
    res=iscanf("%f",&resp);
    if(res<0){
      printf("Error occurred in iprintf\n");
      printf("Error:%s\n",ierrors[ierrno]);
      exit(1);
    }
    printf("Voltage is %f\n",resp);
}
```

Notice that code is inserted after every I/O call to check to make sure the call completed successfully. In this example, if an error occurs, a simple message is printed along with the error that occurred and the program is terminated.

A test program that uses the enhanced error handling mechanism provided by the instrument library would look like:

```
main(){
  INST dvm;
  double resp;

  /* Install an error handler */
  /* Use a pre-defined one */
  ionerror(I_ERROR_EXIT);

  dvm=iopen("voltmeter");
  iprintf("MEAS:VOLT:DC?\n");
  iscanf("%f",&resp);
  printf("Voltage is %f\n",resp);
}
```

Notice that no special error handling code is inserted between I/O calls. Instead, a single line at the top (calling ionerror) installs an error handler that gets called any time an error occurs. In this example, a standard, system-defined error handler is installed that prints an error message and terminates. However, a user-supplied error handling procedure can be specified as well. With this enhanced error handling mechanism, not only does the test program become shorter, but it also becomes easier to write, read, and understand.

If the user installs a custom error handler, the error handler is passed the error number that occurred and the INST identifier of the session that generated the error. With this information, the error handler can perform actions such as:

- Talking to the instrument to get more detailed error information
- Printing a reasonable error message based on the error number
- Setting a flag that is checked by the main program for error conditions
- Using setjmp, longjmp, or HP's try/recover mechanisms to pass the error back to an error block within the main test program
- Accessing session-specific data used by the main test program using the isetdata and igetdata routines, which are described below
- Attempting to recover from the error.

If the test program is compiled using an ANSI C compiler, then a debugging mode can be enabled that provides more information to the error handler, such as:

- A string containing the name of the library routine that created the error
- The name of the source file of the test program that contained the call to the routine that failed
- The line number in the source file that contains the routine that failed.

When using the standard I_ERROR_EXIT error handler, this information and the error that occurred are printed before the application is terminated. This information can be extremely useful in tracking down how and where an error occurred.

Other Functionality

The library contains many other routines that provide more functionality, too many to discuss in this paper. The following types of additional functionality are provided in the library:

- Device and interface clear and reset functions
- Read and set the status byte (STB)
- Triggering activities
- Control of data transfer preferences such as DMA, polled, and interrupt driven I/O
- Function timeouts
- Access to interface and device status information.

The library also provides the isetdata and igetdata routines, which allow a test application to store and retrieve application-specific data. This data is stored and is available on a session-specific basis, so one part of a test program can store data that is used by another part of the test program. This data can be stored and sorted on an instrument-by-instrument basis. This is especially useful for error handler procedures and interrupt handler procedures. Any application-specific data can be stored with the session, including things such as the instrument state, measurement statistics, and current switch fixture settings.

Putting it Together

The following program uses many of the features discussed in this paper. It includes reading and writing, register-based access, interrupts, locking, and error handling.

```
#define on  1
#define off 0
struct dvm_data {
  word id;
  word devtype;
  word stat_ctrl;
  word offset;
  word hold[5];
  word range;
```

```
word measure;
double result;
};
/*
 * Error Handler
 */
volatile int error_escape=0;
void errhdlr(INST id,int error){
  /* try/recover/escape handling */
  if(error_escape)
      escape(error);
    /* Otherwise, print and exit */
    printf("ERROR Occurred!\n");
    printf("Error: %s\n",ierror[error]);
    printf("Filename: %s\n",debug_file);
    printf("Linenum=%d\n",debug_line);
    exit(1);
}
volatile unsigned short vmeiack;
void intrhdlr(INST id,long reason,long sec){
    /* See if it was a VME interrupt */
    if(reason==I_INTR_VXI_VME){
        vmeiack=sec;
    }else{
        printf("Invalid Interrupt\n");
        exit(1);
    }
}
main(){
    INST src,dvm1,dvm2;
    dvm_data *dvm2p;
    double volt,amp;

/* Install error handler */
ionerror(errhdlr);

    /* Open the devices */
    src=iopen("source");
    dvm1=iopen("voltmeter");
    dvm2=iopen("ammeter");

    /* Lock the instruments */
    ilock(src);
    ilock(dvm1);ilock(dvm2);

    /* Get registers for register-based */
    /* ammeter */
    dvm2p=imap(dvm,I_MAP_VXIDEV,0,1,0);

    /* Setup an interrupt handler */
    /* for register-based ammeter */
    ionintr(dvm2,intrhdlr);
    isetintr(dvm2,I_INTR_VXI_VME,on);

    /* Turn on the source */
    /* Catch error if occurred */
    try{
      error_escape=1;
      iprintf(src,"SOUR:DISABLE\n");
      iprintf(src,"SOUR:FREQ 100\n");
      iprintf(src,"SOUR:AMP 25\n");
      iprintf(src,"SOUR:WIDTH 20\n");
      iprintf(src,"SOUR:DURATION 30\n");
      iprintf(src,"SOUR:ENABLE\n");
    }recover{
      printf("Can't set up source\n");
      printf("Continuing with test\n");
      iprintf(src,"SOUR:RESET\n");
    }
    error_escape=0;

    /* Take voltage reading */
    /* Terminate if error occurs */
    ipromptf(dvm1,"MEAS:VOLT:DC?\n","%f",&volt);

    /* Take current reading */
    /* Device is register-based */
    iintroff(); /* Set up critical section */
    dvm2p->range=0x15d2;
    dvm2p->measure=1;
    vmeiack=0;
    while(vmeiack==0){
      iwaithdlr(0);/* Wait for interrupt */
    }
    iintron(); /* Finish critical section */
    amp=dvm2p->result;

    /* Print the results */
    printf("Voltage = %f\n",volt);
    printf("Current = %f\n",amp);
}
```

Conclusion

This library represents a major improvement over previous I/O library designs. Unlike many other libraries, it provides features and ease of use required specifically for instrument control. With this library, any test engineer familiar with the C programming language can rapidly create fast, efficient test applications that are easy to understand and support.

HP-UX is based on and is compatible with UNIX System Laboratories' UNIX* operating system. It also complies with X/Open's* XPG3, POSIX 1003.1 and SVID2 interface specifications.

UNIX is a registered trademark or UNIX System Laboratories Inc. in the U.S.A. and other countries.

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

Achieving High Throughput with Register-Based Dense Matrix Relay Modules

With an onboard FIFO buffer and register-based programming, HP's VXIbus dense matrix relay modules provide high throughput and a downsized, low-cost solution to matrix switching.

By Sam S. Tsai and James B. Durr

Matrix switching, through which several instruments can be connected to multiple devices under test (DUTs) selectively, is a popular switching technique used in electronic test. In traditional rack-and-stack systems, matrix switching requires large amounts of rack space and is costly because of the amount of hardware required. The HP E1465A, HP E1466A, and HP E1467A dense matrix relay modules described in this article provide a downsized, low-cost solution to matrix switching. These VXIbus matrix switching modules each occupy one C-size VXIbus mainframe slot, and with 256 relays per module are one of the highest-density switch modules available.

These devices are suited for VXIbus register-based systems and register-based programming. In addition to their design, this article covers relay module programming and provides benchmarks of throughput speeds achieved with ASCII message-based and register-based programs.

Dense Matrix Relay Modules

The dense matrix relay module shown in Fig. 1 is a two-wire, 256 true crosspoint matrix switch. True crosspoint means that any row can be connected to any column simultaneously. In this design, four 4-by-16 submatrices have been implemented on the main printed circuit board with 256 latching relays. Terminal cards convert the submatrices into 4-by-64 (HP E1466A), 8-by-32 (HP E1467A), or 16-by-16 (HP E1465A) matrices. The terminal card also provides screw terminals to connecting the DUT.

Latching Relays

There are several advantages to using latching relays. First, with 256 relays on the dense matrix module, latching relays prevent excessive current from the power supply if the user closes too many relays accidentally. Second, energy is saved since power is not continually

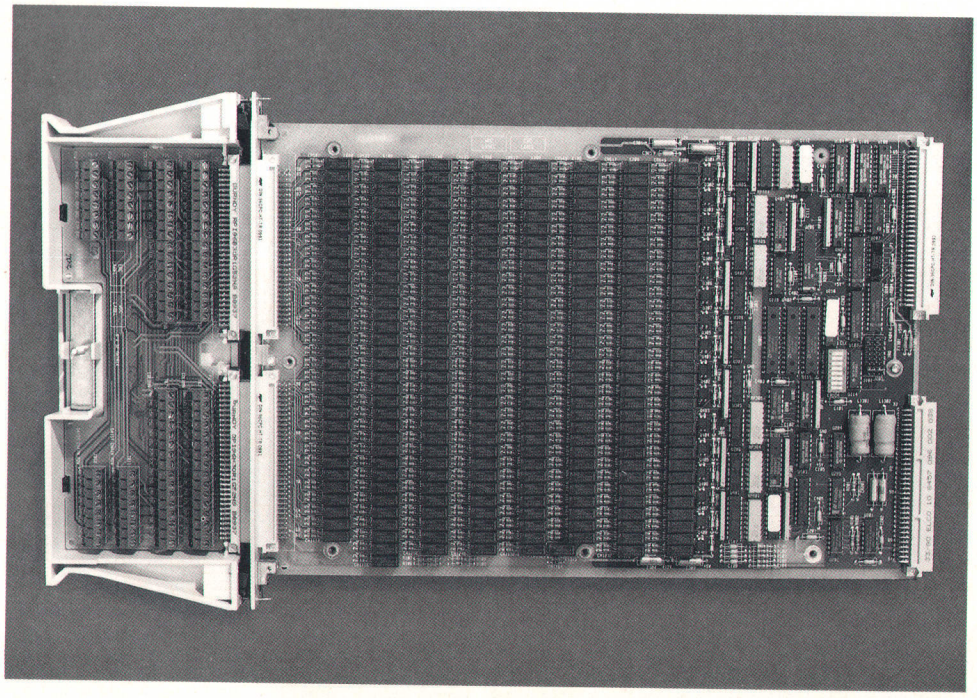


Fig. 1. A dense matrix relay module.

applied to keep a latching relay closed. Third, because power is not continually applied, the relay coil does not heat up. This is important because the two metal contacts inside the relay in effect form a thermocouple. Thus, temperature differences on the relay contacts cause a thermal EMF to be generated. Also, the life of the latching relay is usually longer than the nonlatching relay because of the power that must be continually applied to close a nonlatching relay.

The primary disadvantage of latching relays is that the relay state is unchanged at power-on, power-off, or reset. Therefore, the device's firmware must ensure that all relays are open following these conditions.

Matrix Topology

An example of the relay module's matrix arrangement is shown in Fig. 2. Represented is the matrix for the HP E1467A 8-by-32 dense matrix relay module. The 8-by-32 matrix is formed by connecting together the rows of submatrices A and C, the rows of submatrices B and D, the columns of submatrices A and B, and the columns of submatrices C and D.

On the 8-by-32 dense matrix relay module, the rows in submatrices A and C are rows 0 through 3, and the rows in submatrices B and D are rows 4 through 7. The columns in submatrices A and B are columns 0 through 15, and the columns in submatrices C and D are columns 16 through 31.

High-Throughput Design

In conventional VXIbus switch module designs, the module interrupts its commander's CPU each time a relay is opened or closed. The interrupts cause the CPU to take more time to service other instruments, which decreases system throughput. The dense matrix relay modules make efficient use of the CPU by incorporating first-in-first-out (FIFO) memory blocks. When a function and channel list are sent to the relay module, both are downloaded into the FIFO memory. The CPU is then free to do other tasks. Only after the last channel in the list is opened (or closed) is the CPU interrupted.

To show the advantage of the FIFO buffer consider a reset of the switch module. Following the reset, each channel relay must be open. Because latching relays are used, firmware must write data to each channel to ensure that it is open. Without the FIFO memory, the CPU would be interrupted 256 times (once for each relay) during the reset sequence.

How the Dense Matrix Relay Module Functions

Fig. 32 and the following sequence describe how the switch relay module operates:

- A command is sent to the relay module and stored in FIFO memory.
- Once the data is in memory, the VMEbus timing PAL (programmable array logic) asserts the signal DTACK*. This signals the CPU on the relay module's commander that it is now free to service other instruments.
- The VMEbus timing PAL signals the FIFO interface PAL to execute the command. During execution, the data bus FIFO Empty* flag signals the FIFO interface PAL to read the data bus and address bus FIFOs and generate 7-ms pulses to activate the relays. Only one 7-ms pulse is required per relay bank (16 relays).
- The FIFO Interface PAL reads the data bus and address bus FIFOs until the Empty* flag signals the FIFO interface PAL that the FIFO memory is empty.
- When the FIFO is empty, the FIFO Interface PAL signals the VMEbus timing PAL which asserts IRQ*. This interrupts the command module CPU after the last relay has been activated.

Because the relay module only asserts IRQ* after the last relay is activated, the CPU is not continually interrupted, thus enhancing system throughput.

Programming the Dense Matrix Relay Modules

In a VXIbus system there are message-based and register-based devices. Message-based devices have an onboard microprocessor, which interprets ASCII command strings and returns ASCII formatted results. Register-based devices, such as the dense matrix relay modules, do not have an onboard processor. Communication with these devices is through access to the device registers. How the registers are accessed affects system throughput.

SCPI Programming

One way to program the dense matrix relay modules is with high-level SCPI (Standard Commands for Programmable Instruments) commands. SCPI is an ASCII-based instrument command language designed for electronic test and measurement instruments (see the article on page 15 for a discussion of SCPI commands). SCPI defines standard sets of commands that allow different devices doing the same functions to be programmed with the same commands.

Programming the relay modules with SCPI commands requires the HP E1405 command module. Instrument drivers in the command module convert the SCPI commands to register "peeks" and "pokes." The command module enables the relay modules to be programmed the same as message-based devices.

In a program using SCPI commands, a relay module command is sent over the HP-IB to the command module (Fig. 4). The command statement includes the relay module's HP-IB address, the command, and the data. The format of a typical statement is

```
OUTPUT 70908;"CLOS (@10000)",
```

which is a command from an HP 9000 Series 200 or 300 computer (select code 7) to the relay module at HP-IB secondary address 8 via the HP E1405 command module at the HP-IB primary address 9. (For HP VXIbus systems, the HP-IB secondary address is defined as the logical address divided by 8.)

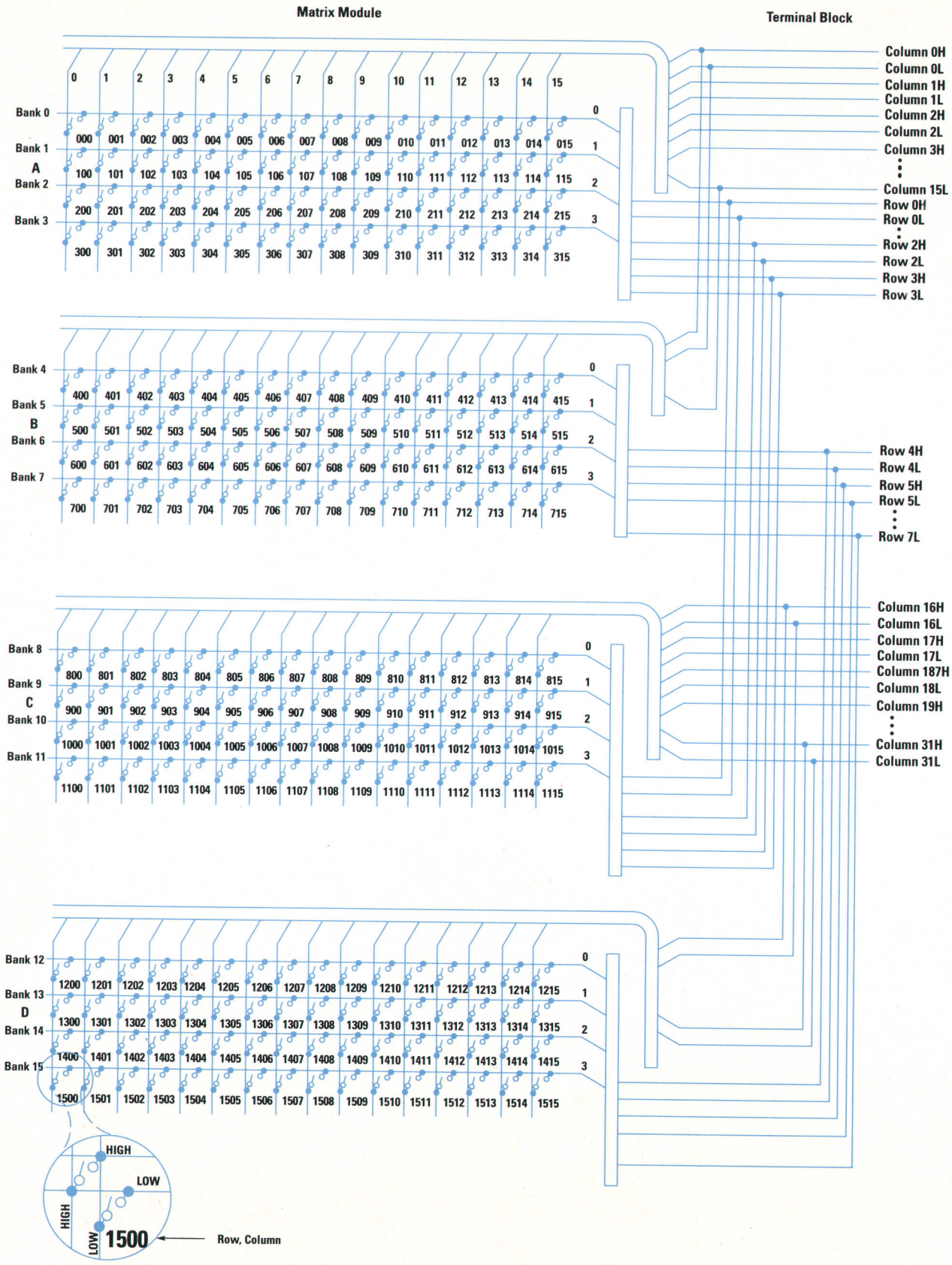


Fig. 2. An 8-by-32 relay matrix topology.

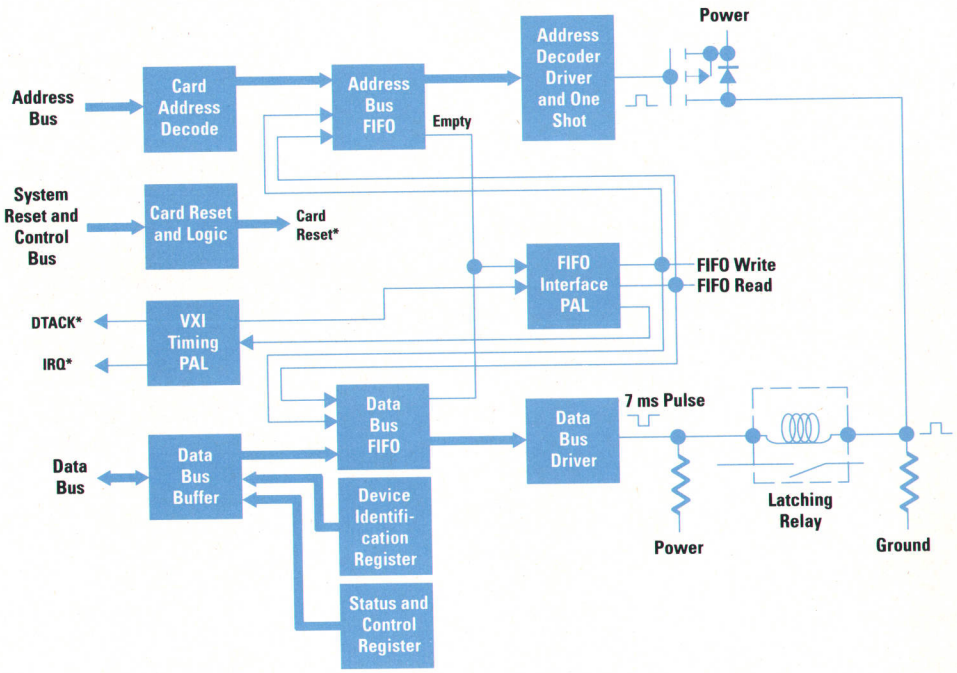


Fig. 3. Functional block diagram for the HP E1465A, HP E1466A, and HP E1467A dense matrix relay modules.

The command module's HP-IB driver stores the ASCII text CLOS (@10000) in the input buffer of the relay module, and signals the parser that the input has a command to be parsed (interpreted). The command (CLOS) is reduced to a call (with a parameter of 10000) to a routine that writes register data to the FIFO memory on the relay card.

The SCPI commands most often used with the relay modules are:

[ROUTe:]CLOSe (@channel_list)
[ROUTe:]OPEN (@channel_list)
[ROUTe:]SCAN (@channel_list)

Channel lists are specified in the format nrrcc:nrrcc

where:

n = relay module number. The module number is based on the module's logical address and the number of modules being programmed. For example, if a switchbox instrument contains three relay modules (which must have consecutive logical addresses), the module with the lowest logical address is module number 1. The module with the next lowest logical address is module number 2, and so on.

rr =the row to be connected to column cc.
cc =the column to be connected to row rr.

For example, on relay module 1, to scan row 0 column 4 through row 0 column 8 (relays 4 through 8 in Fig. 2), the channel would be specified as:

SCAN (@10004):(10008)

Note that the nrrcc:nrrcc format was implemented before the format currently used by the SCPI standard: n(rr!cc:rr!cc).

SCPI Advantages and Disadvantages

The advantage of programming the relay modules with SCPI commands is that this instrument language is common to all HP matrix relay modules and easy to understand. Also, the user only needs to specify a module number, row, and column to connect a row to a column.

The disadvantage of SCPI programming is decreased throughput because of command parsing by the HP E1405 command module. However, as the benchmark programs will show, the speed of the HP E1405 command interpreter is such that significant throughput gains are only achieved by eliminating command parsing with direct register access over the VXIbus backplane.

Register-Based Programming

Register-based programming accesses the device registers directly (Fig. 5). Register programming increases throughput since it eliminates SCPI command parsing by the command module.

Locating the Registers. Register addresses for register-based devices are located in the upper 25% of the VXIbus A16

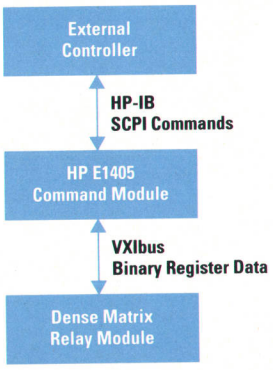


Fig. 4. SCPI communication path to the dense matrix relay modules.

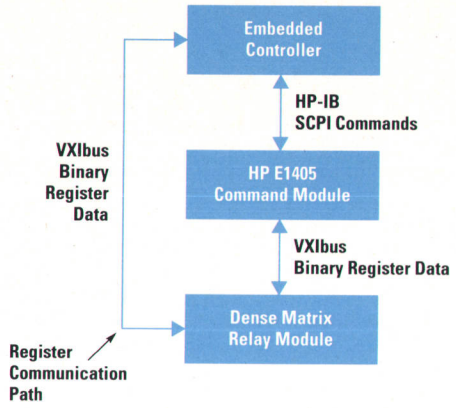


Fig. 5. Register programming communication path to the dense matrix relay modules.

address space in the embedded controller or the command module. Every VXIbus device (up to 256 per system) is allocated a 64-byte block of addresses. Depending on the number of registers a device has, the device may not use all the addresses. Fig. 6 shows how register addresses are mapped into the A16 address spaces of the HP E1480A V/360 embedded controller and the HP E1405 command module.

When programming a register-based device, a hexadecimal or decimal register address is specified. The register address is defined as:

Register Address = Base Address + Register Offset

The Base Address. The base address used to determine the register address depends on the location of the A16 address space. If an embedded controller such as the HP E1480A V/360 controller (Fig. 6a) is used, the base address is computed as:

C000h + (LADDR × 64)h
or
49,152 + (LADDR × 64)

where C000h (49,152) is the starting location of the register addresses in the embedded controller's A16 address space, LADDR is the logical address of the register-based device, and 64 is the number of address bytes per device.

If an HP E1405 command module (Fig. 6b) is used, the base address is computed as:

1FC000h + (LADDR × 64)h
or
2,080,768 + (LADDR × 64)

where 1FC000h (2,080,768) is the starting location of the register addresses in the HP E1405 A16 address space, LADDR is the logical address of the register-based device, and 64 is the number of address bytes per device.

The Register Offset. The register offset is the register's location in the block of 64 address bytes in the A16 address space. Table I lists the register offsets of the dense matrix relay modules.

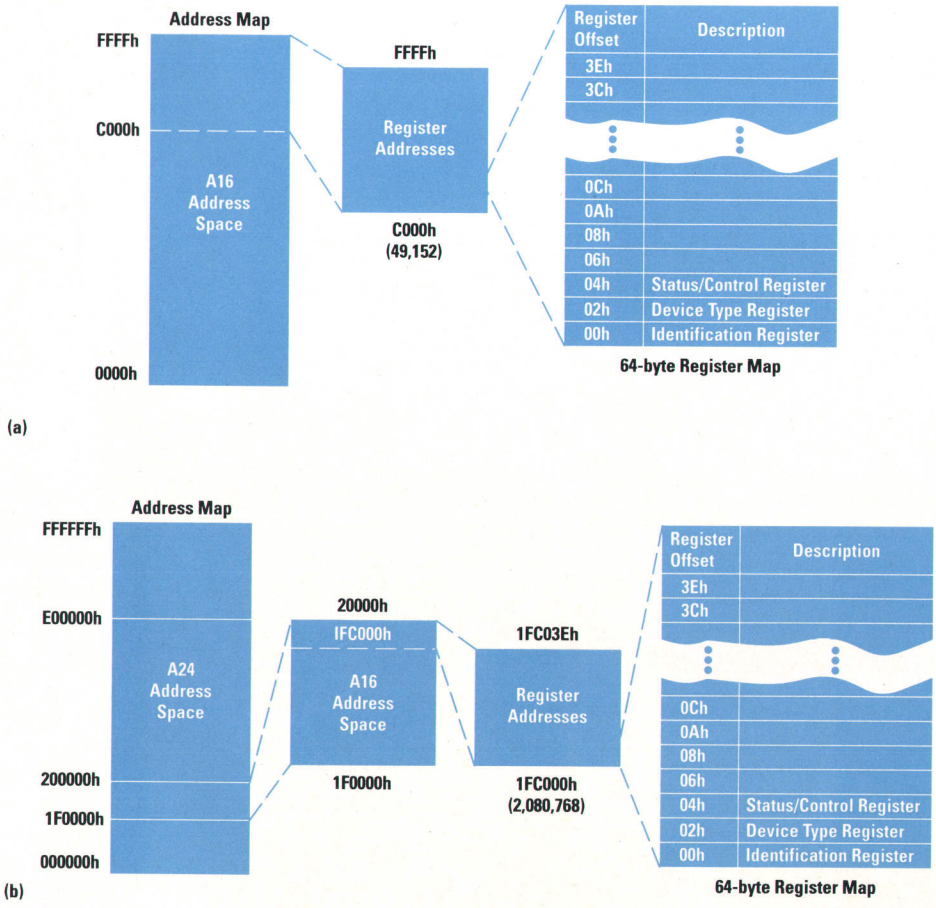


Fig. 6. Register mapping within the A16 address space. (a) A16 address mapping for the HP 1480A V/360 controller. (b) A16 address mapping for the HP 1405 command module.

When specifying the bank on which to close a relay to connect a row to a column, the offset is added to the base address to form the complete register address. For example, assuming the A16 address space is inside the HP 1480A V/360 embedded controller, the logical address of the relay module is 64, and the relay is on bank 0, the complete register addresses is:

Register Address = Module's Base Address+Register Offset
= C000h + (64 × 64)h + 20h
= D020h

Table I
Register Offsets in the Matrix Relay Modules

Register Offset	Register Name	Read/ Write	Description
3Eh	Relay Driver	Write	Bank 15
3Ch	Relay Driver	Write	Bank 14
3Ah	Relay Driver	Write	Bank 13
38h	Relay Driver	Write	Bank 12
36h	Relay Driver	Write	Bank 11
34h	Relay Driver	Write	Bank 10
32h	Relay Driver	Write	Bank 9
30h	Relay Driver	Write	Bank 8
2Eh	Relay Driver	Write	Bank 7
2Ch	Relay Driver	Write	Bank 6
2Ah	Relay Driver	Write	Bank 5
28h	Relay Driver	Write	Bank 4
26h	Relay Driver	Write	Bank 3
24h	Relay Driver	Write	Bank 2
22h	Relay Driver	Write	Bank 1
20h	Relay Driver	Write	Bank 0
**	N/A	N/A	N/A
04h	Status/Control	Read/ Write	Module Status/Control
02h	Device Type	Read	Device Type
00h	Identification	Read	Manufacturer's Identification Number

** Register Offsets 06h, 08h, 0Ah, 0Ch, 0Eh, 010h, 12h, 14h, 16h, 18h, 1Ah, 1Ch, 1Eh

Register Data. The base address and register offset specify a register's location in A16. Programming the relay modules at the register level also requires that the data that opens or closes the relay be sent. There are 16 relays on each bank on the dense matrix relay modules (see Fig. 2). These relays have corresponding bit values of 2^0 through 2^{15}. For example, to specify relay 05, 32 (2^5) or 20h would be the data value sent. Similarly, to specify all 16 relays on a bank, –1 (65,535) or FFFFh would be sent.

Register-Based Commands. The commands used to program the relay module depend on the controller used. The commands used in the benchmark programs given later in this article include:

```
WRITEIO <select code>,<register_number>,<register data>
```
and
```
DIAGnostic:POKE <address>,<width>,<data>.
```

WRITEIO is used with the HP V/360 controller and HP E1405 IBASIC (Instrument BASIC). For example, executing:

```
WRITEIO –16, Register_number,1
```

on the V/360 writes one word of data on the VXIbus backplane (select code –16) to close the relay whose address (and bank offset) are specified by the variable Register_number. Executing the command from IBASIC closes the relay whose address is specified by the variable Register_address:

```
WRITEIO –9826,Register_address,1
```

DIAG:POKE is an SCPI command that allows users to program the relay module registers without using an embedded controller. This command is executed by the HP E1405 command module. For example, the statement

```
OUTPUT 70900;"DIAG:POKE #H1FD020,16,1"
```

closes relay 0 on bank 0 of the relay module.

The command module parses the command header DIAG:POKE. However, the register address and data are written directly to the relay module's FIFO memory.

Advantages and Disadvantages. The primary advantage of register-based programming is increased throughput, which is achieved by eliminating SCPI command parsing and accessing registers from the VXIbus backplane.

The disadvantage of register-based programming is that programming at a manufacturer-specific binary level often causes the programs to be more complex than SCPI programs. Unlike specifying a relay module number, row, and column in an SCPI program, the register-based programs require the programmer to specify a register address, offset, and weighted bit (channel) value.

Benchmark Programs

The following programs measure throughput speed, which is defined here as the *time required to send a command to the dense matrix relay module and for a relay to close*. The programs, written as they might appear in an actual test system, compare typical throughput speeds obtained using SCPI and register-based programs with different controllers and programming languages. Table II summarizes the throughput speeds.

Table II
Throughput Summary

Controller	Command Module	Command	Program	Command Execution Time*
HP E1480A	---	WRITEIO	1	7.3 ms
HP E1480A	HP E1405	DIAG:POKE	2	15.0 ms
HP E1480A	HP E1405	CLOSe	3	13.5 ms
HP 9000 Model 217	HP E1405	DIAG:POKE	4	20.0 ms
HP 9000 Model 217	HP E1405	CLOSe	5	15.0 ms
HP 9000 Model 370	HP E1405	DIAG:POKE	6	15.0 ms
HP 9000 Model 370	HP E1405	CLOSe	7	13.0 ms
HP E1405/IBASIC	---	WRITEIO	8	8.5 ms
HP E1405/IBASIC	---	DIAG:POKE	9	23.0 ms
HP E1405/IBASIC	---	CLOSe	10	16.5 ms
HP Vectra/Turbo C++	HP E1405	DIAG:POKE	11	10.0 ms
HP Vectra/Turbo C++	HP E1405	CLOSe	12	13.0 ms
HP E1480A/ HP-UX 7.0	---	**	13	8.6 ms

*Each benchmark includes 7-ms busy time required for the relay to close and settle. (One 7-ms busy time is required per relay bank.)

**Direct register access in a multitasking system.

The highest throughput is achieved when SCPI command parsing is eliminated. This occurs when the relay module's FIFO memory is accessed directly via WRITEIO from the VXIbus backplane with an embedded controller or with IBASIC. Note that register-based programming using DIAG:POKE offers little if any throughput advantage over the high-level CLOSe command. This is because of the command module's fast command parser.

Benchmark 1

- Configuration: Fig. 7.
- Command: WRITEIO <select code>,<register number>, <register data>
- Language: HP BASIC/WS 6.0
- Program:

```
10  CONTROL 16,25;2 !Map the V/360 A16 addr. space for WRITEIO
20  Base_addr=DVAL("D000",16) !Convert base address to a REAL
30                            !number
40  Reg_addr=Base_addr+32  !Add register offset, store register
50                            !address
60  INTEGER I
70  T1=TIMEDATE     !Time WRITEIO
80  FOR I=1 TO 100
90  WRITEIO -16,Reg_addr;1 !Connect row 0 to column 0 on bank 0
100 REPEAT
110 UNTIL BIT(READIO(-16,Base_addr+4),7) !Monitor relay card status
120 !register bit 7 (busy bit) to determine when relay is closed
130 NEXT I
140 T2=TIMEDATE
150 PRINT "V/360 (WRITEIO): ";((T2-T1)/100.)*1.E+3;"ms"
160                           !Compute time
170 END
```

- Result: V/360 (WRITEIO): 7.2998046875 ms

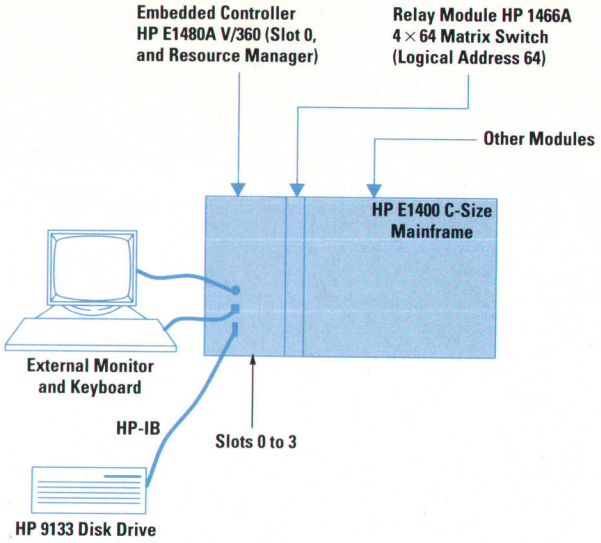


Fig. 7. Configuration for benchmark 1.

Benchmark 2

- Configuration: Fig. 8.
- Command: DIAGnostic:POKE <address>,<width>,<data>
- Language: HP BASIC/WS 6.0.
- Program:

```
10  INTEGER I                 !Declare loop counter variable
20  ASSIGN @Comm TO 70900     !Assign I/O path
```

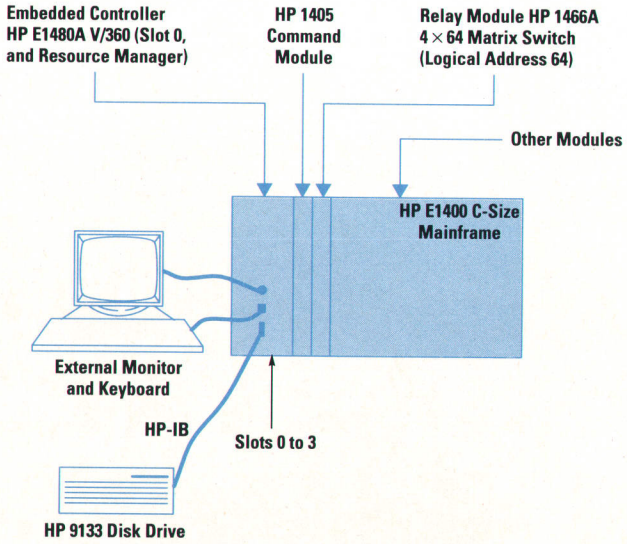


Fig. 8. Configuration for benchmarks 2 and 3.

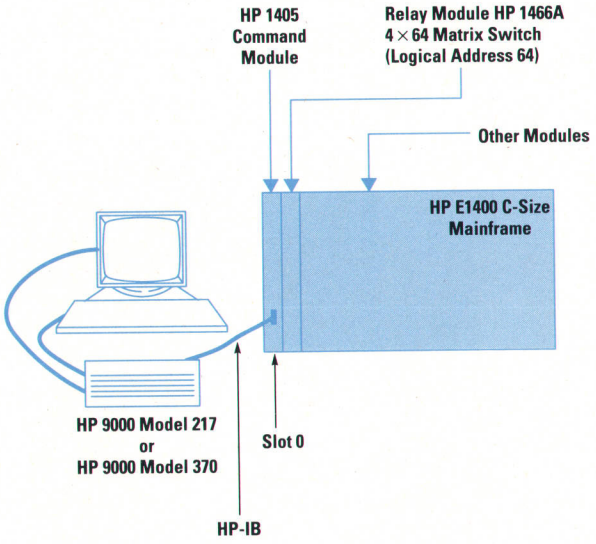


Fig. 9. Configuration for benchmarks 4, 5, 6, and 7.

```
30  T1=TIMEDATE     !Time DIAG:POKE
40  FOR I=1 TO 10
50  OUTPUT @Comm;"DIAG:POKE #H1FD020,16,1"  !Connect row 0 to
60                  !column 0 on bank 0 status bit 7 (busy).
70  REPEAT
80    OUTPUT @Comm;"DIAG:PEEK? #H1FD004,16" !Monitor relay card
90                  !to determine when the relay is closed
100   ENTER @Comm;Status
110 UNTIL BIT(Status,7)
120  NEXT I
130  T2=TIMEDATE
140 PRINT "V/360 (DIAG:POKE): ";((T2–T1)/10.)*1.E+3;"ms" !Compute time
150 END
```

- Result: V/360 (DIAG:POKE): 14.9993896484 ms

Benchmark 3

- Configuration: Fig. 8.
- Command: [ROUT:]CLOS (@channel_list)
- Language: HP BASIC/WS 6.0.
- Program:

```
10  INTEGER I                   !Declare loop counter variable
20  ASSIGN @Switch TO 70908     !Assign I/O path
30  T1=TIMEDATE     !Time CLOS
40  FOR I=1 TO 10
50    OUTPUT @Switch;"CLOS (@10000);*OPC?" !Connect row 0 to
60      !column 0 on bank 0 and Wait for relay to close (*OPC?)
70    ENTER @Switch;A
80    OUTPUT @Switch;"OPEN (@10000);*OPC?" !Wait for relay to open
90    ENTER @Switch;A
100 NEXT I
110 T2=TIMEDATE
120 PRINT "V/360(CLOS COMMAND):";((T2–T1)/20.)*1.E+3;"ms" !Compute
130                                                      !Time
140  END
```

- Result: V/360 (CLOS COMMAND): 13.4994506836 ms

Benchmark 4

- Configuration: Fig. 9.
- Command: DIAG: POKE <address>,<width>,<data>
- Language: HP BASIC/WS 6.0.
- Program: The same as benchmark 2 except for line 140.

```
140 PRINT "217/B6 (DIAG:POKE): ";((T2–T1)/10.)*1.E+3;"ms"
```

- Result: 217/B6 (DIAG:POKE): 19.9981689453 ms

Benchmark 5

- Configuration: Fig. 9.
- Command: [ROUT:]CLOS (@channel_list)
- Language: HP BASIC/WS 6.0
- Program: The same as benchmark 3 except for line 130.

```
130 PRINT "217/B6 SCPI (CLOS COMMAND): ";((T2–T1)/20.)*1.E+3;"ms"
```

- Result: 217/B6 SCPI (CLOS COMMAND): 15.0009155273 ms

Benchmark 6

- Configuration: Fig. 9.
- Command: DIAGnostic:POKE <address>,<width>,<data>
- Language: HP BASIC/WS 6.0
- Program: Same as benchmark 2 except for line 130.

```
140 PRINT "370/WS (DIAG:POKE): ";((T2–T1)/10.)*1.E+3;"ms"
```

- Result: 370/WS (DIAG:POKE): 14.9993896484 ms

Benchmark 7

- Configuration: Fig. 9.
- Command: [ROUT:]CLOS (@channel_list)
- Language: HP BASIC/WS 6.0.
- Program: The same as benchmark 3 except for line 120.

```
120 PRINT "370/WS SCPI (CLOS COMMAND): ";((T2–T1)/20.)*1.E+3;"ms"
```

- Result: 370/WS SCPI (CLOS COMMAND): 13.0004882813 ms

Benchmark 8

- Configuration: Fig. 10.
- Command: WRITEIO <select code>,<register number>,<register data>
- Language: HP IBASIC
- Program:

```
10  INTEGER I     ! Declare loop counter variable
20  Base_addr=DVAL("1FD000",16) !Convert base address to a REAL
30                              !number
40  Reg_addr=Base_addr+32 !Add register offset, store register address
50  T1=TIMEDATE           !Time WRITEIO
60  FOR I=1 TO 100
70  WRITEIO –9826,Reg_addr;1 !Connect row 0 to column 0 on bank 0
80  REPEAT
```

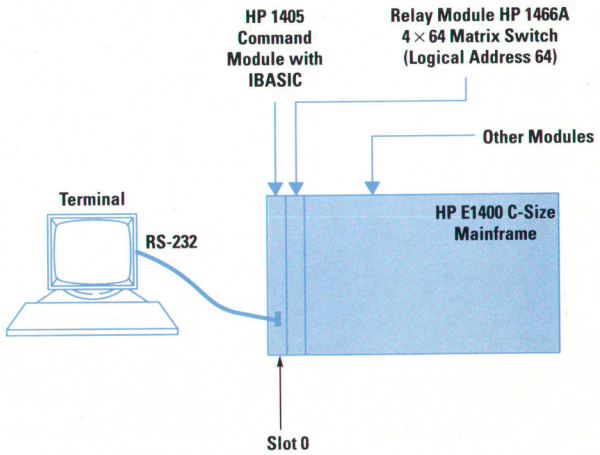


Fig. 10. Configuration for benchmarks 8, 9, and 10.

```
90  UNTIL BIT(READIO(-9826,Base_addr+4),7) !Monitor relay card status
100         !register bit 7 (busy) to determine when the relay is closed
110 NEXT I
120 T2=TIMEDATE
130 PRINT "IBASIC(WRITEIO): ";((T2-T1)/100.)*1.E+3;"ms" !Compute time
140 END
```

- Result: IBASIC (WRITEIO): 8.49975585938 ms

Benchmark 9.

- Configuration: Fig. 10.
- Command: DIAG:POKE <address>,<width>,<data>
- Language: HP IBASIC.
- Program: Same as benchmark 2 except for lines 20 and 140.

```
20   ASSIGN @Comm TO 80900. !Assign I/O path
140  PRINT "IBASIC (DIAG:POKE): ";((T2-T1)/100.)*1000.;"ms"
```

- Result: IBASIC (DIAG:POKE): 23.0009765625 ms

Benchmark 10

- Configuration: Fig. 10.
- Command: [ROUT:]CLOS (@channel_list)
- Language: HP IBASIC.
- Program: Same as benchmark 3 except for lines 20 and 120.

```
20 ASSIGN @Switch TO 80908. !Assign I/O path
120 PRINT "IBASIC (CLOS COMMAND):  ";((T2-T1)/20.)*1000.;"ms"
```

- Result: IBASIC (CLOS COMMAND): 16.4993286133 ms

Benchmark 11

- Configuration: Fig. 11
- Command: DIAGn:POKE <address>,<width>,<data>
- Language: Borland Turbo C++ Version 1.0.
- Program:

```
/* Include the following header files */

#include <stdio.h>
#include <string.h>
#include <time.h>
#include <chpib.h> /* File is in HP-IB command library */
#include <cfunc.h> /* File is in HP-IB command library */

/* Defines the command module HP-IB address */
#define ADDR 70900L
```

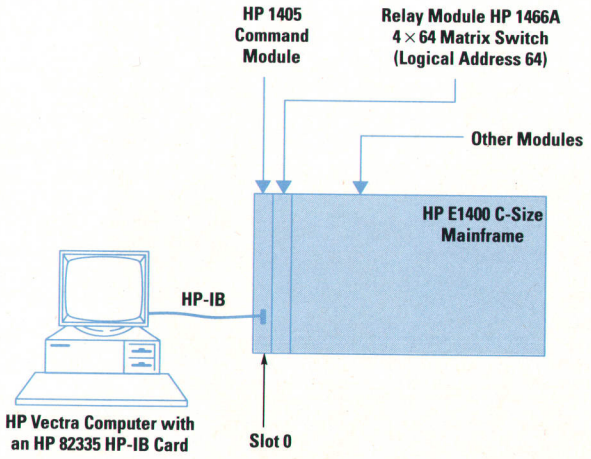


Fig. 11. Configuration for benchmarks 11 and 12.

```
/* Defines the relay modules register address */
#define DIAG_OUT "DIAG:POKE #H1FD020,16,1"
#define DIAG_IN "DIAG:PEEK? #H1FD004,16"

int main(void)
{
    time_t          T1, T2;
    int             loop;
    float           init_status, last_status = 0.;

    /* Set HP-IB timeout for error checking */
    error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");

    /* Get initial status bit value */
    error_handler(IOOUTPUTS(ADDR, DIAG_IN, strlen(DIAG_IN)),
      "OUTPUT command");
    error_handler(IOENTER(ADDR, &init_status), "ENTER command");

    /* Determine initial time of test */
    T1 = time(NULL);

    /* Start loop to determine time */
    for (loop = 1; loop <= 1000; loop ++)
    {
    /* Close relay for each loop */
      error_handler(IOOUTPUTS(ADDR, DIAG_OUT,
                strlen(DIAG_OUT)), "OUTPUT command");

    /* Determine if relay is closed before executing next loop*/
      while((init_status - last_status) != 128)
      {

    /* Read status value */
              error_handler(IOOUTPUTS(ADDR, DIAG_IN,
                    strlen(DIAG_IN)), "OUTPUT command");
              error_handler(IOENTER(ADDR, &last_status), "ENTER
              command");
      }
    /* Reset the status value for next loop */
      last_status = 0.;
    }

    /* Determine time at end of test */
    T2 = time(NULL);

    /* Calculate and display test time */
    printf("Time = %f seconds", (difftime(T2,T1) / 1000));
    return 0;
}

    /* Error checking routine */
int error_handler (int error, char *routine)
{
    char ch;
    if (error != NOERR)
    {
        printf ("\n Error %d %s \n", error, errstr(error));
        printf (" in call to HP-IB function %s \n\n", routine);

        printf ("Press 'Enter' to exit: ");
        scanf ("%c", &ch);
        exit(0);
    }
    return 0;
}
```

- Result: Time = 0.010000 seconds

Benchmark 12

- Configuration: Fig. 11.
- Command: [ROUT:]CLOS (@channel_list)
- Language: Borland Turbo C++ Version 1.0.
- Program:

```
/* Include the following header files */
#include <stdio.h>
#include <time.h>
#include <chpib.h>

    /* Defines the relay module HP-IB address*/
#define ADDR 70908L

    /* Defines the relay open command */
#define SCPI_CLOSE "CLOS (@10000);*OPC?"
#define SCPI_OPEN "OPEN (@10000);*OPC?"

int main(void)
{
    time_t T1, T2;
    int loop, length = 5;
    char into[6];

    /* Set HP-IB timeout for error checking */
    error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");
    /* Determine initial time of test */
    T1 = time(NULL);

    /* Start loop to determine time */
    for (loop = 1; loop <= 500; loop ++)
    {
      error_handler(IOOUTPUTS(ADDR, SCPI_CLOSE,
                strlen(SCPI_CLOSE)), "OUTPUT command");
      error_handler(IOENTERS(ADDR, into, &length),
                "ENTER command");
      error_handler(IOOUTPUTS(ADDR, SCPI_OPEN,
          strlen(SCPI_OPEN)), "OUTPUT command");
      error_handler(IOENTERS(ADDR, into, &length),
                "ENTER command");

     }

    /* Determine time at end of test */
    T2 = time(NULL);

    /* Calculate and display test time */
    printf("Time = %f seconds", (difftime(T2,T1) / 1000));

    return 0;
}

    /* Error checking routine */
int error_handler (int error, char *routine)
{

    char ch;
    if (error != NOERR)
{
      printf ("\n Error %d %s \n", error, errstr(error));
      printf (" in call to HP-IB function %s \n\n", routine);

      printf ("Press 'Enter' to exit: ");
      scanf ("%c", &ch);
      exit(0);

    }
    return 0;
}
```

- Result: Time = 0.013000 seconds

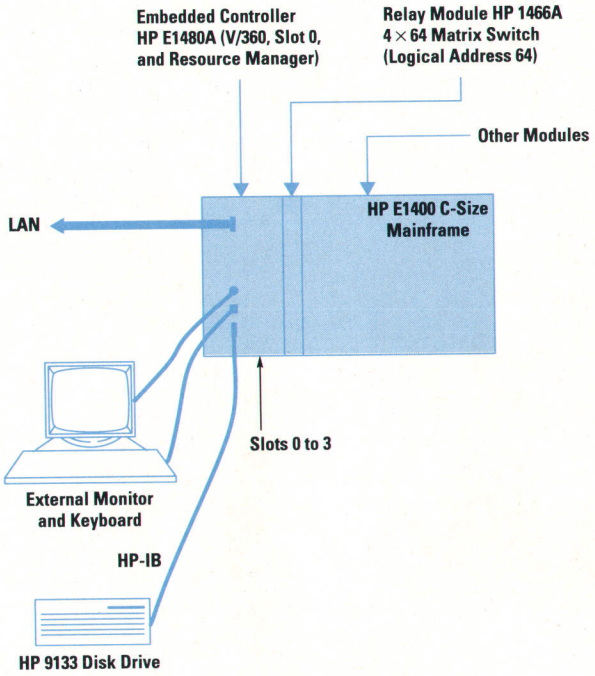


Fig. 12. Configuration for benchmark 13.

Benchmark 13

- Configuration: Fig. 12.
- Command: Direct register access.
- Language: C.
- Program:

```
#include   <time.h>
#include   "sys/vxi.h"
#include   <fcntl.h>
#include   <stdio.h>

#define LA   64
#define looptimes 5.0
main() {

int stat;
int fd;
int i;
int j;

struct dev_regs {
   unsigned short id_reg;
   unsigned short device_type;
   unsigned short status_reg;
   unsigned short dummy[13];
   unsigned short bank0_channels;
} *dev;

struct timeval first,
    second,
    lapsed;

struct timezone tzp;

/*      open the device file      */
fd=open("/dev/vxi/primary",O_RDWR);
if (fd<0) {
    perror("open");
    exit(1);
```

```
}

dev=(struct dev_regs *)vxi_get_a16_addr(fd,LA);

/* create 7-ms busy time */
   gettimeofday (&first,&tzp);
   for (j=0; j<=7000; j ++);
   gettimeofday (&second,&tzp);

if (first.tv_usec > second.tv_usec) {
 second.tv_usec += 1000000;
 second.tv_sec--;

}

lapsed.tv_usec = second.tv_usec - first.tv_usec;
lapsed.tv_sec = second.tv_sec - first.tv_sec;

printf ("Channel closing (busy) time = %ld sec %ld usec
   \n",lapsed.tv_sec,lapsed.tv_usec);
  gettimeofday(&first,&tzp);
 for (i=0; i<=looptimes; i ++)
  {
/*     Close channel 1, wait 7-ms (busy time)    */
  dev->bank0_channels=0x0001;
  for (j=0; j<=7000; j ++);
  dev->bank0_channels=0x0000;
  for (j=0; j<=7000; j ++);
  }
  gettimeofday (&second,&tzp);

if (first.tv_usec > second.tv_usec) {
 second.tv_usec += 1000000;
 second.tv_sec--;

}

lapsed.tv_usec = (second.tv_usec - first.tv_usec)/10.;
lapsed.tv_sec = (second.tv_sec - first.tv_sec)/10.;

printf("Command execution and busy time = %ld sec %ld usec
   \n",lapsed.tv_sec,lapsed.tv_usec);
/*rtprio(0,RTPRIO_RTOFF);*/
}
```

- Results: Channel closing (busy) time = 0 sec 7322 usec
  Command execution and busy time = 0 sec 8718 usec

Conclusion

The dense matrix relay modules are among the highest-density VXIbus switch modules available today. FIFO memory on the relay modules enhances system throughput by allowing the modules to interrupt their commander's CPU only after the last relay in their channel list is closed.

The relay modules can be programmed using ASCII-based SCPI commands and the HP E1405 command module, or can be programmed directly at the register level. Benchmark programs show that the highest throughput is achieved with register-based programs in which SCPI command parsing is eliminated, and the registers are accessed from the VXIbus backplane. Subtracting the 7-ms relay settling time from each benchmark shows that access from the VXIbus backplane is up to 26 times faster than the HP-IB. However, it is the command parsing time, rather than HP-IB or VXIbus speeds, that has the greatest impact on throughput.

Benchmarks using DIAG:POKE were run because of its similarity to commands supported by other command parsers. The results show that DIAG:POKE, an SCPI command that writes data directly to the relay module's FIFO memory, offered approximately the same throughput performance as the high-level CLOSe command. This is attributed to the efficient SCPI parsing algorithms of the HP E1405 command module and shows that with a fast command parser, there is little throughput advantage to register-based programming unless command parsing is eliminated entirely.

Acknowledgments

We would like to thank Ron Hanson, Chuck Platz, and Conrad Proft for the useful technical discussions. A special thanks goes to Peter Meyer for his C programs for the HP Vectra computer, and to Art Boyne and Frank Goss for their reviews.

Bibliography

1. *VMEbus Extensions for Instrumentation System Specification*, Revision 1.3, July 1989.
2. *Standard Commands for Programmable Instruments Manual*, Version 1990.0, April 1990.
3. *HP E1326B/E1411B 5 1/2-Digit Multimeter User's Manual*, Edition 1, HP Part Number E1326-90003.
4. L. DesJardin, "VXIbus versus GPIB: Is VXIbus actually faster?," *VXIbus Journal*, November 1990, pp. 11-14.

Mass Interconnect for VXIbus Systems

The HP 75000 family of VXIbus products includes a set of interconnect hardware that enables automatic test system developers to mount DUTs easily to HP's VXIbus mainframe.

by Calvin L. Erickson

The interconnect hardware in an automatic test system consists of the components that connect the device under test (DUT) to the test system mainframe. Typically, automatic test equipment (ATE) product development has focused on instrumentation, controllers, and software. The result is that the interconnect hardware in a test system has become a major component of system cost. In most test systems, interconnects consist of discrete wire and connector harnesses, and their fabrication is custom and labor intensive. Many systems also require a mass interconnect for interfacing with multiple devices under test DUTs.

The development of the VXIbus has brought these issues into sharper focus. Greater density requires that more signal lines must be packed into less space. Higher speeds require shorter lead lengths and better connectors. Finally, economics and time-to-market constraints demand standard parts suitable for many applications.

This paper discusses the development of mass interconnect products specifically for VXIbus systems. Based on the HP ATS 2000 system resource interface, these products mount directly on the front of the HP VXIbus mainframe. This paper also discusses the trade-offs involved in incorporating a mass interconnect into a VXIbus test system.

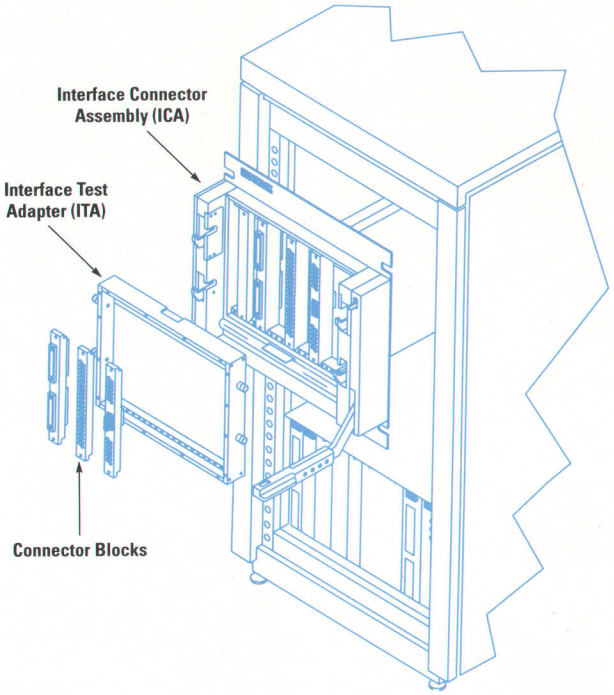


Fig. 1. The kinds of interconnect products in the HP 75000 system resource interface.

HP Interconnect Components

As part of the HP 75000 family of VXIbus instruments, a number of mass interconnect products are available for use in VXIbus test systems. These products, known as the HP 75000 system resource interface, allow a single test system to service a large variety of units under test. Fig. 1 shows the products included in the HP 75000 system resource interface.

The interface connector assembly (ICA) is the heart of the system. It serves as the primary interface between the test system and the DUT. The ICA is typically rack-mounted and wired directly to test system resources such as switching, sensors, and sources. The ICA provides locations for mounting ICA connector blocks and aligning connector blocks and their mating halves, and a mechanism for overcoming the connector mating forces. The HP 75000 system resource interface connector assemblies include:

- The HP 9420A ICA. This is the standard rack-mount version shown in Fig. 1.
- The HP E3720A VXIbus ICA. This ICA mounts directly on the front of a VXIbus mainframe. It also allows direct access to VXIbus modules and provides the greatest opportunity for short lead length.
- The HP E3722A hinged ICA. This ICA also mounts directly on the front of a mainframe and hinges down to allow access to VXIbus modules.

The interface test adapter (ITA) is the frame that mates with the ICA. It provides locations for mounting connector blocks. Typically, several ITAs are purchased, one for each type of DUT. A frame or fixture is often built on the ITA, customizing it to the particular DUT. Fig. 2 shows an ITA customized for functional testing of a printed circuit board. The HP 75000 system resource interface includes two different ITAs:

- The HP 9421A. This ITA is used with the HP 9420A ICA and the HP E3722A hinged ICA.
- The HP E3721A VXIbus ITA. This ITA is used with the HP E3720A VXIbus ICA.

Different kinds of connector blocks are available for use with the ICAs and ITAs (see Fig. 3). These blocks are wired and then installed in the appropriate location

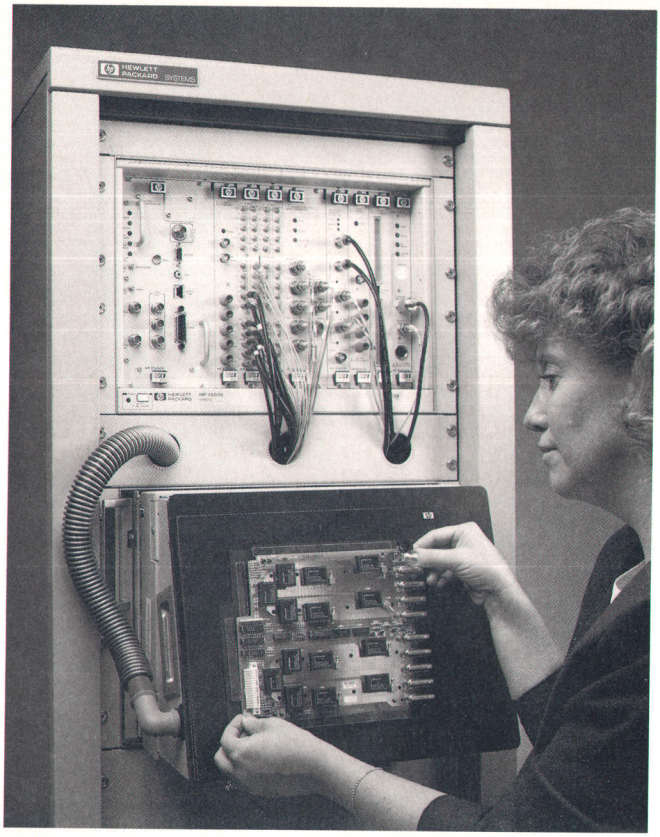


Fig. 2. A typical automatic test application in which an interface test adapter (ITA) is customized for the functional testing of a printed circuit board.

within the ICA or ITA. The three primary choices are 192-pin general-purpose, 36-contact coaxial, and 24-contact power. These connector blocks are compatible with each ICA and ITA choice listed above.

Customer Requirements

The list of customer requirements for mass interconnect in a VXIbus test system is extensive. Most of these requirements derive from the need to test a large variety of DUTs. The mass interconnect must have a variety of connectors that can handle different types of signals. These include low-level precision signals, high-frequency signals, and high-power signals. Sometimes, it is necessary to keep lead lengths as short as possible to maintain signal integrity. The connectors must have a long life to allow frequent changes of the ITA. The ITA must have a rugged and versatile construction to allow the addition of a variety of fixtures suitable for testing everything from printed circuit assemblies to automobiles.

Most test systems require many modes of wire routing. For the VXIbus, these modes include:

- VXI-to-ICA. For example, a VXIbus relay module can have many connections directly to an ICA connector block.
- VXI-to-VXI. The same relay module can have an analog bus connection to a neighboring module in the same mainframe.
- VXI-to-System. A VXIbus slot 0 module may require an HP-IB connection to the controller.
- ICA-to-System. A non-VXIbus signal generator may require connection to an ICA connector block.

Test systems must be easy to configure and reconfigure. This requires that VXIbus modules must be easy to install and remove. Wiring should be accessible. Standard cable assemblies should be available that satisfy most of the wiring requirements. Different ICA configurations should be available for mounting in front of VXIbus or standard rack mounting.

Finally, some customers require that the mass interconnect conform to industry standards such as ARINC 608 and MATE.

Product Development

The project definition dictated that we rely on existing products to satisfy customer requirements. From a practical standpoint, a lack of resources prevented us from designing a new product from scratch. Therefore, one of our major decisions concerned the selection of a vendor. There were two viable choices. One vendor had already modified their existing interconnect product line (connector block, ICA, and ITA) to fit on the VXIbus. Their design was generally sound, and contained only a few minor problems. Another vendor had a different product line that conformed to the recently emerging industry standard ARINC 608, which has become increasingly important in some segments of the marketplace. Although this product line had not been adapted to VXIbus, it was already established within HP. Both products were similar in cost, quality, and density. In the end, the decision to use the second vendor was based on two primary criteria: leverage and industry standards.

We gained leverage from previous HP engineering experience in test systems, the willingness of the vendor to provide engineering expertise, and the use of the same connector blocks across the entire family of ICAs and ITAs. HP's Advanced Manufacturing Systems Operation (AMSO) had the products set up on the HP ATS 2000 product line. This allowed us to adopt the HP 9420A ICA, the HP 9421A ITA, and the connector blocks without

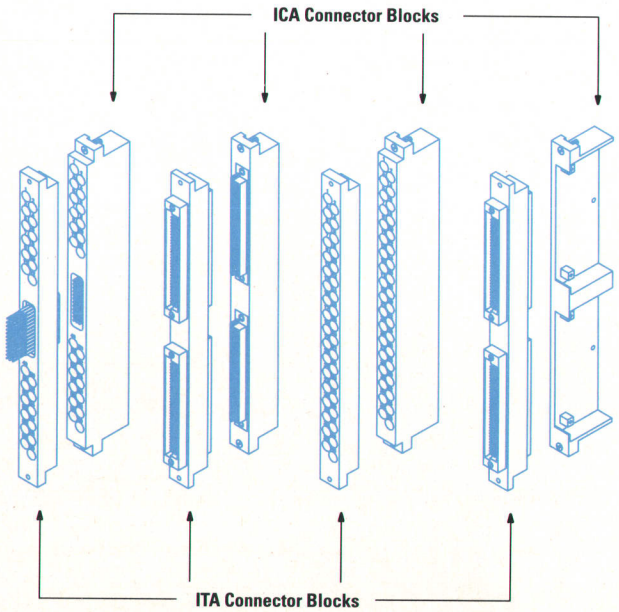


Fig. 3. ITA and ICA connector blocks.

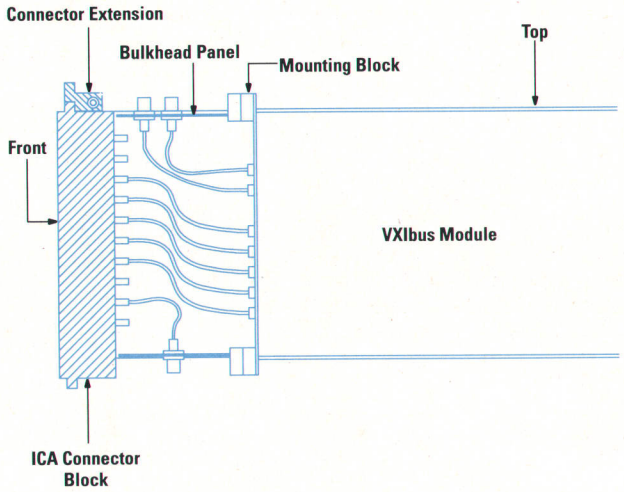


Fig. 4. An HP E3730A interface terminal module (ITM) with the cover removed.

going through full qualification testing. We still had to solve the problem that these interconnect components did not mount on the front of a VXIbus mainframe. The vendor solved this problem for us by agreeing to make modifications to the standard ICA based on our design concepts. Finally, we made the decision to use the same connector blocks across the entire family. This prevented duplicate documentation and qualification efforts.

Modifications to the standard ICA for use on the front of a VXIbus mainframe faced several design challenges. A primary design goal was to minimize the wire length between the VXIbus module and the ICA connector block. This meant that the connector block had to be assembled directly to the VXIbus module. An interface terminal module (ITM) similar to those already used in the HP VXIbus switch products was a logical choice. This interface terminal module attaches to a VXIbus module on one side and includes an ICA connector block on the other (see Fig. 4). The two ends are wired together, minimizing wire length. Note that a better solution would have been to include the ICA connector block as part of the VXIbus module. This was not practical because VXIbus does not specify the type and location of front-panel connectors. Most systems include modules from various vendors. Thus, there is no way to control all the connectors and their locations.

The next challenge was providing easy installation and removal of VXIbus modules past an ICA mounted on the front of the mainframe. The first vendor demonstrated one solution to this problem. Their ICA simply hinged down and out of the way. Although this provided excellent module access, it had two problems. The first was the requirement for very tight alignment tolerances between the mainframe and the ICA. The second was that the VXIbus module, and not the sturdy ICA frame, supported the interconnect mating forces. We decided that installing the ITM-to-VXIbus assembly directly through a stationary VXIbus ICA was a better solution (see Fig. 5). If the connector block were allowed to float within the interface terminal module, tolerances between the mainframe and the ICA could be kept loose. Also, if the connector block could be screwed into the ICA as originally designed, mating forces would be properly supported. There was only one problem: the standard ICA was not tall enough to allow a VXIbus module to pass through the opening. Increasing the height of the opening just enough to allow passage of a VXIbus module solved this problem. A connector extender was designed to increase the height of the standard connector blocks to fit in the new stretched VXIbus ICA. This modification did not affect the ICA alignment and closure mechanisms because only the top member was changed. These mechanisms reside in the bottom and sides.

Another challenge was that the standard connector blocks are on 19.05-mm (0.75-in) centers. C-size VXIbus modules are on 30.48-mm (1.2-in) centers. This was handled by simply adjusting the hole patterns in the VXIbus ICA and VXIbus ITA, making the interface terminal module 30.48 mm (1.2 in) wide, and supplying appropriate filler panels

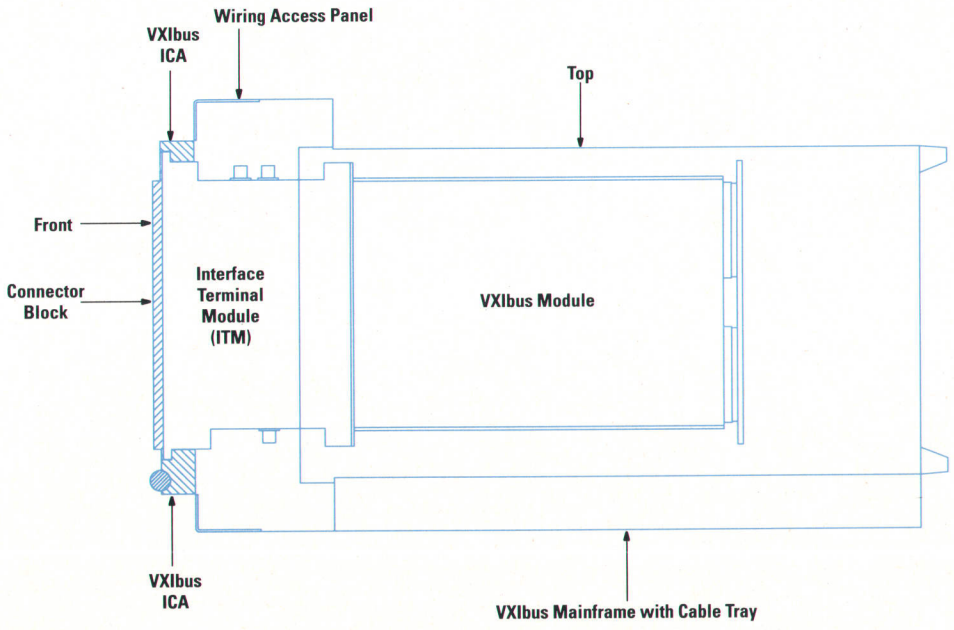


Fig. 5. An HP E3720A VXIbus ICA with an interface terminal module.

for use with the narrower connector blocks. A side effect of this is that the HP E3720A VXIbus ICA is no longer truly compatible with ARINC 608. We decided that this was an acceptable compromise if all the connector blocks were the same.

Other challenges included the need to maintain easy access to modules and to facilitate all modes of system wiring. These needs were satisfied with careful placement of the VXIbus ICA relative to the rack mounting surface and the inclusion of access panels for wiring and a special tool for the module mounting screws.

The development of the HP E3722A hinged ICA was much more opportunistic. A customer requested that we supply an ICA hinged to the front of a mainframe. For this product, interface terminal modules are not used and all wires between the VXIbus module and the ICA connector block are flexible, so that opening the hinged ICA allows access to the VXIbus modules. The vendor developed the changes based on one of their previous designs. Customer interest has remained strong and the E3722A is now a standard product (see Fig. 6).

Product Description

The three ICAs and their associated ITAs and connectors described above meet system wiring requirements in different ways. The following sections describe and illustrate these differences by showing the ICAs installed in a typical rack-mounted test system.

HP 9420A Interface Connector Assembly. The HP 9420A interconnect products are identical to those used in the HP ATS 2000 product. These include the HP 9420A ICA, the HP 9421A ITA, and three sets of connector blocks (general-purpose, coax, and power). These products are not specific to VXIbus and can be useful in any rack-mount application.

The HP 9420A ICA can be rack-mounted in any location in a test system. It conforms to ARINC 608, and provides

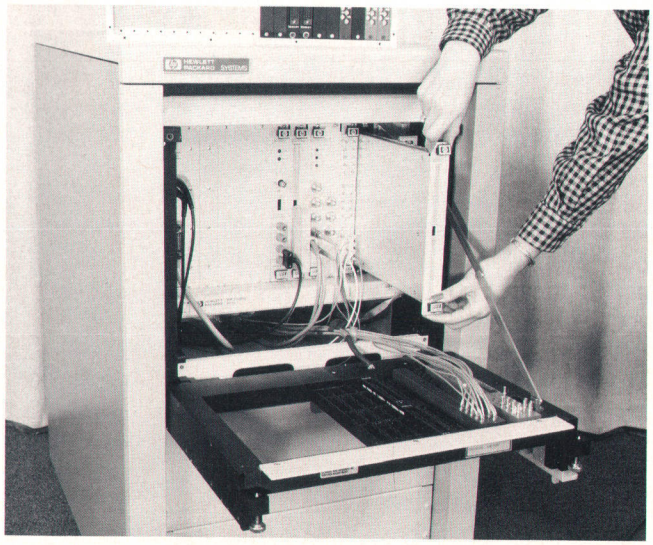


Fig. 6. An HP E3722A hinged ICA.

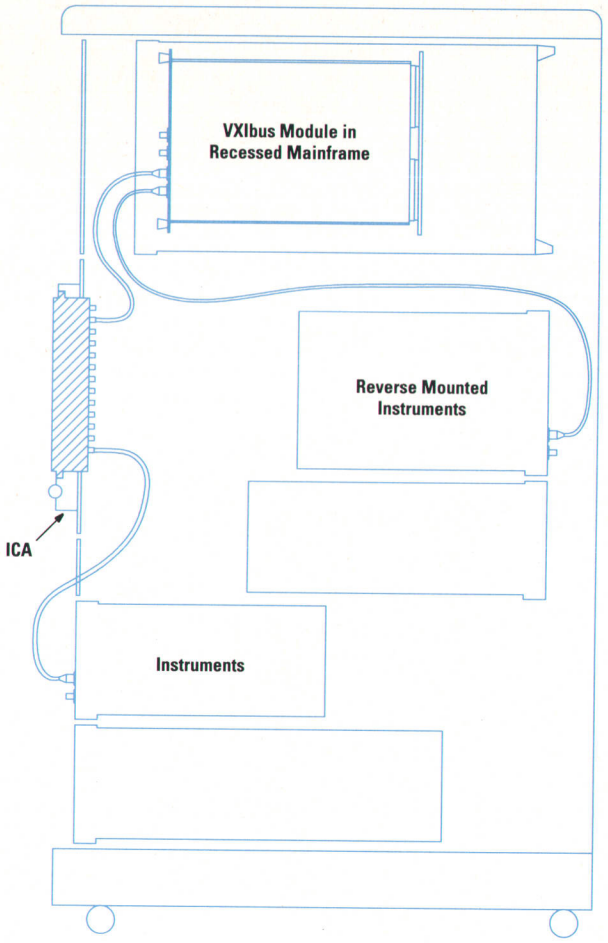


Fig. 7. Typical system wiring for an HP 9420A ICA.

21 slots on 19.05-mm (0.75-in) centers. Although it is the most versatile ICA, it may use more rack space and require longer lead lengths than desired (see Fig. 7).

The HP 9421A ITA is designed to mate with the HP 9420A ICA. It has a frame with 21 slots. Tapped holes are provided for attaching additional framework specific to each DUT. This framework and associated wiring must be provided by the customer.

The general-purpose connector blocks (ICA and ITA) include two 96-pin connectors. The coax connector blocks include positions for 36 contacts, purchased separately. The power connector blocks include 24 sense contacts and positions for 24 30-A contacts, purchased separately. All the contacts have a life of 25,000 cycles.

HP E3720A VXIbus Interface Connector Assembly (VXIbus ICA). The HP E3720A is an ICA modified to fit on the front of a VXIbus mainframe. It provides 13 positions on 30.48-mm (1.2-in) centers. Each position is aligned with the corresponding slot in the mainframe. The main differences between the HP E3720A and the standard HP 9420A ICA include:

- Flanges to position the VXIbus ICA relative to the VXIbus mainframe (these flanges also allow rack mounting anywhere in the test system)

- Access panels above and below the VXIbus ICA (removal of these panels provides access to system wiring and eases the task of module installation and removal)
- The top horizontal bar shifted up a sufficient distance to allow VXIbus modules to be inserted through the VXIbus ICA
- The bottom horizontal bar cut out to allow greater access to system wiring
- Support by both horizontal bars for 13 connector blocks instead of the standard 21
- A special tool to gain easier access to the VXIbus module mounting screws.

Fig. 8 shows wire routing in a slot that does not require an interface terminal module.

The HP E3730A interface terminal module (ITM) is a housing that mounts on the front of a VXIbus module (see Fig. 9). It is designed for use only with the HP 3720A VXIbus ICA. Features of this product include:

- Mounting blocks for mounting the ITM onto most VXIbus modules (This is accomplished by removing the handles from a module and replacing them with the blocks.)
- Standard connector blocks that mechanically float relative to the ITM (This accommodates tolerance buildup between the mainframe and the VXIbus ICA.)
- Bulkhead panels for connections to other parts of the system
- Provisions for installing terminal boards (see Fig. 10) supplied with many HP VXIbus switch products (These boards supply functions such as matrices, thermocouple reference, and so on.)
- A "funnel" for tool guidance when installing a module in the mainframe
- A pull ring to ease the task of removing modules.

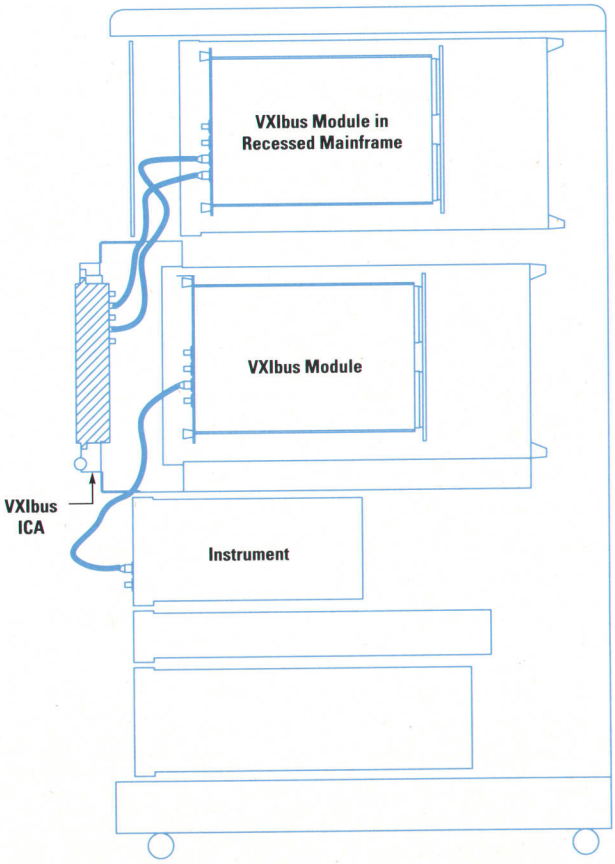


Fig. 8. Typical system wiring for an HP E3720A VXIbus ICA without an interface terminal module.

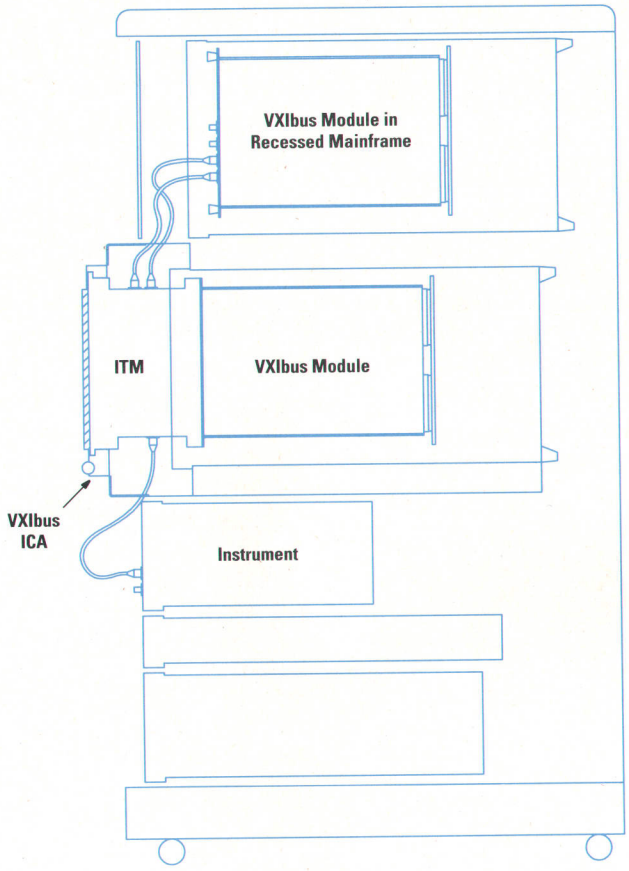


Fig. 9. Typical system wiring for an HP E3720A VXIbus ICA with an HP E3730A interface terminal module.

The E3721A VXIbus interface test adapter (VXIbus ITA) is designed to mate with the E3720A VXIbus ICA. It is identical to the HP 9421A ITA except that it supports 13 connector blocks instead of 21.

HP E3722A Hinged Interface Connector Assembly (Hinged ICA). The HP E3722A is another ICA that has been modified to fit on the front of a VXIbus mainframe. Like the HP 9420A ICA, it conforms to ARINC 608 and provides 21 slots on 19.05-mm (0.75-in) centers. The main differences between the HP E3722A hinged ICA and the standard HP 9420A ICA include:

- Flanges to position the hinged ICA relative to the VXIbus mainframe, which include hinges on the bottom and knurled screw tiedowns on top
- Tapped holes across the bottom for cable tie points.

Fig. 11 shows the wire routing with a hinged ICA.

System Integration Process

The process of including a mass interconnect in test system development is very complex. The following simplified procedure may help the system integrator through many of the issues.

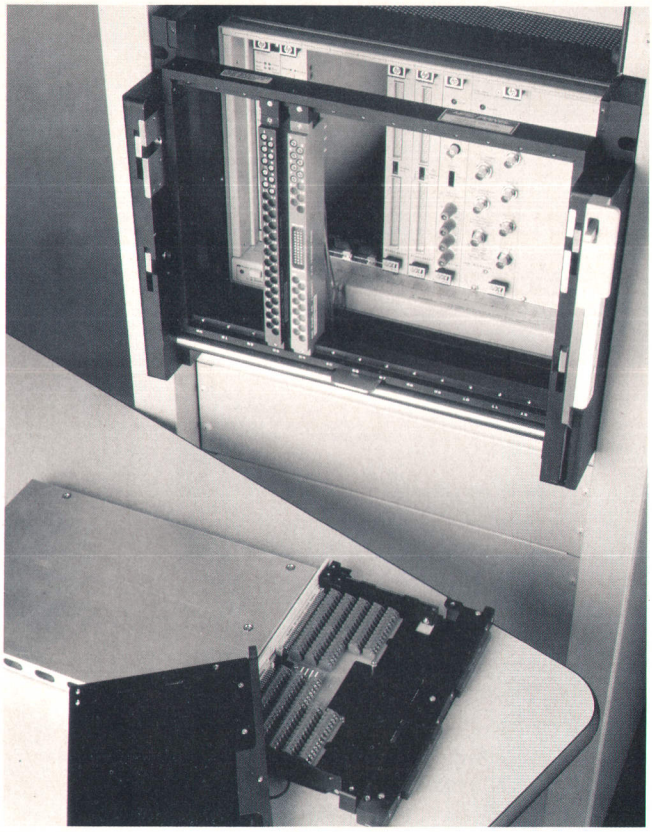


Fig. 10. In the foreground is an HP E3730A interface terminal module terminal board with the cover removed. In the background is an HP E3720 VXIbus interface connector assembly attached to an HP 75000 Series C cardcage.

1. The first step is to identify the number and type of DUTs that will be tested by the system. Usually, a different ITA will be built for each DUT, with one more for system self-test. A list of necessary system resources should be created for each ITA. Each list should include sources such as power supplies, sensors such as voltmeters and counters, and components for switching and communication. The different ITA lists can be combined into a single list of system resources required for all tests. Each item on the list will translate into a single line on the ICA.

2. The second step is to determine if a mass interconnect is necessary. The following questions should help in this decision.

- How many different DUTs will be tested? More than two or three may indicate a need for mass interconnect.
- How often will ITAs be changed? More than 400 to 500 times through the life of the system may indicate a need for mass interconnect.
- How many lines must be fed through the mass interconnect: more than 200 to 400 general-purpose lines, or more than 20 to 50 coax or power lines? If so, mass interconnect may be appropriate.
- What type of signals must be connected?
- Do appropriate mass interconnect connectors exist?

In many cases, a mass interconnect is not appropriate. A high-quality, military-style circular connector may be adequate.

3. The next step is to refine the list made in step 1. Trade-offs must be made between maximizing system flexibility, minimizing complexity and cost, and maintaining critical performance specifications. Particular attention should be given to switching networks. Consider the following:

- Custom configurations behind the ICA limit system flexibility, but they also reduce ITA complexity and cost.
- Switch networks are essential for minimizing resources and providing system flexibility, but switches have limited life. They add complexity and reduce system performance.

Add lines to the list to account for switching networks, ITA identification, and so on. To each line, attach any important performance specifications such as frequency, voltage, and current.

4. The next step is to make a preliminary layout of the entire system. In most cases, it is best to start with a block diagram. The diagram can then be used to make a sketch of a rack showing all system components. This sketch should include all sources, sensors, switches, mass interconnects, controllers, and so on. Now is the time to think about wiring issues such as grounding and critical lead lengths. Remember to leave room for growth and flexibility.

Selection of the appropriate ICA occurs at this time. This decision should be based on the above considerations and the following:

- Is compatibility required with other ITAs? These ITAs may already exist or they may be planned as part of a large project.
- Is compatibility required with an industry standard such as ARINC 608? This would limit the choice to the HP 9420A or the HP E3722A.
- What type of signals will be connected? How many lines will be connected? What is the required cycle life of the connectors? The answers to these questions may indicate the use of less-expensive interconnect products such as the HP 34592A quick interconnect.

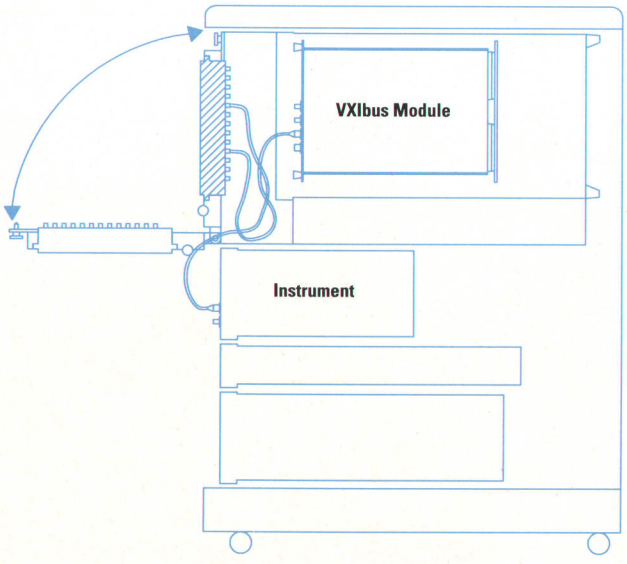


Fig. 11. Typical system wiring for an HP E3722A hinged ICA.

- Is rack space limited? If so, the HP E3720A or HP E3722A may be the best choice. Both ICAs mount on the front of the VXIbus mainframe. Remember, however, that other system components may be reverse mounted behind the HP 9420A without sacrificing rack space.
- Is a reduction in access to VXIbus front panels acceptable? Any ICA mounted in front of the VXIbus mainframe will limit access to VXIbus modules. Front-panel indicators will not be visible and modifying modules and wiring will be more difficult.
- Is short lead length between the VXIbus modules and the ICA connector important? If so, the HP E3720A may be the best choice.
- Do most of the ICA wires (at least 60 to 75 percent) connect to VXIbus modules directly in a single mainframe? If so, the HP E3720A may be the best choice. If there are many wires that connect elsewhere in the system, the HP 9420A could be a better choice.
- Are the limitations of the HP E3722A hinged ICA acceptable? A practical limitation in density is imposed by the requirement that all wires must bend every time the ICA is hinged down. Because the hinge is at the bottom, all wires must be tied at that point. This impedes wiring that must connect to resources above the VXIbus mainframe.

5. The fifth step is to refine the system layout and interconnect list. In particular, add detail on the layout of connector blocks in the ICA, including pinout definitions for each block. It is usually a good idea to retain a generic functional grouping. In other words, dedicate some blocks to sources, some to sensors, some to power, some to high frequency, some to switching networks, and some for growth. This functional grouping is particularly important for systems that will be testing many different DUTs. Remember that many of the custom connections can be made inside the ITA.

6. Finally, it is time to compile a list of all parts and products. For the system side, this should include the ICA, interface terminal modules (for the E3720A), ICA connector blocks and blanks, cable assemblies, and contacts. For the DUT side, this should include ITAs, ITA connector blocks and blanks, cable assemblies, contacts, and all parts necessary to customize the ITA for a particular DUT.

Conclusion

The integration of mass interconnect into a test system adds significant, but often essential, complexity. The system integrator is required to consider many factors. These factors include DUT quantity and type, signal characteristics and density, system components, available rack space, diagnostics, reconfiguration, and so on. One of the most important factors is the availability of appropriate mass interconnect products. The HP 75000 system resource interface provides an excellent set of tools for use with VXIbus-based test systems. Appropriate use of these products can greatly influence system cost, development time, and performance.

Acknowledgments

I would like to thank the following people for their key roles. At HP's Advanced Manufacturing Systems Operation, Mike Stefanisko for his technical expertise and advice, Dan Meitus and Jim McGillivary for their support. At MacPanel Company, Troy Rector and Kellie Coble for their responsiveness. At HP's Loveland Instrument Division, Charlie Schmidt for the design of the ITM, and in particular, Bryan Thompson for his involvement in the definition and in the details of introducing these new products.

A Manufacturing-Oriented Digital Stimulus/Response Test Instrument

This digital functional tester consists of pattern I/O, timing, and command modules configured in a VXIbus mainframe. The maximum pattern rate is 20 MHz and pin-to-pin skew is less than 6 ns.

by David P. Kjosness

Test engineers and system integrators have used analog instrumentation for years in their automated functional test systems. Until now, however, there has been little in the way of cost-effective, manufacturing-oriented digital instrumentation with which to round out their toolkits. The HP 75000 Model D20 was created to fill this void.

The decade of the 1980s saw an explosive increase in the amount of digital circuitry used in all manner of electronic assemblies. Today, an estimated 80% of all circuit boards are manufactured with substantial digital content. In the 1990s, the use of digital circuitry will continue to increase, as will the use of advanced packaging such as fine-pitch surface mount technology (SMT) and multichip modules (MCMs). Also, the proliferation of open architectures will require testing to rigid specifications. The effect of these developments will be a greater emphasis on digital functional testing in future manufacturing schemes.

Functional testing involves the transfer of information to and from the device under test (DUT). A stimulus is sent to the DUT to evoke some response, which is then analyzed to determine if the DUT is operating correctly. If the stimulus and response are digital, then it is a digital functional test. If some part of the stimulus or response is not digital (i.e., analog or mechanical), then it can be said to be a mixed-signal test. In functional testing, the stimulus application and response measurement are generally performed through the assembly's edge connectors; connections to internal nodes are kept to a minimum.

The uses for functional testing can vary considerably. Some manufacturers use it simply to verify connections from connector pins to internal components. Others use it to exercise the functions of their assemblies in go/no-go tests. Still others develop complex fault isolation and diagnostic tests to pinpoint failed components within their assemblies. Regardless of test complexity, there is a need for digital instrumentation that can be integrated with analog instruments like voltmeters and counters.

A General Model for Digital Interfaces

Digital interfaces come in myriad forms. There are standard backplanes (like VXIbus) and proprietary backplanes specific to certain products. There are also non-backplane interfaces, both standard (like SCSI) and proprietary. However different, all digital interfaces share certain characteristics which can be exploited when designing a digital stimulus/response instrument.

The signals of the interface fall into two major categories. First, all but a few of the lines are dedicated to carrying information to or from the DUT. These include the data and address buses on a computer backplane, for example, and will be called *pattern lines* here. Second, there are a few signals, the *control and handshake lines*, which facilitate the information transfers on the pattern lines. Control signals are created by the tester to regulate the transfer process; strobes and clocks are typical examples. In many interfaces, the DUT responds to control signals with a handshake signal to acknowledge a data transfer or to modify the rate of transfer.

Let's define some terms. A *cycle* is the sequence of events necessary to transfer one bit to or from the DUT on each of the interface's pattern lines. A particular interface specification may define more than one type of cycle—read and write, for instance. The specification of a cycle's pattern data along with that cycle's type forms a *vector*. An ordered list of vectors is a called a *sequence*. A complete test consists of the execution of one or more sequences.

As an example, consider a data transfer cycle on the VME (or VXI) bus. Suppose that the tester is acting in the bus master role, writing data to the DUT (Fig. 1a). At the beginning of the cycle, the tester places valid information on the address (A01-A31), address modifier (AM0-AM5), long word(LWORD*) and interrupt acknowledge (IACK*) lines. After allowing for propagation delay and settling time, the address strobe (AS*) is asserted. Also, near the beginning of the cycle, WRITE* is asserted and the data to be written is put on the data bus (D00-D31). This is followed, again after a suitable delay, by the assertion of one or both of the data strobes (DS0*, DS1*). It is now up to the DUT to acknowledge the data transfer by asserting DTACK*, after which the tester will negate the address and data strobes, then wait for the DUT to negate DTACK*. When DTACK* goes false, the cycle is complete and another cycle can begin.

A read cycle (Fig. 1b) is similar, except that the tester negates WRITE* and tristates (doesn't drive) the data bus. Instead, when the data strobes are asserted, the DUT puts its response data on the data bus and asserts

DTACK*. The tester then accepts the response and negates the data strobes, causing the DUT to tristate the data bus and negate DTACK*, completing the read cycle.

In this example, the address, address modifier, and data buses are identified as pattern lines. The address and data strobes are control signals and DTACK* is a handshake line. LWORD*, IACK*, and WRITE* could be treated as pattern lines but are really control-like in nature since they serve to define the type of cycle.

Some observations about the two categories of signals are in order here, beginning with pattern lines. First, pattern lines each assume a single value in each cycle; they make at most one transition per cycle. Second, pattern lines are most often collected into sets that have common functions and timing. These sets are called *buses* and usually contain multiples of eight pattern lines. There might be a 16-bit data bus or a 24-bit address bus, for example. A third observation is that some, but usually not all, pattern lines may be bidirectional. That is, they may carry information first one way and then the other during the course of the test. Data buses, for example, are typically bidirectional but address buses are generally not.

The few control and handshake lines in an interface differ from the pattern lines in a number of ways. First, they are single lines, not contained within buses. (Sometimes an interface will have what is called a control bus, which is really a collection of individual control and handshake signals, not the same as a pattern bus). Another difference is that these lines usually make more than one transition in each cycle. In the VMEbus example above, the data and address strobes, as well as DTACK*, are first asserted, then negated, thereby making two transitions in each cycle. There can be many repetitions of a clock signal in a cycle. A third difference is that control and handshake lines are unidirectional, not bidirectional. Finally, for a given cycle type, the control and handshake lines have the same behavior regardless of the information on the pattern lines.

Handshaking can be either synchronous or asynchronous. If there is no explicit clock signal in the interface, it is an asynchronous handshake. The VME and VXI buses employ this type. On the other hand, if there is a clock signal in the interface, then handshake and control transitions must usually be synchronized to the clock transitions, creating the synchronous variety. An example of this is when a device requests a wait state in a personal computer. When there is a clock signal in the interface, it can be generated either by the tester or by the DUT. In the latter case, the tester must be able to synchronize itself with the DUT-supplied clock, and this forms a type of handshake as well.

Model D20 Requirements

To be useful, a digital stimulus/response instrument must be capable of emulating the DUT's interfaces. Its application to manufacturing testing imposes other requirements, including high test quality and throughput, ease and flexibility of integration and use, and low downtime.

Achieving high test quality places several demands on the instrument's timing capabilities. The tester must first of all be capable of exercising the DUT at full speed. Also, timing relationships at the DUT must closely resemble those defined by the interface specification. This means that the tester must have good resolution and accuracy for edge placement and cycle duration. Since different types of cycles may have different durations or different timing relationships between the various signals, the tester must be capable of changing timing on the fly. Finally, since there may have to be a substantial length of cable between the VXIbus mainframe and the DUT, there must be some means of compensating for the resultant cable propagation delays.

Good signal fidelity is another requirement for high test quality. A tester must deliver clean stimulus waveforms to the DUT and not be difficult for the DUT to drive cleanly.

Maximizing test throughput requires not only that devices be tested at full speed, but that time spent transferring information between the test controller and the tester be minimized. The amount of data that must be transferred must be small, and the transfer rate must be high.

Fixturing is an important aspect of production testing, so a variety of methods of connection to the DUT must be accommodated. In some cases it is possible to put the

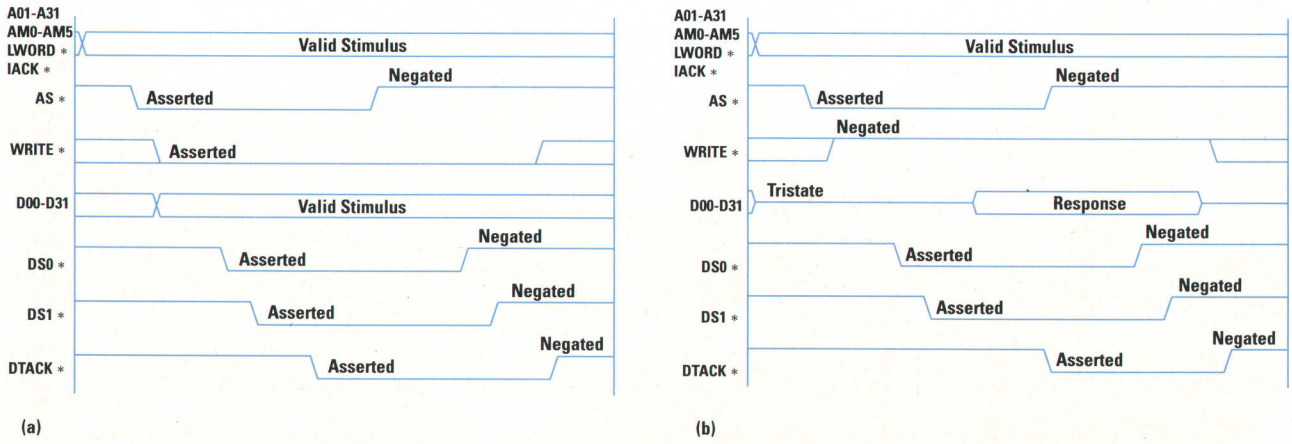


Fig. 1. (a) VMEbus write cycle timing diagram. (b) VMEbus read cycle timing diagram.

DUT close to the instrument, but in others there may be one or two meters of cable separating them. There may also be a mass interconnect device involved, such as the HP 9420A or E3722A interface connector assemblies.

When doing mixed-signal testing, the digital instrument must be synchronized with analog instrumentation. Flexible trigger input/output capability is therefore needed.

Functional test systems use a variety of instrument controller platforms: personal computers, UNIX* workstations, and HP BASIC workstations, to name a few. To be compatible with all controllers, a digital instrument should have an ASCII command language like the other instruments in the system. SCPI (Standard Commands for Programmable Instruments, a superset of IEEE 488.2) command format is preferred. There should also be efficient non-ASCII options for transferring large volumes of pattern data.

Finally, reducing downtime requires a reliable instrument with built-in self-test to identify failed modules quickly. Replacing a module must not require time-consuming recalibration to meet specifications.

Model D20 Architecture

With an eye to the above requirements, the HP 75000 Model D20 design team investigated various digital stimulus/response instrument architectures. We looked at a number of existing products as well as several new proposals. What follows is a discussion of the architecture that emerged from our investigation and the decisions that shaped it.

The fact that interface signals fall into two categories (pattern and control/handshake) suggested a similar division of roles in the Model D20. We therefore created pattern I/O modules (HP E1451A and E1452A) for patterns and a timing module (HP E1450A) for control signals and handshaking.

The timing module generates eight control signals and performs both synchronous and asynchronous handshaking with the DUT. It also accepts and generates triggers for synchronization with other instrumentation, and it provides timing information to the pattern I/O modules in the form of twelve pattern clocks on the VXIbus local bus.

Each pattern I/O module has thirty-two channels, arranged as four eight-bit *ports*. Ports are independent of one another and can be programmed for stimulus output or response input. The two types of pattern I/O modules are identical except that the HP E1451A passes pattern clocks on to the next mainframe slot and the HP E1452A terminates them.

A Model D20 system is configured as shown in Fig. 2. The HP E1450A timing module is placed farthest to the left, followed by a number of HP E1451A pattern I/O modules. One HP E1452A terminating pattern I/O module completes the instrument.

* UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

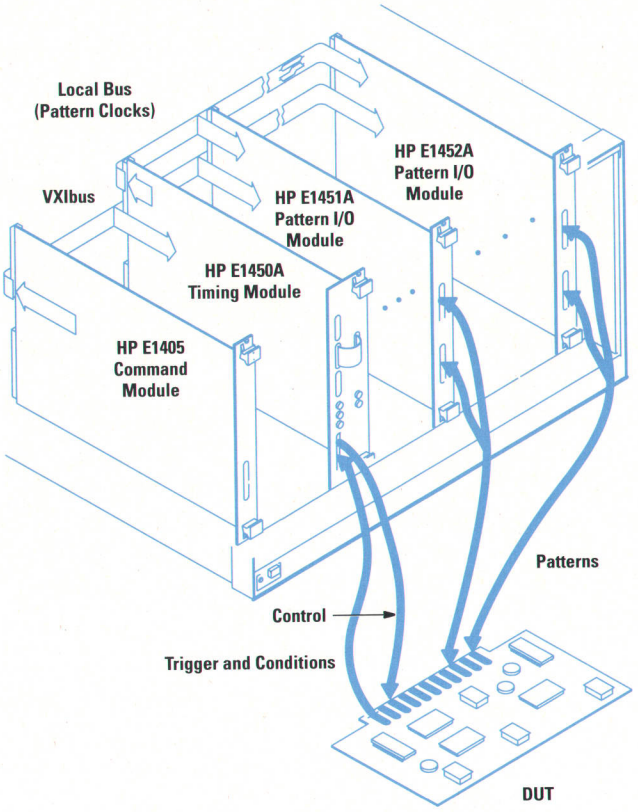


Fig. 2. HP 75000 Model D20 configuration.

A single timing module can service up to ten pattern I/O modules, a full mainframe. Furthermore, up to three timing modules can be linked in a master/slave arrangement, allowing up to thirty pattern I/O modules to be included in the instrument. The timing module is optional when an external clock source is available. As a result, instruments with from 32 to 960 pattern lines and from 0 to 24 control lines can be assembled.

The pattern I/O and timing modules can interface directly to the DUT, but in case the DUT must be some distance from the mainframe, *pods* are provided. A pod contains active circuitry at the end of a two-meter cable. It buffers and reconstructs stimulus waveforms, eliminating any distortions a length of cable might cause, and gives a lower output impedance than a cable would. It also buffers responses from the DUT, providing high-impedance inputs which are easier to drive cleanly than long cables. The HP E1454A pattern I/O pod buffers sixteen bits of pattern data; two pods are used with each pattern I/O module. The HP E1453A timing pod buffers the control and handshake signals associated with the timing module. The HP E1456A and E1455A pods are electrically identical to the HP E1454A and E1453A, respectively, but are mechanically different to allow them to be mounted in the HP 9420A and E3722A interface connector assemblies.

Given this basic architecture for the Model D20, attention was turned to the question of message-based versus register-based programming interfaces. Message-based interfaces simplify instrument programming and provide for controller independence. Since they employ onboard

processors, they allow built-in self-test for quick identification of failures. Sophisticated message-based interfaces can, however, slow data transfer rates, limiting throughput. They require significant amounts of board space, and in the case of the Model D20, where there is not one module guaranteed to be a part of the instrument, each module would be required to have one. This would raise the cost significantly.

Register-based interfaces, on the other hand, provide direct control of the hardware and maximum data transfer rates. They are less expensive and take less board space. Programming register-based modules as complex as the Model D20's can be a daunting task, however.

Fortunately, recent versions of the HP E1405 command module, with their downloadable firmware capability, provide a single place from which to control multiple register-based modules with an SCPI command language. This approach was taken with the Model D20, realizing the benefits of a message-based interface while still allowing fast, unrestricted register-based operation.

Maximum pattern rate is a key specification for a digital instrument, and many standard and proprietary interfaces were examined to determine how fast the Model D20 should run. The VME and VXI buses have a maximum pattern rate of 10 MHz. Personal computers, while having clock rates of 25 MHz or more, have pattern rates of 12.5 MHz or less. Most other interfaces have speeds of the same order. It became apparent, then, that an instrument capable of 20-MHz pattern rates would be viable, provided it could produce control signals at up to 40 MHz.

Picking 20 MHz as a maximum pattern rate meant that much of the Model D20's circuitry could be implemented with off-the-shelf CMOS technology. The low power of CMOS places low demands on the mainframe power supply and cooling system and minimizes failure rates. Using commonly available technology minimized both the time to market and the cost.

Since the vast majority of digital devices have either TTL or CMOS interface levels, the Model D20 needs to be compatible with both. To this end, all outputs swing to CMOS levels (ground to +5 volts), allowing the instrument to drive either TTL or CMOS DUT inputs. Conversely, all inputs have 1.5-volt typical thresholds, allowing them to be driven by either TTL or CMOS DUT outputs.

Pattern I/O Modules

Realizing that most of the pattern signals in a DUT interface can be collected into buses that have multiples of eight lines led to the concept of the pattern I/O port. A port is a self-contained, independent, eight-bit I/O structure with a single clock to control its timing. Each pattern I/O module contains four ports. Compared with so-called "per-pin" architectures, this "per-byte" approach reduces cost considerably without seriously limiting flexibility in real applications.

As shown in Fig. 3, each port consists of an address counter, a pattern and control memory, input and output stages, clock selection and delay circuitry, and a comparator. At the beginning of a sequence, the address counter is preset. It then increments for each cycle (vector) of that sequence. The memory has 65,536 (64K) twelve-bit words. Eight bits of each word are used for pattern data while the remaining four provide cycle-by-cycle control for the port. The output and input stages provide the interface to the DUT (if there is no pod) or to the pod cable. The clock for each port can be one of the pattern clock signals from the timing module or an external clock signal. Since there is only one pattern value associated with each vector, a single clock edge per cycle provides all the timing information a port needs. Pattern clocks are subjected to a variable delay for the purposes of deskew and cable-length compensation, to be described below.

A port can be programmed to perform one of three different tasks, with its memory assuming a different role for each. The first mode of operation is stimulus, in which the eight pattern bits of each memory word are sent through the output stage to the DUT. In this mode, one of the control bits provides cycle-by-cycle tristate

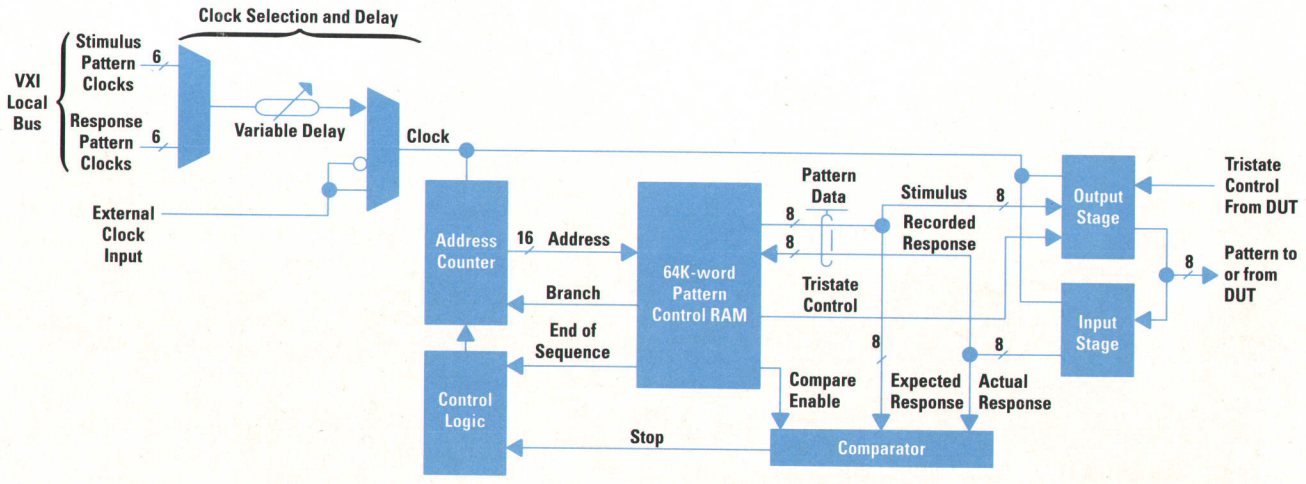


Fig. 3. Pattern I/O port block diagram.

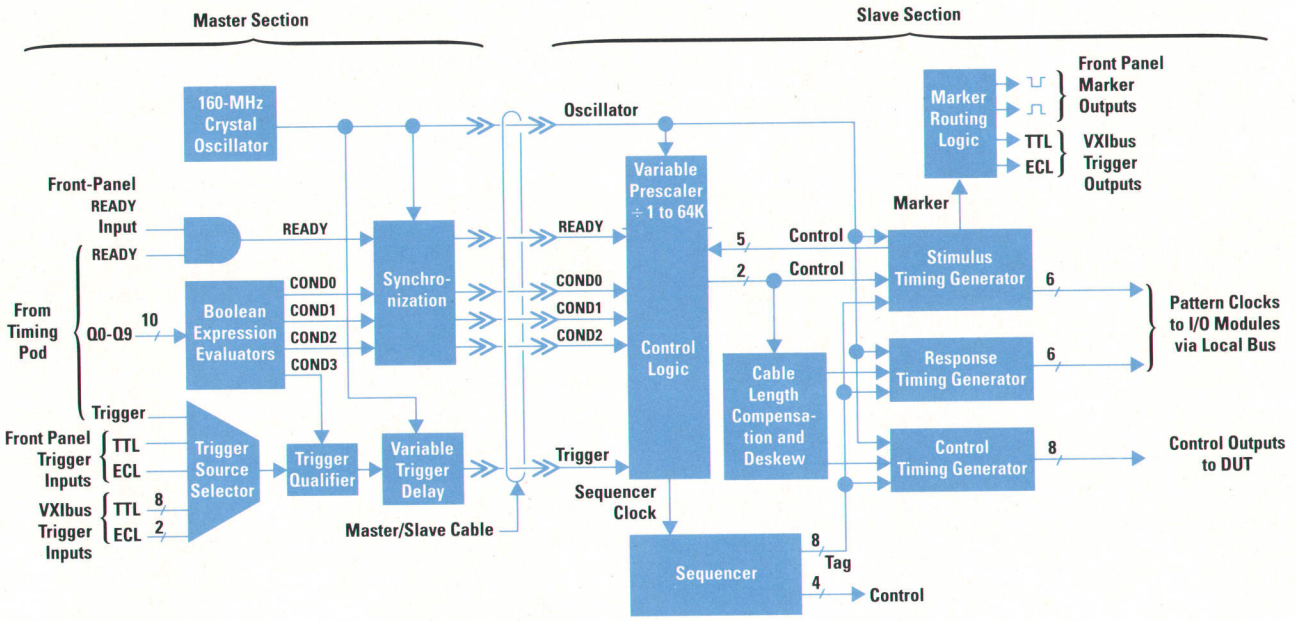


Fig. 4. Timing module block diagram.

control for the whole port; when tristated, outputs assume a high-impedance state—neither high nor low. An additional input to the output stage allows external tristate control.

The second mode is record, in which eight bits of response pattern data are sampled by the input stage and written into the memory in each cycle. Finally, there is a compare mode, which brings the comparator section into play. In this mode, eight bits of expected response data from the memory are compared with the eight bits of actual response from the DUT. Another control bit enables this comparison on a cycle-by-cycle basis, and there is a static mask register that allows any bits to be declared "don't care". When a comparison fails, the port stops, thus preserving the memory address of the first failure and both the expected and the actual responses.

One of the remaining two control bits is used to indicate the end of a sequence, causing the port to stop. The last is a branch bit used to implement a diagnostic looping feature by forcing the address counter to load a nonsequential address from a branch destination register (not shown in Fig. 3).

There is a high degree of flexibility in this port concept. Multiple ports with duplicate programming and the same pattern clock can form pin groups corresponding to the DUT's buses. IEEE 488.2, the basis for SCPI, restricts integer data to thirty-two bits or less, thereby limiting firmware support for pin groups to no more than four ports. Larger buses can be emulated by duplicating the programming for more than one pin group. Ports can be programmatically reassigned to different pin groups as necessary to test different devices.

Bidirectional buses can be tested by connecting a stimulus pin group in parallel with a response pin group. An advantage of this approach is that the two pin groups are otherwise independent, allowing complete flexibility in timing and making cable-delay compensation possible. Another advantage is that the resources to implement bidirectional buses need only be purchased for those pattern lines that are bidirectional, and not necessarily for all.

Paralleling pin groups creates other possibilities, as well. Responses can be simultaneously recorded and compared, for example. Also, multiple stimulus or response pin groups can be combined, to double, triple, or even quadruple the Model D20's pattern rate and memory depth.

Compare mode and deep memory increase test throughput by reducing the amount of information that must flow between the instrument and the controller for each device tested. Compare mode does this by eliminating the need to move what could be millions of bits of response data into the controller for go/no-go analysis after each sequence. It is only necessary to interrogate two registers in each response port. Of those ports indicating a comparison error, the one with the lowest memory address identifies the first failure. Deep memory allows a long sequence or multiple shorter ones to reside in the Model D20. This reduces the incidence of reloads during testing.

Timing Module

The HP E1450A timing module accepts a number of inputs from the DUT and/or other instrumentation and produces control outputs to the DUT and pattern clocks to the pattern I/O modules. It also generates triggers for other instruments. Its overall block diagram is shown in Fig. 4.

The timing module's inputs include triggers from various sources, two READY lines for synchronous handshaking, and ten lines (Q0-Q9) that are used mainly for asynchronous handshaking and trigger qualification.

All of these inputs must be synchronized with the internal 160-MHz oscillator, and all the timing modules in the

instrument must "see" the inputs at the same time. For this reason, the timing module is divided into master and slave sections, which are linked by an external cable. A timing module becomes a slave by having its slave section connected to the master section of another timing module.

Master Section

As Fig. 4 shows, the master section accepts inputs, processes them, and synchronizes them with the internal oscillator. The two READY lines, one from a front-panel connector and one from the pod, are ANDed together, then sent to the slaves after being synchronized. Lines Q0-Q9 become variables in four user-defined Boolean expressions. These lines address a fast memory that has been filled with the truth tables of the expressions (labeled "Boolean Expression Evaluators" in Fig. 4), and the results appear on four condition lines. Three of the conditions (COND0, COND1, and COND2) are for general handshaking and synchronization, and are sent to the slaves, while the fourth (COND3) becomes a trigger qualifier. Available trigger sources include the eight TTL and two ECL VXIbus trigger buses on the backplane, an input on the timing pod, and both TTL-level and ECL-level front-panel connectors. Positive or negative slopes can be selected for the pod and front-panel trigger inputs but slopes for the VXIbus trigger buses are fixed by the VXIbus standard. Trigger qualification is done by sampling the state of COND3 when the selected edge from the selected source occurs. If COND3 is true, then the trigger event is recognized, but it is not passed on to the slaves until after a programmable delay. This trigger delay is provided by a sixteen-bit counter, giving a range of zero to $(2^{16} - 1) \times 6.25$ ns (almost 409.6 μs). READY, COND0, COND1, COND2, the qualified and delayed trigger, and the oscillator signal appear at three front-panel connectors, from which they drive the slave sections via master/slave cables.

Slave Section

While the master section deals with inputs to the timing module, the slave section is concerned with the module's outputs, namely control signals, pattern clocks, and triggers. Recall that control signals are generally higher-speed than patterns, with possibly many transitions per cycle. They are essentially arbitrary digital waveforms. Pattern clocks (which provide timing information to the pattern I/O modules via the VXIbus local bus) and trigger pulses can be thought of and created in the same way. Therefore, the timing module has three similar timing generators: one to create marker (trigger output) pulses and six stimulus pattern clocks, one for six response pattern clocks, and one for the eight control signals. To accomplish system deskew and pod/cable delay compensation (discussed later), the three timing generators need to operate in different time frames and therefore must be separate. These timing generators, along with a prescaler, a sequencer, several control state machines, and some cable-length compensation and deskew circuits, make up the timing module's slave section.

Fig. 5 shows the slave section in more detail. The stimulus, response, and control timing generators are all similar, consisting of a ten-bit address counter, 1024 words of memory, an output register, and a little random logic. At the beginning of each cycle, the address counter loads a new value, then begins incrementing through successive memory addresses. The memory data is clocked through the output register, creating waveforms. Cycles are defined by putting appropriate data into a range of memory locations, each corresponding to what is termed a *subcycle*. Many cycles can be defined at the same time, as long as the sum of all of the subcycle locations does not exceed the size of the memory.

Oscillator frequency determines the timing resolution of the instrument, and should therefore be as high as possible. However, memory access time and logic delays limit the maximum oscillator frequency. Sticking to the policy of using relatively inexpensive, readily available parts, a 160-MHz oscillator is used, for a best-case resolution of 6.25 ns. This number is an even divisor of 50 ns (the pattern I/O ports' minimum cycle time) and provides adequate resolution in most cases.

The control timing generator's memory is eight bits wide, each bit corresponding to a control signal output. Similarly, the response timing generator's memory is six bits wide, one per response pattern clock signal. The stimulus timing generator has twelve-bit words, however. Six bits are for stimulus pattern clocks and the rest are for the ECL marker pulse, for testing the three conditions from the master section, and for both conditional and unconditional end-of-cycle definition.

Two synchronous bits control each timing generator. The four resulting operations are next subcycle, next cycle, hold, and inhibit. The next subcycle operation causes the address counter to increment and the new memory data to be clocked out on the next oscillator cycle. The next cycle operation is similar, except that instead of incrementing, the address counter loads the first address of another cycle. The hold operation freezes the address counter but allows the output register to update its contents. Finally, the inhibit operation causes the oscillator to be completely ignored; nothing changes state.

The prescaler is a programmable sixteen-bit, divide-by-N counter which causes the timing generators to hold for $N - 1$ out of every N oscillator cycles. This lengthens each subcycle and changes the Model D20's timing resolution to $N \times 6.25$ ns. The maximum subcycle duration is $2^{16} \times 6.25$ ns (409.6 μs).

At the end of each cycle, a next cycle operation occurs. The timing generator address counters are loaded with an address determined by the sequencer. This sequencer makes the Model D20's on-the-fly timing changes possible by either causing the same cycle to be repeated or switching to a different cycle. The sequencer functions like a pattern I/O port configured for stimulus operation, except that its memory contains eight-bit tags instead of pattern data, and its four control bits have different functions. The sequencer is clocked once, furnishing one tag per cycle. Starting addresses in the timing generator memories are formed by multiplying the tags by four (left-shifting them two places). Three of the four remaining bits permit cycle-by-cycle control of trigger arming,

marker (trigger output) pulse generation, and breakpoints. The last bit designates the end of the sequence.

State Machines

As can be seen in Fig. 5, the control logic section provides control of the timing generators through the actions of several state machines. The outputs of the run/stop, trigger, condition, and EOC/SEQCLK state machines are combined with the prescaler's output (PCLK) into the two timing generator control bits. The EOC/SEQCLK state machine monitors the near-end-of-cycle (NEOC) bit from the stimulus timing generator memory to determine when the

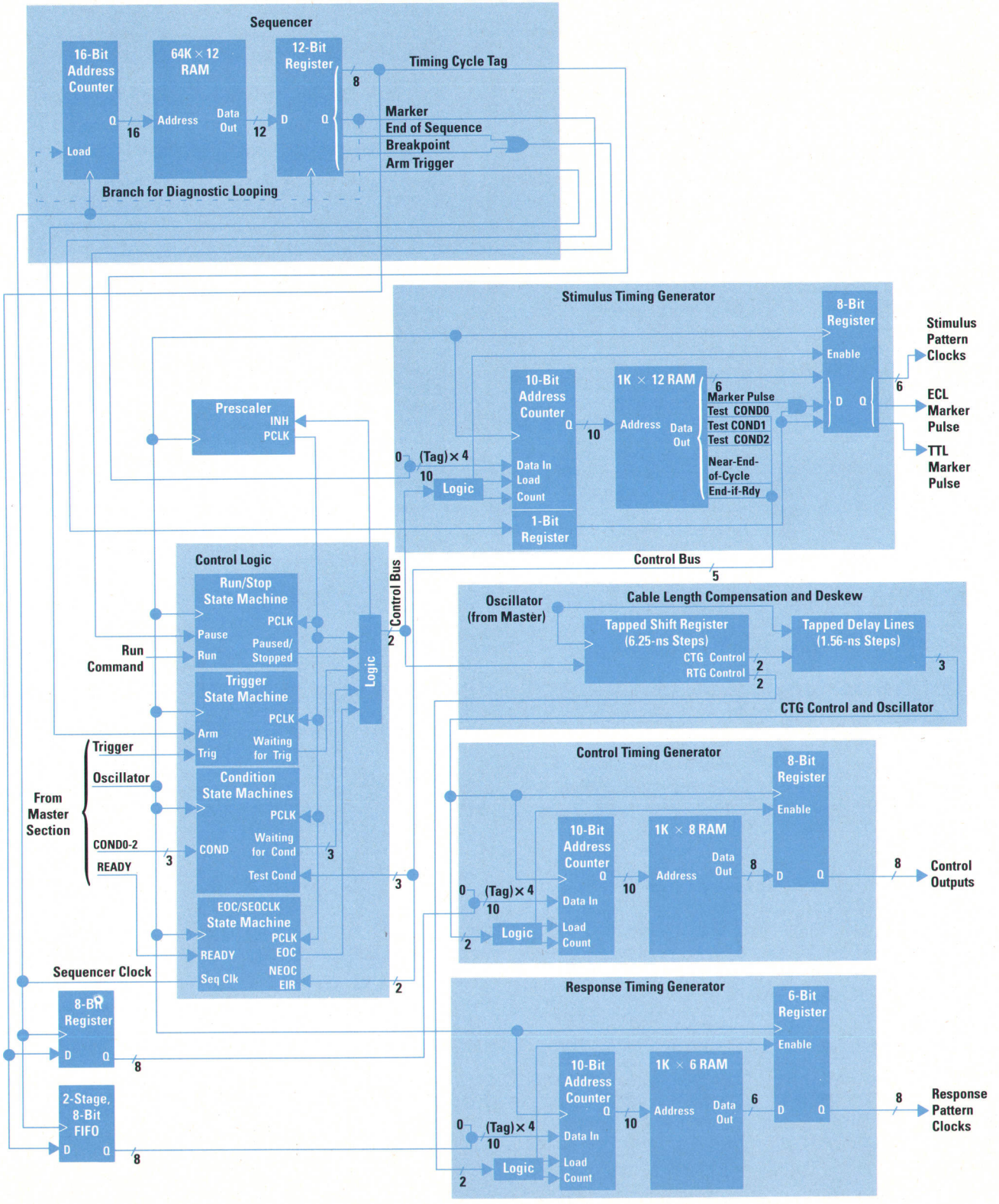


Fig. 5. Timing module slave section detail.

end of the cycle is at hand. It causes another cycle to begin by initiating a next cycle operation in the last oscillator period of the current cycle. It also issues a clock pulse to the sequencer to make the next tag and control bits available.

An end-if-ready (EIR) bit in the stimulus timing generator memory works like NEOC, except that it is ANDed with the READY line from the master section. If the DUT is indicating that it is ready for the next cycle to begin when EIR is encountered, the cycle is terminated in a manner similar to when NEOC occurs.

An example will show how this end-if-ready feature forms the basis for synchronous handshaking. Suppose that the DUT is a device that requires a continuous clock input (to be provided by a Model D20 control output) and that it can request wait states, which must lengthen a cycle by whole clock periods, up to some maximum number. This is typical of a personal computer card. The Model D20 cycles would then be programmed to be the maximum length allowed, but an EIR would be placed at the point where a cycle normally ends and at whole DUT clock periods thereafter. If the DUT is not requesting a wait state, it drives READY true, and the cycle terminates at the first EIR. If, on the other hand, the DUT needs a wait state, it makes READY false, causing the cycle to go past the EIR. The cycle will then terminate at a subsequent EIR, once READY goes true, or at the end of the cycle as defined by NEOC. Placing EIRs at DUT clock period intervals ensures that the tester will always generate DUT clocks with the proper period.

For each of the condition lines from the master section there are a corresponding state machine and control bit from the stimulus timing generator memory. When a control bit is true, the associated state machine tests its condition at the end of that subcycle. If the condition is false, the timing generators and prescaler are inhibited, and all activity stops. Once all the conditions being tested become true, operation resumes.

Referring back to the VMEbus example above (Fig. 1), recall that the instrument must first wait for the assertion, then negation, of DTACK* by the DUT. For this example, two conditions could be programmed to correspond to the assertion and negation of DTACK*, respectively. The Model D20 would then be programmed to test these conditions at appropriate points in its cycles.

When the sequencer arms the trigger in a particular cycle, the trigger state machine inhibits the timing generators and prescaler immediately before the beginning of that cycle. Operation continues once a trigger is received from the master section. Trigger events that arrive while the trigger is not armed are ignored.

The Model D20 can be synchronized with an external clock signal by feeding the clock into a trigger input and then arming the trigger immediately before the expected clock edges. The variable trigger delay (in the master section, Fig. 4) provides the means to change the phase of the instrument relative to the external clock. This is a simple way to align stimulus patterns and control signals with other events in the DUT. However, trigger inputs are synchronized with the internal 160-MHz oscillator, resulting in 6.25 ns of random error, or jitter.

The run/stop state machine can be idle, paused, or running. If it is idle (its initial state) or paused, the timing generators and prescaler are inhibited. The running state is entered, allowing operation to begin, when a run command is received. A pause flag, which can be set or cleared at any time, is tested at the end of each cycle, with the paused state resulting if it is true. A run command with the pause flag set causes a single cycle to be executed, while a run command with the flag cleared either starts or continues the sequence. At the end of a sequence, firmware places the instrument back into the idle state.

The breakpoint and end-of-sequence control bits from the sequencer are ORed together and then fed to the run/stop state machine. The resulting sequencer pause line has the same effect as the pause flag. Thus, the instrument pauses at each breakpoint and at the end of the sequence. These two bits are separate so that breakpoints can be distinguished (by reading a status register) from the ends of sequences.

The fourth control bit from the sequencer causes the stimulus timing generator to produce two types of marker pulses. One type is always two subcycles wide and can be directed to one or both of the VXIbus ECL trigger buses. The second type of marker pulse lasts the whole cycle and can be sent to any or all of the VXIbus TTL trigger buses. This second pulse also appears in both active-high and active-low forms on the front panel.

System Deskew and Cable-Length Compensation

Suppose that two outputs from a digital instrument are to have transitions that happen at exactly the same time. This is impossible to achieve in reality, and the difference in time between the transitions is called skew. A similar notion applies to inputs that are supposed to sample DUT responses at the same time, but don't. Skew is the main component of tester timing error and should therefore be minimized at the DUT.

There are many sources of skew in an instrument like the Model D20. A big contributor is the statistical variation in delay between instances of otherwise identical timing or data paths. Two pattern I/O ports can have substantially different delays from their pattern clock inputs to their stimulus outputs. Pods will also differ from unit to unit.

Even two modules with equal delay will have skew between their outputs because of backplane propagation time. The farther a pattern I/O module is from the timing module, the later it will receive pattern clocks. Each slot introduces a delay of about 800 ps.

Now suppose that there are pods (and their cables) between a Model D20's pattern I/O ports and a DUT (see Fig. 6). Suppose also that a stimulus transition is meant to occur at the same instant ($t = 0$) as a response transition at the DUT. For this to occur, the instrument must generate the stimulus before $t = 0$. Likewise, the response arrives at the response port after $t = 0$. To create the proper timing relationships at the DUT, response pattern

clocks must lag stimulus pattern clocks by approximately twice the delay of a pod and its cable.

Minimizing skew and compensating for pod/cable delay both require that internal timing relationships be adjusted according to the actual delays of various parts of the system. When a Model D20 module or pod is manufactured, its delays are measured and stored in electrically erasable read-only memory (EEROM) in that module or pod. Every time the instrument is turned on, its EEROMs are read by the firmware, which then uses the information to compute how much delay to add using variable delays in the pattern I/O and timing modules. This compensates for the major contributors to skew, as well as the pod/cable delays, without any special measurements, fixturing, or programming on the part of the user. Even after a failed module or pod is replaced, the instrument automatically meets its skew specifications.

In many cases there will be a significant length of cable between the instrument (with or without pods) and the DUT. For these situations, there is a command that allows users to specify the delay to the DUT and back. The Model D20 then corrects for this round-trip time as well as its own, in the manner described above. Since many cables have specified propagation velocity, timing accuracy can be maintained at the DUT, even in the absence of pods. This feature also makes it feasible to build custom interface circuitry and compensate for its delay. For example, translators to and from ECL levels or differential drivers and receivers for the SCSI bus could be employed.

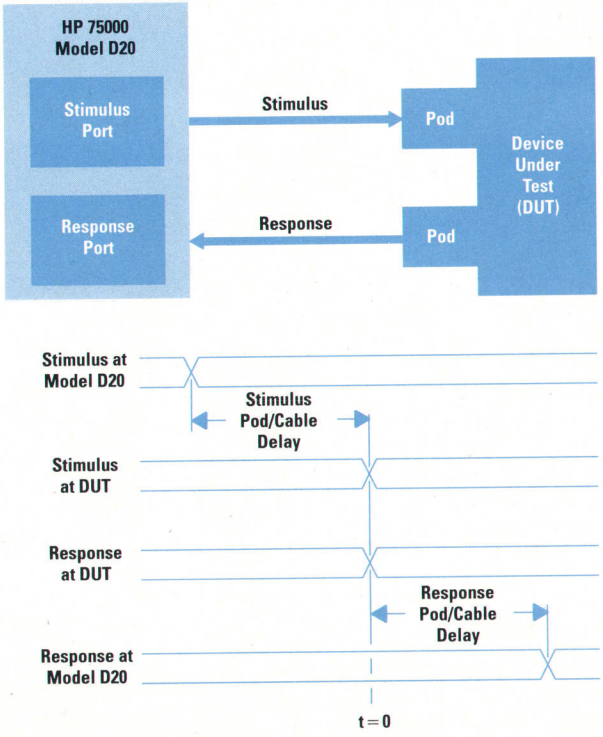


Fig. 6. Pod and cable delay compensation.

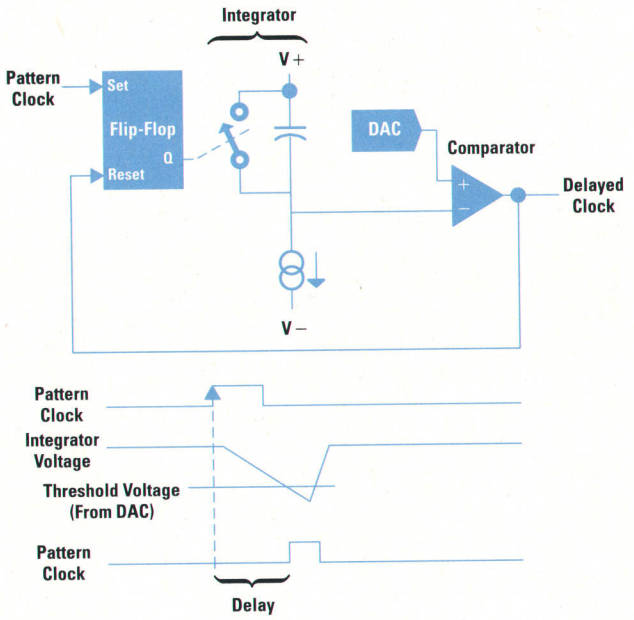


Fig. 7. Variable clock delay.

The maximum possible delay from a pattern clock at the output of the timing module to a stimulus output (including backplane, pattern I/O port, and pod delays) is a known constant. Since actual delays are always lower, the Model D20 can be deskewed by adding an appropriate delay in each port's clock path. The circuit that does this is shown in Fig. 7. When the pattern clock arrives, it sets the flip-flop, which opens the switch and allows voltage to start building on the integrator capacitor. When this voltage crosses a threshold set by the digital-to-analog converter (DAC), the comparator trips. This generates the delayed clock signal and resets the flip-flop, closing the switch and rapidly resetting the integrator in preparation for the next pattern clock. Thresholds that are more negative take longer to reach and therefore result in longer delays. The total range of delay variation is 25 ns and the worst-case resolution is 130 ps.

To compensate for pod and cable delays, response pattern clocks must be delayed with respect to stimulus pattern clocks by approximately 50 ns. The cable length compensation circuitry in the timing module (Fig. 5) accomplishes this by delaying the two control bits to the response timing generator with a twelve-stage tapped shift register. This results in up to 75 ns of delay in 6.25-ns steps. The clock delay circuit in each response port interpolates these steps and deskews the port as described in the paragraph above.

The control timing generator is much faster than a worst-case stimulus port located at the far end of the backplane. Delay must therefore be introduced to deskew the control signals. This is done with the same tapped shift register that is used for the response timing generator. The two control bits and the accompanying oscillator signal are then fed through tapped delay lines to subdivide the 6.25-ns shift register steps into quarters. Thus,

control timing generator delay can be adjusted with 1.56-ns resolution.

The most-significant causes of skew are measured or characterized at the factory and compensated in the deskew process. The many factors that cannot be compensated combine to form the Model D20's skew specification. Two main sources of this residual skew are the differences in delay among the eight bits of a port and between rising and falling edges. Other contributors include power supply and temperature variations, aging, inconsistency between backplanes, and available deskew resolution. The net effect is that stimulus transitions (excluding going into or out of tristate), control signal transitions, and actual response sampling times, will all be within ±3 ns of their programmed values. This gives the Model D20 a pin-to-pin skew of 6 ns.

Since all actions are initiated by crystal-controlled circuitry, an additional error of only 0.01% is introduced for events that are separated in time. This is usually insignificant, being on the order of picoseconds in most cases.

Conclusion

The HP 75000 Model D20 was created to fill the need for digital stimulus/response capability in a wide range of manufacturing functional test applications. Characteristics of digital interfaces and the VXIbus platform were exploited in the design, resulting in a high-performance instrument that is both cost-effective and easy to integrate and use.

Acknowledgments

No project of this size is brought to completion without honest efforts from too many people to name here. The author would, however, like to recognize the accomplishments of the rest of the R&D hardware and firmware team. Lee Gregory's architectural insight and solid circuit design put the Model D20 on firm footing from the beginning. Phil Mizuno implemented the pattern I/O modules. Tim Lock designed cost-effective yet EMI-proof packaging for both modules and pods. Rick Adams wrote all the firmware with the exception of the timing correction algorithms, which fell upon Jim Epstein's shoulders. Bob Hetzel and Dave Swigert designed and built the production test and calibration systems, respectively, while Bob Vienot's technical skills continued to amaze us all. Finally, Larry Jones provided the leadership and the glue for the team, making the tough calls when necessary.

Digital Test Development Software for a VXIbus Tester

This software provides ease of use and direct control for the complex hardware of the HP 75000 Model D20 tester. It uses a spreadsheet paradigm and separates the programming of pattern data from that of timing.

by Kenneth A. Ward

The HP E1496A digital test development software is designed to make it quick and easy to use the HP 75000 Model D20 VXIbus digital functional tester (see article, page 59). In one sense, the software acts as a front panel for the instrument; since the Model D20 consists of VXIbus modules, it has no conventional front panel. The HP E1496A digital test development software provides the user with a graphical environment in which digital tests can be developed and debugged. The challenge for software like this is to let users think in terms of the problem they need to solve instead of thinking of how to program the hardware.

Hardware Overview

The HP 75000 Model D20 hardware consists of two types of modules: the HP E1450A timing module and the HP E1451A and E1452A pattern I/O modules. The pattern I/O modules contain the digital stimulus and response data. Each module has 32 channels, divided into four ports of eight bits each. Each port can be programmed for either stimulus or response, and has the ability to do a real-time compare or to record the response data. Each port can store up to 65,535 patterns, and can be clocked at up to 20 MHz. Stimulus data for each port can be an eight-bit value or can be tristate (not driven). Expected response data can be an eight-bit value or don't care. Up to four ports can be grouped together to form pin groups up to 32 bits wide. Each port can be externally paced, or can be programmed to take timing from the timing module via one of twelve timing channels. Although each port can be clocked at most every 50 ns (20-MHz maximum clock rate), the timing module provides 6.25-ns resolution, so two different ports can be clocked just 6.25 ns apart.

The timing module provides the timing for the pattern modules along with high-speed control outputs, condition inputs, and triggers. There are six timing channels for stimulus timing and six timing channels for response timing. The eight high-speed control output lines can be programmed to provide arbitrary digital control waveforms at clock rates up to 40 MHz. Ten condition input lines and a ready input line allow handshaking with the device under test (DUT). The timing module also has the ability to wait for and produce triggers so the test can be coordinated with other instruments.

The timing module is programmed in terms of timing cycles. Each timing cycle specifies the length of time for the cycle, when each timing channel will produce a clock, what the waveform for each control output is, and what to do with the condition and ready inputs. Up to 256 timing cycles can be defined. Each of the 65,535 vectors can refer to any of these timing cycles, potentially using a different timing cycle for each vector and switching between them with no pause in the test. This is called changing timing on the fly.

Massive amounts of data are involved here. For a system with ten pattern I/O modules, there can be over five megabytes of pattern data. There can be 256 different timing cycles, each specifying timing for twelve timing channels and waveforms for eight control output lines. A friendly means of entering, editing, and viewing all of this data is highly desirable. The Model D20 modules have a command set that is compatible with the Standard Commands for Programmable Instruments standard (SCPI), but SCPI is a textual, programmatic interface. What is needed is a visual interface that gives the user a feeling of direct control of the hardware.

Software Design Decisions

Presenting massive amounts of pattern data to the user was the first problem to solve. After surveying several different methods, the spreadsheet paradigm was chosen. It was expected that users would already have some familiarity with spreadsheets, and could apply that knowledge to make the HP E1496A digital test development software more immediately intuitive.

The next important decision was to separate the programming of pattern data from the programming of timing. The hardware is divided in this manner, and there are many benefits that can be realized by separating the two. The most common DUTs envisioned for the HP 75000 Model D20 hardware are digital devices, which usually function in terms of cycles. Anytime a cycle is executed on a bus, the times when the values change on the bus and the states of the bus control lines remain constant every time the cycle is executed. The only things that change are the values on the bus. Separating the timing from the pattern data allows the user to program the

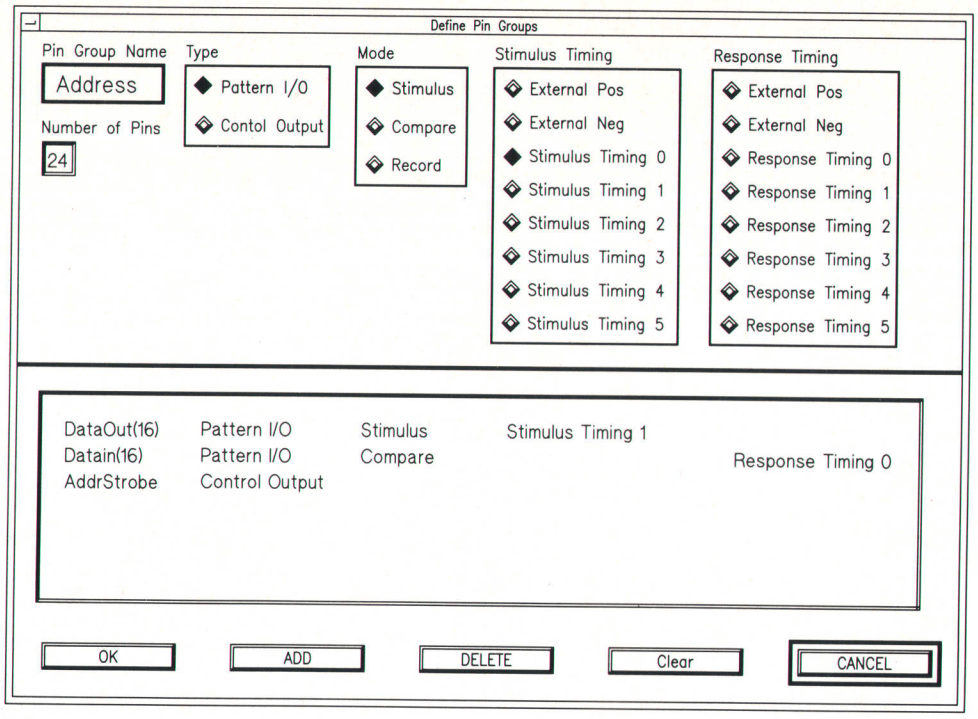


Fig. 1. Define Pin Groups dialog box.

digital test the same way the DUT works, a major advantage.

For the HP E1496A digital test development software, this meant that there would be two spreadsheets: one for pattern data and one for timing cycle definitions. The timing events are specified graphically, since timing information is inherently graphical, like a data sheet timing diagram.

The HP E1496A digital test development software is designed specifically for the HP 75000 Model D20 hardware. It is not meant for general-purpose instrument control or for integrating multiple instruments into a complete test system. The purpose of this software is to create and edit the digital test, run the hardware for the purpose of debugging that test, and produce an SCPI output file that contains all of the commands to program the hardware with the defined test.

Defining Pin Groups

The user wants to program in terms of DUT names, such as address, rather than in terms of the tester, such as slot 4, port 1. The software allows the user to enter names that represent the pin groups on the DUT. Two types of pin groups can be defined: pattern I/O and control output. For DUT pin groups that have large numbers of pins and values that change only once per timing cycle, it is best to use the pattern module ports and the pattern I/O pin group type. For this type of pin group, three other attributes must be specified: the number of pins, the mode, and the timing channel. The number of pins can range from one to thirty-two. The mode determines whether the pin group is to be a stimulus to the DUT, to perform a real-time comparison of the response from the DUT, or to record the response from the DUT. The timing channel specifies which timing-module timing channel the pin group will use for timing, or if it is to be externally paced. When assigning timing channels, if two pin groups specify the same timing channel, they will always have the same timing. If two pin groups specify different timing channels, they can have different timing.

Fig. 1 shows the Define Pin Groups dialog box with the pin group Address being defined. As a result of this definition, the software will assign three ports of a pattern I/O module to be Address, and will program them to be stimulus ports that are clocked from stimulus timing channel 0. The assignment of pin group names to tester ports is automatic, taking the first available ports where there are enough contiguous unassigned ports to satisfy the specified number of pins. This assignment can later be viewed and modified by the user.

For DUT pin groups that are control pins, it is best to use the timing module control outputs and the control output pin group type. Control output type pin groups can only be one pin wide, and the mode and timing channel attributes do not apply, since they are for controlling pattern I/O ports. A pin group assigned to be a control output is automatically assigned to the first available control output line, like the automatic assignment of pattern I/O type pin groups to pattern I/O ports.

Defining Timing

The timing cycle spreadsheet (Fig. 2) allows the user to enter DUT timing information graphically. The spreadsheet has DUT pin group names down the left side, one per row, and has timing subcycle numbers across the top, one subcycle per column. There are 1020 subcycles available. Each timing subcycle has a cell for defining a timing event for each pin group. The timing subcycles are grouped together and named to form the timing cycles. Each subcycle represents a period of time equal to the system timing resolution, which can be set by the user. Each timing cycle represents a period of time equal to

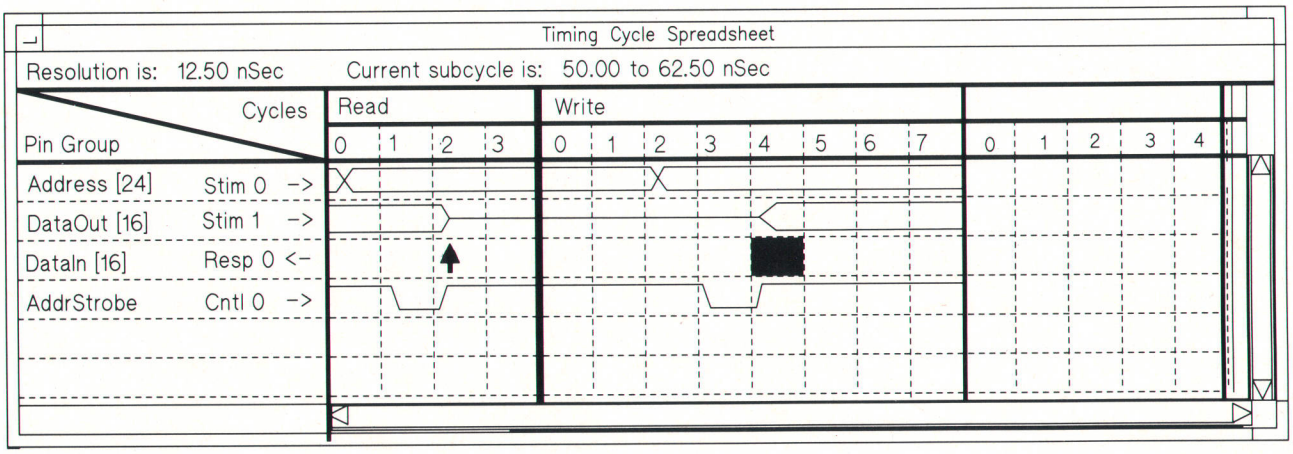


Fig. 2. Timing cycle spreadsheet with timing filled in.

the system timing resolution times the number of subcycles in the timing cycle, and must be at least 50 nanoseconds long (20-MHz maximum clock rate). The system timing resolution must be a multiple of 6.25 ns, the minimum system resolution. The maximum timing resolution is 0.4096 ms (65,536 times 6.25 ns).

Every pattern I/O type pin group must have one and only one timing event per timing cycle. The control output type pin groups can have multiple timing events per timing cycle. Timing information is entered into the timing cycle spreadsheet by selecting a cell, and then selecting the timing event to go into that cell from the Timing Cell Values menu (Fig. 3).

Once the timing cycle waveforms have been entered, the length of the timing cycle must be set. This is done by selecting the subcycle column that is to be the last subcycle in the timing cycle, and then selecting a menu choice. A heavy vertical bar at the end of the subcycle indicates the end of the cycle. Beyond the end of the cycle, the subcycle numbering starts over again from zero, since this represents the number of the subcycle relative to the start of the cycle. The subcycles to the right of the end of the cycle are now available to define another timing cycle.

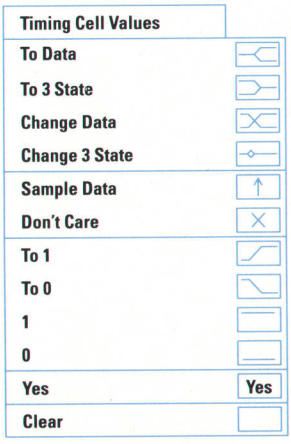


Fig. 3. Timing Cell Values menu.

Finally, each timing cycle must be given a name. These symbolic names are used by vectors in the pattern sequence spreadsheet to reference the timing cycles. Timing cycles are named by selecting the entire timing cycle and then selecting a menu choice, which brings up a dialog box for entering the desired name.

Defining Patterns

The pattern sequence spreadsheet (Fig. 4) allows the user to view and edit the test vectors. The pattern I/O type DUT pin group names are on the left, one per row. Across the top are vector numbers, one per column. There are 65,535 vectors available. Each vector has a cell for the value for each pin group, and in the row labeled Timing Cycle, a cell for specifying the name of the timing cycle to be used for applying that vector. These timing cycle names link the vectors to timing cycles defined in the timing cycle spreadsheet. The timing cycle defines when a pin group's port data is clocked, while the pattern sequence spreadsheet defines what that data is. In the example shown in Figs. 2 and 4, each vector that refers to timing cycle Read will be 50 ns long, and will clock out the Address value immediately. Any vector that refers to timing cycle Write will be 100 ns long, and will clock out the Address value after a delay of 25 ns.

Data is entered into the pattern sequence spreadsheet by selecting a cell and then entering or editing the value for the cell by using menu choices or the keyboard. The operations of entering data and moving from cell to cell are consistent with other spreadsheet programs. If a timing cycle name is specified for the vector and that timing cycle specifies that a pin group should be tristate or don't care, then the corresponding cell on the pattern sequence spreadsheet contains an X to indicate that no data needs to be entered for that cell. Data can also be specified as tristate or don't care by entering an X into the cell. For pin groups of more than one pin, the pin group can be expanded into one row per bit, allowing the values to be viewed and edited as binary digits. The base

in which the pattern sequence spreadsheet displays data is selectable.

When the development system software is connected directly to the HP 75000 Model D20 hardware, more pattern sequence spreadsheet features become available that are very useful when debugging the test. For pin groups that are programmed to record input data, the data will be read back from the tester and displayed on the pattern sequence spreadsheet when the test pauses or stops. If a comparison error is detected by the hardware, the pattern sequence spreadsheet will update and display the vector that failed. For each pin group for which a failure was detected, the actual received data is displayed along with the expected result and the pin group mask value. For each pin group programmed to compare data, the compare data mask value can be set to mask off individual bits from the comparison. Breakpoints can be set on vectors; these cause execution to pause before the vector is executed.

Importing Pattern Data

For users who generate vectors with a program or with a simulator, the development system software can import files in the pattern capture format (PCF). This is an ASCII file format designed by the Manufacturing Test Division of Hewlett-Packard for just this purpose. If the user produces vectors in the PCF format, or translates simulator output to the PCF format, the HP E1496A digital test development software can bring those vectors into the pattern sequence spreadsheet.

Combining Tests

Tests can be combined by merging the currently loaded test with a test stored in a file. This allows a test to be developed in a modular fashion. A test can be created as a number of short tests, each for a different part of the DUT, and then all of the sections can be brought together to form the complete test.

Mixed-Signal Testing

Mixed-signal testing requires the ability to coordinate with other instruments. The HP 75000 Model D20 hardware does this by using triggers. On the pattern sequence spreadsheet there are two rows labeled Arm Trigger and Assert Marker. Entering On into a vector on the Arm Trigger row will cause the tester to wait for a trigger before executing that vector. Entering On into a vector on the Assert Marker row will cause the tester to produce an output trigger pulse.

Handshaking with the DUT

Synchronous and asynchronous buses both typically use some kind of handshaking protocol to indicate when more time is needed and when transfers can continue. Asynchronous buses typically have control lines that tell the DUT when there is data available, and other control lines for the DUT to indicate when it is ready to go on. For the control lines to the DUT, control output type pin groups can be used. For the control lines from the DUT, three conditional tests can be checked in the timing cycles. Each conditional test is defined by giving it a name and specifying a Boolean logical expression that uses ten timing module input lines (Q0-Q9) as variables. When a conditional test is enabled, a row for it appears on the timing cycle spreadsheet, with its name shown at the left (Fig. 5).

If Yes is entered into the condition row for a particular subcycle, the test will execute until a vector specifies this timing cycle. Execution of this timing cycle will continue until the subcycle containing the Yes is encountered. The tester will then wait until the result of the Boolean expression becomes true.

Synchronous DUTs typically work in a different fashion. If the DUT cannot respond at the expected time, it will change the state of a control line to indicate that it is not ready. This is sampled during the cycle, and if the DUT is not ready a wait state is inserted into the cycle. The timing module has a special input line just for situations like this. This line can be checked with the End If Ready test, which is enabled in the same way as the other conditional tests. Another row appears on the timing cycle spreadsheet when this test is enabled; it is labeled End If Ready.

If Yes is entered into the End If Ready row for a subcycle, the test will execute until a vector specifies this timing cycle. This timing cycle will continue to execute until the subcycle containing the Yes is encountered. The READY

Pattern Sequence Spreadsheet

Number Base: 10

Pin Group \ Vector	0	1	2	3	4	
Address ->	00256	00139	01234	00235		
DataIn ->	00016	X	00032	X		
DataOut <-	X	00075	X	00128		
Timing Cycle	Read	Write	Read	Write		
Arm Trigger						
Assert Marker						

Fig. 4. Pattern sequence spreadsheet with four vectors filled in.

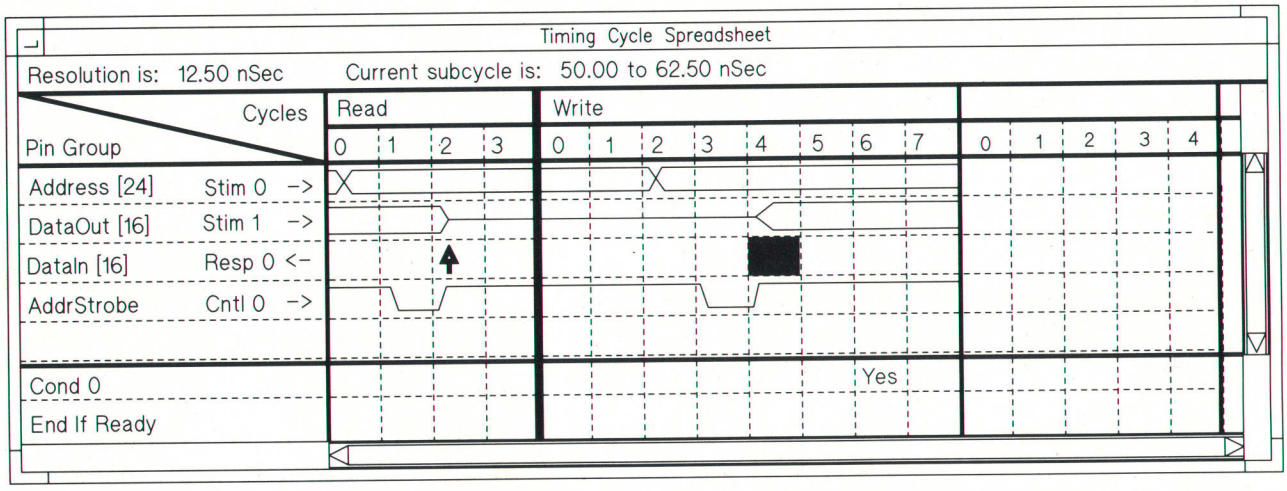


Fig. 5. Timing cycle spreadsheet with condition 0 and End If Ready at bottom.

input to the timing module is then sampled. If it is true, the timing cycle ends just as if the subcycle had been specified as the end of the cycle. If the line is not true, the timing cycle will continue to execute. The subcycles following the End If Ready would be programmed to provide the wait state. If it is necessary to check for more wait states, the process can be repeated as long as there are subcycles available. In this way, the test can be programmed to tolerate a variable number of DUT wait states.

Debugging the Digital Test

There are two aspects to debugging the digital test: running the built-in rules check to check for violations of tester rules and constraints, and running the test. The rules check does extensive checking on the test defined. Many constraints need to be observed to ensure that the test will execute in the manner expected. For example, consecutive vectors may have been defined such that when the timing cycles they specify are put back to back, they place timing events within 50 ns of each other. The rules check ensures that the test violates none of the constraints of the hardware.

To run the test, the user selects a menu choice that brings up a dialog box that allows the user to run the defined test directly on the HP 75000 Model D20 hardware. Fig. 6 shows the Run Test dialog box. From this dialog box, the user can run the test and examine any data recorded or failures found as a result of running the test. The test can be single-stepped from either the start of the test or from a breakpoint set in the pattern sequence spreadsheet. When the test is paused, execution can be continued until the next breakpoint or until the end of the test, whichever occurs first. The user can also request direct I/O, which presents a dialog box that gives the user direct control of the pins on the HP 75000 Model D20.

Producing the SCPI File

The object of the HP E1496A digital test development software is to produce a test that runs on the HP 75000 Model D20 hardware. This is done by producing an output file of SCPI commands. This file contains all of the SCPI commands to program the instrument with the defined test. It can be produced in either a portable ASCII format or a more compact binary format. The resulting file can be later incorporated into whatever test environment the user may have.

Conclusion

The HP E1496A digital test development software is an important addition to the HP 75000 Model D20 tester. It allows the user to concentrate on creating the test for the DUT instead of on programming the hardware. The graphical presentation allows the user to understand and use the features of the hardware quickly and easily. The spreadsheet metaphor allows convenient manipulation of the large amounts of data involved. The software's ability to create and debug digital tests, yet cooperate in an open environment by allowing vectors to be brought in

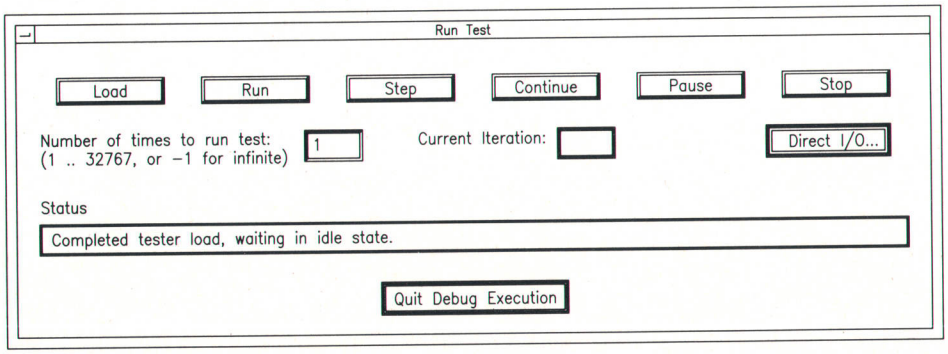


Fig. 6. Run Test dialog box.

from other sources, and its ability to produce a test that is portable to other environments, make this software a powerful tool.

Acknowledgments

The creation of this software was a team effort. My thanks to the other members of the team for their help with this paper. Special thanks to Joe Ward, Jake Smaltz, and Don Lenhert for their help as well.

The VXIbus in a Manufacturing Test Environment

Engineers at HP's Loveland Instrument Division have found that using the VXIbus and the SCPI programming language provides benefits such as reduced test development time and system support costs.

by Larry L. Carlson and Wayne H. Willis

The adoption of an industry-standard modular instrument architecture was the natural evolution for both the defense and commercial industries. Now, VXIbus is fast becoming one of the modular architectures of choice and VXIbus vendors are providing users with a broad choice of test tools such as instruments, switches, DUT interfaces, and special-purpose functions. Recently, new products with test capability not available in the HP-IB have emerged. VXIbus users have access to the same compatibility benefits the industry has enjoyed for years integrating HP-IB instrumentation. VXIbus along with the new standard instrument language, SCPI (Standard Commands for Programmable Instruments) promises users greater flexibility for building systems that are much easier and less costly to reconfigure or upgrade to meet growing test requirements.

This article focuses on the benefits of the VXIbus and SCPI in a commercial manufacturing application for functional test. The information is based on the implementation of a new test strategy and experience gained at HP's Loveland Instrument Division where precision digital multimeters and modular instruments are manufactured. This article also explains how the VXIbus and SCPI are implemented to reduce test development time and system support costs. It examines the importance of these new standards in building a standard test platform that is designed to be easily upgraded and configured for a variety of testing applications.

Functional Test on the Manufacturing Floor

The manufacturing profile at HP's Loveland Instrument Division is characterized as low-volume and high product mix. This means that production volumes are on the order of 10 to 100s per month and the total number of different products is high, more than 250. Production characteristics such as these typically mean that the cost of testing can be very high because of the need for a wide variety of test capability. Both in-circuit and functional test are employed in this production situation. Functional test is almost always used because of the need to do calibration and in-circuit test is used when the economics justify it.

Functional Tester Characteristics

Historically, modular instrumentation has been an important part of the test strategy. Test systems were typically built using HP-IB rack-and-stack instruments along with modular instrument products for housing switching and some functional capability such as digital multimeters, counters, and digital I/O. Lack of a large breadth of modular products to choose from was one of the obstacles to taking full advantage of these proprietary modular systems. Furthermore, their obsolescence meant that few, if any, replacements were available for upgrading the system, and long-term system maintenance was a problem. An industry standard architecture such as VXIbus solves the breadth of product offering and obsolescence issues.

Custom engineered products are typically built into test systems to meet special measurement and control needs for functions not commercially available. Custom equipment is expensive to design and build and costly to document and support. Modular instrument systems have provided a vehicle for custom designs because of the availability of breadboard modules that are powered by mainframe power. The package, interface, and documentation are provided by the module supplier leaving the task of designing and documenting the unique circuitry to the test engineering design team. Breadboard modules are usually available for most modular systems including VXIbus. However, VXIbus development tools (not available for the earlier modular systems) facilitate the job of custom design. The real advantage of the VXIbus is that it protects custom design investments because they can be used on other compatible test platforms on the manufacturing floor. On the other hand, because of the broad choice of VXIbus (and compatible VMEbus) functions available on the market, the need to build custom functions is greatly reduced.

Standard Test Platform

In the last ten years a large number of different test systems were developed to test the various products introduced into manufacturing. There is a very high cost associated with maintaining and supporting so many test systems. Some of the older test subsystems have become especially difficult to support because of poor parts availability and equipment obsolescence. Today, products are being developed at a faster pace, and competitive pressures are driving the need for shorter time to market. Therefore, test system development time and cost must be reduced. Also, the number of test platforms required

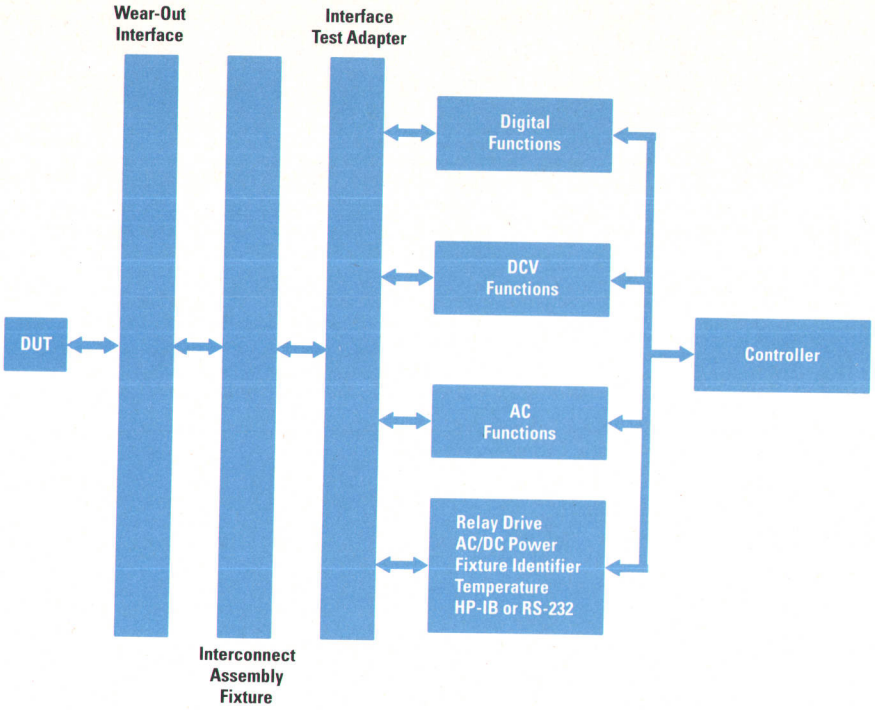


Fig. 1. Test platform for testing modular type products.

to test a rapidly expanding mix of products must be reduced. This requires devising a test strategy that not only includes an upgrade path, but also allows a sufficiently equipped system to be reconfigured easily to test any one of many DUTs.

A generic test platform using standard hardware and software architectures and a flexible DUT interface is the basis for this test strategy. Fig. 1 depicts this test platform, which can test numerous modular-type products. This test platform can be configured to test products from simple switch multiplexers to precision digital multimeter modules. A flexible DUT interface that allows rapid system reconfiguration has a replaceable wear-out mechanism for protecting the interface adapter's connections.

Because of an anticipated broad offering of products from VXIbus vendors, the generic test system can be upgraded with new capability so that it can keep pace with new product test needs and still keep the size of the test system to a minimum. Also, by including the test requirements of many existing products, we can migrate them to this new system to reduce the number of different test platforms. As manufacturing is expanded to other locations around the globe, this generic test platform becomes important in containing the cost of system duplication and support.

SCPI plays an important role in test development productivity and system upgrade strategy. Upgrading an older test system's test code, typically written in various instrument-specific languages, is difficult because the program typically has not been well-documented and the original design engineer is not available to make the modifications. The SCPI standard instrument language opens the door for developing a test software that is hardware independent. This allows substitution of new hardware (usually a superset of the old) for old hardware and requires only minor or no programming changes. Because SCPI is an integral part of an upgrade strategy, products are chosen that support it.

SCPI contributes significantly to test development productivity. Once a test programmer learns SCPI, there is no need to spend time learning the instrument-specific command language for each piece of equipment in the system. A 6:1 time savings in writing instrument drivers using SCPI has been experienced by our manufacturing test engineering group.

Floor space in manufacturing is costly. The use of VXIbus has helped reduce a three-bay system to two bays. As more VXIbus products become available, equivalent HP-IB rack-and-stack functionality will be put into the VXIbus mainframe to free space either for other rack-and-stack functionality not yet available in VXIbus, or to further size reduction.

Capacity flexibility is another key benefit of the generic test platform. Having an entire production line go idle because one instrument in the test system is down is something manufacturing managers fear. Having more than one of the generic test platforms means that if one system goes down for any reason, within a few minutes one of the others can be reconfigured to get the production line running again, thus cutting downtime losses.

Test Throughput

Without question, test throughput is an issue for the manufacturer who produces products in great volume. However, when production volume is relatively low, throughput is an issue when test system capacity is reached. The ability to increase throughput can eliminate the cost of equipping and building duplicate systems in which more equipment has to be maintained, calibrated, and supported, thus raising overall support costs. The VXIbus is expected to enhance the ability to fine tune the generic test system for optimum throughput.

So how does the VXIbus improve system throughput? VXIbus manufacturers are providing the test-and-measurement market with intelligent message-based VXIbus instrument modules whose measurement throughput is much greater than the previous generation of HP-IB instruments. Advances in instrument language interpreters and faster instrument reconfiguration designs are responsible for much of this improvement. For even higher throughput speeds, register-based programming via direct memory-mapped access to registers on either register-based or message-based instruments is possible. Having the flexibility of both register-based and message-based programming available in a single VXIbus mainframe allows test developers to develop the test program quickly in the high-level SCPI language, then improve it later with register-level programming where throughput bottlenecks have been identified. See the article on page 41 for more about register-based throughput.

Conclusion

Functional test strategies based on industry standards hold great promise for reducing time-to-market and total test cost. The VXIbus architecture can lower custom engineering investments because it is an easier environment to design and support. VXIbus can save costly rack space and protect investments from product obsolescence. VXIbus integrates easily into a system with rack-and-stack HP-IB instrumentation, and with SCPI implementations on both architectures, software development and reuse are virtually architecture independent. This gives the systems designer the greatest latitude in matching the hardware to meet the test needs. Finally, with VXIbus implementations, the test developer has the flexibility of accelerating the system's throughput to increase manufacturing productivity and minimize costly test system duplication.

Authors

April 1992

6 VXIbus System Architecture

Lawrence A. DesJardin

Larry DesJardin is VXIbus program manager at HP's Loveland Instrument Division. He also serves as secretary on the board of directors of the VXIbus Consortium. He joined HP Laboratories in 1977 after receiving a BS degree in engineering from the California Institute of Technology. He also has an MSEE degree (1978) from Stanford University. Larry has held various engineering and management positions for HP's voltmeter and computer-aided-test product lines. An expert on the VXIbus standard, he has written several technical articles on the standard and is the recipient of the John Fluke, Sr. Memorial Award for participation in its creation. His work with test equipment has resulted in two patents on instrumentation techniques. He is a member of the IEEE Instrumentation Society and the American Society of Test Engineers. Larry was born in Redwood City, California and now lives with his wife in Longmont, Colorado. Skiing, biking, scuba diving, and boardsailing are among his leisure activities.

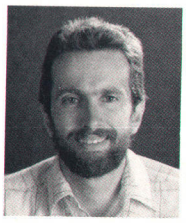
14 Designer's Perspective

Gregory A. Hill

Software development engineer Greg Hill developed two of the gate arrays used in HP's VXIbus products. He is one of the authors and architects of the VXIbus specification. His current responsibilities include participating as a technical representative to the VXIbus Consortium and as VXIbus technical coordinator at HP's Loveland Instrument Division (LID). Greg received his BSEE (1977) and MSEE (1978)

degrees from Texas Tech University. He joined LID in 1979. Before joining HP, he worked on high-voltage pulse generation and EMP testing at BDM Corporation. HP products he has worked on include the HP 3497A/3498A data acquisition and control unit and the HP 3235A switch and test unit. He also developed a process control system and software for LID's printed circuit board shop. Greg is the author of two articles on VXIbus protocol for the VXIbus Journal and is involved in VXIbus, VMEbus, and Futurebus+ standards activities. He was born in Wichita, Kansas and now lives in Loveland, Colorado with his wife and two young boys. He is active in church activities and ministries.

Steven J. Narciso



HP's VXIbus technical director Steve Narcisco joined HP's Loveland Instrument Division in 1979. He received his BSEE degree in 1978 and his MSEE in 1979 from Purdue University. He led the design team in the development of the HP 3245A universal source and assisted in the design of the HP 3235A switch and test unit. Before joining HP Steve worked for NASA designing data communication interfaces. He has written several papers on the VXIbus which have been presented at various test conferences. His efforts in circuit design have resulted in a patent for reducing settling time in filters. He was born in Englewood, New Jersey and currently lives in Longmont, Colorado with his wife and three children. In his leisure time, Steve enjoys tennis, hiking, training his golden retriever, and spending time with his children.

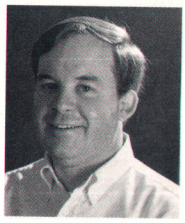
24 Mainframe Firmware

Paul B. Worrell

Currently a project manager for new software projects, Paul Worrell worked as a firmware design engineer and project manager on the VXIbus mainframe firmware. Paul began his career at HP as a summer intern at HP's Loveland Instrument Division in 1979, and became a full-time employee in 1980. He received his BSEE degree from the University of Missouri in 1979 and an MSEE from Stanford University in 1984 through HP's honors coop program. Paul has worked on the HP 3421A data acquisition control unit and the HP 44456A software package. He has also served as a technical representative to the VXIbus Consortium. Paul was born in Hannibal, Missouri and currently lives with his wife in Loveland, Colorado. His outside-of-work interests include home improvements, constructing automobiles, and building audio equipment.

30 Real-Time Multitasking

Christopher P. Kelly



Design engineer Chris Kelly was responsible for the firmware on the HP 1300 B-size and HP 1405 C-size VXIbus controllers. Chris joined HP's Loveland Instrument Division in 1983 after receiving an MS degree in computer science from Colorado State University that same year. He also holds a BS degree in biology and chemistry (1975) from the University of New Mexico. One of his past assignments was the firmware revision for the HP 3457A multimeter. Before joining HP, Chris worked on real-time data acquisition systems, analysis systems, and control systems at Lovelace Biomedical Environmental Institute in Albuquerque, New Mexico. He has authored and coauthored several articles and papers in electronic test magazines, computer science symposiums, and scientific journals. Because he likes to work on the boundary between hardware and firmware, his professional interests include real-time operating systems and data acquisition firmware. Work in church organizations, public service communications with ham radio, and severe weather spotting for the National Weather Service are some of the activities he is involved in. Born in Seattle, Washington, Chris now lives in Loveland, Colorado with his wife and three young boys. In his spare time, he enjoys amateur radio (advanced class), shooting sports, fishing, and camping.

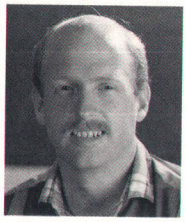
35 VXIbus Programming in C

Lee Atchison

Software design engineer Lee Atchison joined HP in 1987 at the Information Networks Division in Cupertino, California, where he worked on SNA network software drivers. He is now located at HP's Measurement Systems Operation, in Loveland, Colorado. Lee received BEE and BCS (computer science) degrees from the University of Minnesota in 1987. For the VXIbus project, he worked on the instrument I/O library, the instrument communication strategy, and the software drivers for the HP V/360 controller. Lee is a member of the IEEE Test and Measurement Societ and the IEEE Computer Society, and participates in VXIbus Consortium activities. He is also chairman of the IEEE P-1174 standards committee. His professional interests include instrument I/O, networks, and operating systems. Lee was born in Minnesota and lives with his wife in Ft. Collins, Colorado. He is a Walt Disney memorabilia collector.

41 Matrix Relay Modules

James B. Durr



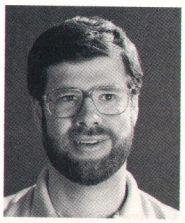
Learning products developer Jim Durr wrote the SCPI and register-based benchmark programs for the VXIbus dense matrix relay module. Jim received a BSEET from DeVry Institute of Technology in 1981. He joined HP's Lake Stevens Instrument Division in 1981 and later relocated to HP's Loveland Instrument Division. Jim has developed user manuals that cover SCPI and register-based programming for VXIbus products. He has also developed manuals for the HP 3245A universal source and for the HP 3852A data acquisition and control unit. He was born in Riverton, Wyoming and currently lives in Loveland, Colorado with his wife and two children. He enjoys bird hunting, skiing, and biking into backcountry lakes and streams.

Sam S. Tsai

The electrical and mechanical design for several of the VXIbus modules such as the HP E1472A RF multiplexer, the HP E1403A A/B active module carrier, and the HP E1466A 4 × 64 matrix switch are some of Sam Tsai's contributions to HP's VXIbus product line. Sam received a BSME degree in 1975 from National Cheng-Kung University in Taiwan, an MSME in 1979 from Texas Tech University, and an MSEE in 1987 from Colorado State University. He joined HP's Loveland Instrument Division in 1979. One of his early contributions was to the design of the HP 3235A switch and test unit. Sam was born in Taiwan and now lives in Westminster, Colorado with his wife and two daughters. Teaching Mandarin to his children, watching movies, and landscaping are some of his leisure activities.

52 Mass Interconnect

Calvin L. Erickson



A graduate of Colorado State University (BSME 1978) and the University of Manchester Institute of Science and Technology (MScEE 1985) in Manchester, England, Calvin Erickson joined HP in 1979 at the former Civil Engineering Division in Loveland, Colorado. Calvin was the project manager for the VXIbus C-size system hardware and switches and was the mechanical representative on the original VXIbus standardization committee. He has worked as the mechanical engineer on various voltmeter and cardcage product developments at HP's Loveland Instrument Division. At the Civil Engineering Division, Calvin worked on distance meter product development. He participates in the development of young scientists and engineers by working in the visiting scientist program at the high school where his wife teaches. He is also active in church activities.

Calvin was born in Greeley, Colorado and currently lives in Loveland, Colorado. Travel, hiking, and skiing are his main interests outside of work.

59 Digital Test Equipment

David P. Kjosness

The development of the HP E1450A timing module was the main responsibility of design engineer Dave Kjosness. Dave joined HP in 1979 after receiving an MSEE degree from the University of Colorado that same year. He also received a BA degree in physics and mathematics in 1976 from Western State College of Colorado. In the past Dave has worked on an integrated circuit for a fractional-N frequency synthesizer, a timing generator for a proprietary integrated circuit tester, and various switching modules for the HP 3235A switch and test unit. His work has resulted in two patents related to RF switching, and his professional interests include high-speed analog and digital circuitry. Dave was born in Gunnison, Colorado and now lives in Longmont, Colorado. He is married, has two children, and is active in scouting. To stay in shape, Dave enjoys running, bicycling, and weightlifting. He also enjoys camping and handgun shooting.

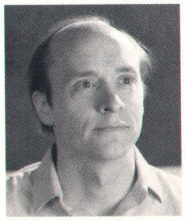
69 Development Software

Kenneth A. Ward

Ken Ward is a member of the team that developed the software for the HP 75000 Model D20 digital functional tester. He joined HP's Loveland Instrument Division in 1982. He received a BSEE degree (1978) from Kansas State University and an MS in computer and information science (1980) from Ohio State University. Before joining HP, he worked as a software engineer at Bell Telephone Laboratories in Columbus, Ohio. Ken is a member of the IEEE and the ACM. Born in Manhattan, Kansas, he now lives with his wife in Loveland, Colorado. His leisure activities include playing trumpet with several local groups, hiking, and camping in the mountains.

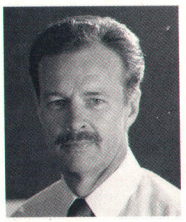
75 VXIbus in Manufacturing

Wayne H. Willis



Formerly a manufacturing test engineering manager at HP's Loveland Instrument Division, Wayne Willis is currently a sales development manager at the same division. Before joining HP in 1979, he worked as an R&D engineer for a company that manufacturers water analysis instrumentation. Wayne received a BSEE degree from Iowa State University in 1979. He was born in Marshalltown, Iowa and currently lives with his wife and three children in Loveland, Colorado. Wayne spends much of his spare time playing with his children and is an avid computer hobbyist.

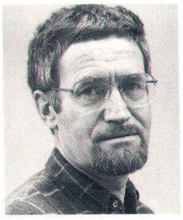
Larry L. Carlson



Currently a product manager at HP's Loveland Instrument Division, Larry Carlson joined HP in 1962 after receiving a BSEE degree from Colorado State University that same year. He also received an MSEE in 1967 from the same school. Larry has held positions in production engineering, product development (analog voltmeters), and marketing. His publications include several application notes, and he is a coauthor of a basic instrumentation handbook. He was born and currently lives in Loveland, Colorado with his wife and two children, both of whom are studying engineering at the University of Colorado. Larry's hobbies and other interests include tennis, hiking, and biking.

81 Peak Power Analyzer

Dieter Scherer



Dieter Scherer was project manager for development of the HP 8990A peak power analyzer at HP's Stanford Park Division. Born in Bad Reichenhall in Bavaria, he attended the Technical University of Munich, graduating in 1967 with a Diplom Ingenieur in electrical engineering. At Stanford Park, which he joined in 1967, he designed wideband hybrid amplifiers, microwave oscillators, and mixers for the HP 8660 family of synthesizers, was responsible for the RF modules of the HP 8662A synthesized signal generator and the low phase noise circuits for that instrument, developed GaAs hybrid circuits for microwave signal generators, and served as project manager for phase noise measurement instrumentation and the HP 8990A. One patent on a gain strobing technique has resulted from his work. In 1972 he received an MSEE degree from Stanford University. A senior member of the IEEE, he has authored papers on low phase noise design, RF synthesizer design, phase noise measurement, peak power sensors, and peak power measurement. Since 1981, he has been a part-time lecturer in RF design at Stanford University. Dieter's interests include furniture design, photography, hiking, and boardsailing. He's married and has two sons.

William E. Strasser

Bill Strasser did analog design for the RF power and oscilloscope channels of the HP 8990A peak power meter. Since joining HP's Stanford Park Division in 1984, he has designed power, phase noise, and noise figure instrumentation. A member of the IEEE microwave theory and techniques group, he is named as a co-inventor on two patents, one on a wide dynamic range amplifier and one on gain strobing. Bill was born in Washington, D.C. He received his BSEE degree in 1984 from Lehigh University and his MEE degree in 1986 from Cornell University, specializing in microwave theory and electrodynamics. He is married. When he's not at HP or working on his home, he enjoys boardsailing and other outdoor activities.

James D. McVey

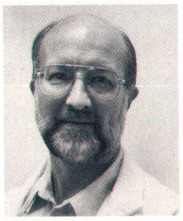
R&D engineer Jim McVey was one of the designers of the HP 8990A peak power analyzer at HP's Stanford Park Division. He came to HP in 1985 after receiving BS and MS degrees in electrical engineering from Stanford University in 1984 and 1985. In addition to the HP 8990A, he has contributed to the design of the HP 8780A and 8782A vector signal generators, the HP 11759B RF channel simulator, and the HP 8981B vector modulation analyzer.

Wayne M. Kelly

Wayne Kelly received his BSEE and MSEE degrees from California State University at San Jose in 1963 and 1965. After ten years with GTE Sylvania designing microwave components and subsystems, he joined HP's Microwave Semiconductor Division (then HP Associates) in 1975 and designed a radar preamplifier. After moving to the Stanford Park Division, he designed a vector modulator for the HP 8780A and HP 8782A vector signal generators and a frequency agile upconverter for the HP 8791 Models 11 and 21 frequency agile signal simulators. For the HP 8990A peak power analyzer, he designed the 1.05-GHz, +10-dBm sensor check source. A member of the IEEE microwave theory and techniques group, he has authored a paper on a figure of merit for X-band GaAs power amplifiers, and his work has resulted in a patent on mixers and multipliers using slot/coplanar transmission line. Born in Oakland, California, Wayne is married and has two children. He's a skier and has developed the special skill of building massive wood decks on concrete retaining walls.

90 GaAs Technology

Michael C. Fischer



Development engineer Mike Fischer has been with HP since 1973. He has worked on phase noise measurement, developed frequency standards, and contributed to the development of the HP 8990A peak power analyzer, for which he did system design and testing, sensor amplifier design and testing, and verification source design. He's presently doing producibility improvement engineering. A member of the IEEE and the Audio Engineering Society, he

has authored many technical papers and articles on the measurement and control of phase noise and frequency stability, and is named an inventor on seven patents on signal processing and measurement and on frequency standards. A native of Coleman, Texas, he received his BSEE degree from the University of Texas in 1966. Before coming to HP, he designed frequency standards and navigation receivers for Tracor, Inc. and Magnavox. Mike has three children. He serves as a Boy Scout leader and is interested in automobiles, audio equipment, sailing, and photography.

Michael J. Schoessow

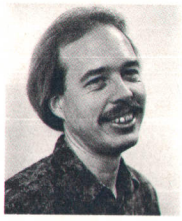
Mike Schoessow joined HP's Stanford Park Division in 1981 as a production engineer, then moved to R&D in 1987. Since then, he has contributed to the design of the HP 86792A agile upconverter and the HP 70100A power meter. For the HP 8990A peak power analyzer, he was responsible for the baseband circuit design, ground management, and electrical interference management. A member of the Audio Engineering Society, he has coauthored papers on graphic equalizer design and loudspeaker equalization. Mike was born in Milwaukee, Wisconsin and received his BSEE degree from the University of Wisconsin in 1981. He is married. His interests include ultrahigh-voltage research, kite flying, restoring antique radios, gourmet cooking, and hiking. In his spare time, he works on a design for the "ultimate"communications receiver.

Peter Tong

Now studying for JD and MBA degrees at Santa Clara University, Peter Tong has moved from the R&D lab to become a legal associate on HP's corporate legal staff. A native of Hong Kong, he received his BSEE degree in 1980 from the University of Hawaii and his MSEE and PhD degrees in 1981 and 1985 from the California Institute of Technology. As an R&D engineer with HP's Stanford Park Division starting in 1984, he was responsible for the design and production introduction of the HP R347B and Q347B noise sources and for microwave design and calibration of the HP 84812/13/14A peak power sensors. He has published eight technical papers in the fields of microwave design, optical systems, and semiconductor technology. Peter is married, enjoys tennis and swimming, and is presently serving as vice president of the California Institute of Technology Alumni Association.

95 Automatic Calibration

David L. Barnard



A software design engineer with HP's Stanford Park Division, Dave Barnard was responsible for firmware design for the HP 8990A peak power meter. With HP since 1980, he did reliability engineering for two years before moving to the R&D lab, where he worked on the HP 11729C carrier noise test set. He received his BSEE degree in 1980 from the University of Colorado at Boulder and is a member of the IEEE. Dave is active in his church and is interested in music, particularly guitars, keyboards, and MIDI sequencing. He enjoys traveling in Latin America and studying the culture, and he participates in a bilingual tutoring program. He's a native of Longmont, Colorado.

James A. Thalmann

Jim Thalmann received his BSEE degree from the University of Illinois in 1976 and his MSEE degree from Stanford University in 1979. Since 1976, he's been a development engineer with HP's Stanford Park Division and has contributed to the design of the HP 8683/84 signal generators, the HP 11729A carrier noise test set, and the HP 11759A/B RF channel simulators. He also worked on the firmware and automatic calibration scheme for the HP 8990A peak power analyzer. In his spare time, he's an avid cyclist.

Henry Black

Henry Black joined HP's Stanford Park Division in 1988 as an R&D engineer. He was responsible for firmware development for the HP 70100A MMS power meter and the HP 8990A peak power analyzer. Before coming to HP, he did early design work on synchronous data communications with ICL and designed 60-Hz energy measurement instrumentation with Robinton Products, Inc. He recently left HP to become chief engineer with Information Consultants, Inc. A registered professional engineer and an amateur radio operator, he was born in Wallasey, England and received his BA degree from The Open University.

101 500-MHz Network Analyzer

Koichi Yanagawa

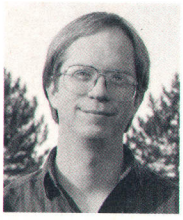
Koichi Yanagawa was project manager for development of the HP 8751A network analyzer. He's been with Yokogawa-Hewlett-Packard since 1974 and has contributed to the design of the HP 4194A impedance/gain-phase analyzer, the HP 4278A capacitance meter, and the HP 42841A bias current source. His work has resulted in one U.S. patent and several Japanese patents on analog circuits. Born in Toyama, Japan, he received his BS degree in electrical engineering from Tohoka University in 1971 and his MS degree in electrical engineering from Kyoto University in 1974. He's a member of the Institute of Electronics, Information, and Communication Engineers. Koichi is married, has two children, and enjoys music and hiking.

110 Measurement Coprocessor

Michael P. Moore

Mike Moore was project leader for the HP 82324A high-performance measurement coprocessor. He joined HP's Corvallis Division in 1978 and contributed to the design of the HP 85 and HP 87XM computers and the HP 82300A and HP 82324A measurement coprocessors. When the HP 82324A project moved from the Corvallis Workstation Operation to the Measurement Systems Operation (now the Measurement Software Division), Mike followed it to Colorado, where he now lives. A native of San Francisco, he graduated from the U.S. Naval Academy with a BS degree in electrical science in 1968 and served ten years in the U.S. Navy, attaining the rank of lieutenant commander. In 1975 he received his MSEE degree from Oregon State University, specializing in solid-state electronics. He is married, has two daughters, and enjoys camping, hiking, fishing, cross-country and downhill skiing, and running.

Eric N. Gullerud



Eric Gullerud is a software development engineer with HP's Measurement Software Division. Born in Colorado Springs, Colorado, he received his BSEE degree from the University of Illinois in 1981 and joined HP's Corvallis Division the same year. He served as a chip designer for HP Series 10 calculators, developed software for Northwest IC Division design tools and for IC testing, and developed software for the HP 82300A and HP 82324A measurement coprocessors, moving back to his native Colorado when the HP 82324A project moved. Eric is married and has three children. His leisure interests include music, camping, and hiking.

The Peak Power Analyzer, a New Microwave Tool

Gallium arsenide sensor design, a new calibration approach, switched amplification and processing of the envelope signals, leveraged digital oscilloscope technology, and microprocessor control provide calibration-free, accurate pulsed microwave power measurements.

by Dieter Scherer, William E. Strasser, James D. McVey, and Wayne M. Kelly

Microwave signals used in radar, telemetry, navigation, and communications applications are typically pulsed. A Doppler radar, for example, operates on the basis of pulse modulation and is critically dependent on the characteristics of the pulse power envelope, such as peak and average power, rise and fall times, pulse width, duty cycle, and pulse repetition rate. For satellite communication links in TDMA (time division multiple access) mode, the concerns are absolute and relative power levels and delays between individual pulses within a frame or from frame to frame. Even if the microwave signal is only frequency keyed or phase switched, like a QPSK signal in a digital radio application, it is often important to measure power overshoot and ringing at transition times.

The performance of systems like these can be determined to a large degree by accurate power and time measurements of the microwave pulse envelope. One stage earlier, the design, performance, and stress of the components making up such systems can be characterized by pulse parameters such as rise time, overshoot, peak power, power compression, spike leakage, pulsed gain and reflection, and gain suppression. Delay measurements are also important, not only along the microwave signal path, but also between the control signal and the microwave pulse response. Generally, the microwave response of a system or component to control stimuli is relevant in analyzing pulsed performance.

Common Ways to Measure Pulsed Microwave Power

What are the common ways to measure pulsed microwave power? The simplest approach is the use of a thermocouple sensor and a power meter like the HP 437B. This combination reads the average power of the microwave pulse train. Knowing the pulse duty cycle and assuming a rectangular pulse shape, the HP 437B can calculate the pulse-top power.

The spectrum analyzer can also be a tool for measuring microwave pulses. For correct amplitude determination, the desensitization factor needs to be added to the displayed amplitude and the user has to distinguish between line and pulse spectra. The spectrum analyzer yields information on pulse repetition frequency, pulse width, and pulse power. The accuracy of the latter two measurements again depends on the assumption of rectangular pulse shape. Either method is highly inaccurate if the pulses show significant overshoot, ringing, or droop, and wouldn't work at all with an irregular pulse train like the pulsed frames of TDMA signals.

High-bandwidth oscilloscopes can be used in these cases. The HP 54124T, for instance, is capable of displaying the RF signal directly up to 50 GHz. While the sampling oscilloscope has plenty of speed to observe the rise time and ringing in the applications mentioned above, it falls short with regard to accuracy and dynamic range.

Diode Detector with Oscilloscope

None of the above solutions can comprehensively satisfy the measurement needs indicated at the beginning of this article. What is missing is an instrument that accurately detects and displays the power envelope of complex pulsed microwave signals, allows instantaneous power measurements on the pulses and convenient measurements of pulse parameters, all at an economic price.

A potential solution has been in use on benches for a long time: a microwave detector connected to a 50-ohm oscilloscope input. However, it is difficult and cumbersome to make meaningful measurements on the displayed pulse with this setup. The video output of the detector is far from being an accurate replica of the microwave pulse envelope, voltage or power. The causes reside in part in the nonideal detector, and in part in the limitations of the oscilloscope.

Detector Nonlinearity. The diode detector has a square-law response at low power (the video voltage is proportional to the square of the RF voltage), a linear response at high power, and a wide transition range in between (approximately -15 to $+10$ dBm). Even if the user manages to calibrate the top level of the displayed signal with a substituted known RF signal, any measurement involving other levels, such as rise time, is highly inaccurate.

Detector Temperature Dependence. The temperature sensitivity of the diode detector is high and complex, changing rapidly with signal level and temperature. For instance, a 10°C change from calibration can cause errors in power measurement as great as 50%. Very large errors can result

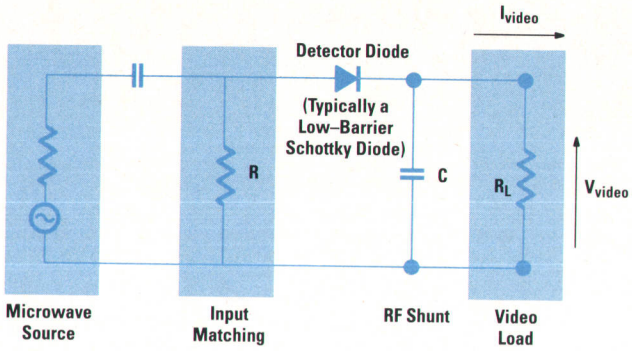


Fig. 1. The basic diode detector.

from connecting the detector to a device under test that is hot or warming up.

Frequency Response of the Video Output. The diode junction capacitance and device lead inductance are the major device parasitics affecting the frequency response of the detector output. The junction resistance also changes at high signal levels, which indirectly causes the frequency response to vary with the signal level to some degree. The unflatness of a low-barrier Schottky diode detector in X band may be around ±0.3 dB, which translates to a ±7% error in power.

Mismatch Error. Mismatch represents a major error source. It is proportional to the product of the reflection coefficients of the detector and the device under test (DUT). A typical low-barrier Schottky detector (SWR <1.5) connected to a DUT with an SWR of 2.0 may give a maximum mismatch error of ±13%. Unless the detector and DUT impedances are accurately known in magnitude and phase, this error cannot be calibrated out.

Microwave Signal Harmonics. The microwave signal and harmonics of the signal are detected as a vector sum when the detector operates in the linear range. A −20-dBc harmonic may therefore cause as much as a 21% error in power measurement.

Slow Video Response. A detector requires a RF bypass capacitor on the video side (see Fig. 1). In combination with the capacitive loading of the oscilloscope, this capacitor slows down the rise time of the pulse envelope signal. The time constant can be decreased by choosing a low-impedance termination for the oscilloscope input, say 50 ohms. Unfortunately, the lower impedance also causes a corresponding decrease in detector sensitivity because the diode detector at low signal levels represents a current source.

Limited Oscilloscope Sensitivity. The low video signal levels obtained with a low-impedance termination are two orders of magnitude below typical oscilloscope sensitivities.

This list of obstacles doesn't yet include the problems of accurately calibrating the measurement. Nevertheless, the concept of envelope detection is viable and holds the promise of an economic solution. It requires only one broadband microwave component to translate the pulsed microwave signal to a video signal where it can be processed at much lower cost.

Commercial solutions based on the principle of diode detection have been on the market for some time. For example, the HP 8900C/D peak power meters use diode detectors, but represent only partial solutions. More advanced are several non-HP peak power meters which address the detector deficiencies with a calibration source and limited calibration processing.

New Peak Power Analyzer

The HP 8990A peak power analyzer (Fig. 2) is a new type of instrument that represents a comprehensive solution to the problem. The design team took a fresh look at the challenges of diode detection. Our goal was to transform the inaccurate, cumbersome bench setup into a carefree product that measures accurately and meets the complex measurement requirements of modern microwave systems. The use of GaAs IC technology in the sensor design, a new calibration approach, switched amplification and processing of the envelope signals, broad leveraging of modern digital oscilloscope technology, and extensive use of microprocessor power in signal calibration and processing accomplished this task.

Fig. 3 is a block diagram of the HP 8990A peak power analyzer.

Sensor. A new approach was taken in the sensor design.[1] It involves a GaAs-based diode technology, integration of a balanced circuit on GaAs, and a new calibration scheme (see the article "GaAs Technology in Sensor and Baseband Design," page 90). The use of planar doped barrier diode technology improves the frequency response and consistency of the diode characteristic.

A balanced diode circuit including the terminating load is implemented as a GaAs IC. The integration of the detector circuit minimizes mismatch loss, thanks to the very low parasitic reactances on the IC. The balanced circuit effectively minimizes the potential problem with even-order harmonics of the RF signal. Both integration and balanced design help minimize the effect of thermoelectric voltages, which could otherwise mask low-level signals.

A three-dimensional calibration scheme takes care of the problems of nonlinearity, temperature dependence, and frequency response. Extensive calibration data is taken over a wide range of power levels, frequencies, and temperatures. Condensed as a matrix of coefficient sets, the calibration data is stored on an EEPROM supplied within the sensor. When a sensor is connected to an analyzer, the analyzer reads the calibration matrix and reconstructs a precise curve of sensor output voltage versus input power for a given frequency and temperature. The frequency is entered by the user, and the temperature is measured by a thermistor chip close to the detector IC. The analyzer continuously reads the thermistor chip and automatically triggers the rebuilding of the sensor curve for any minor temperature shift. This feature of carefree calibration is especially helpful when a sensor at room temperature is connected to a hot device under test, such as a power amplifier (see the article, "Automatic Calibration for Easy and Accurate Power Measurements," page 95).

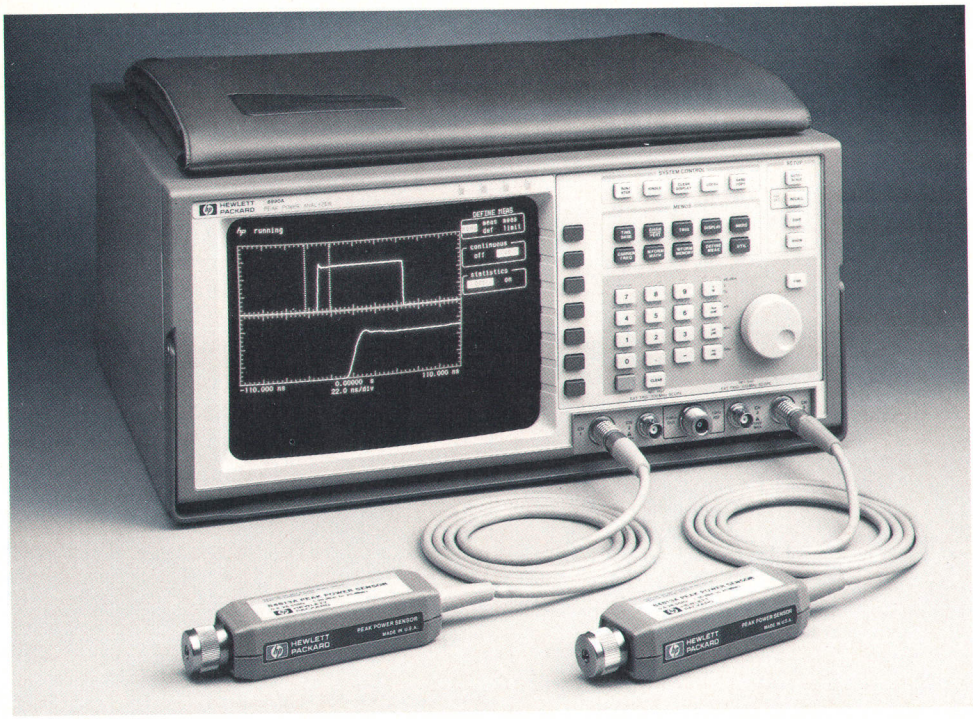


Fig. 2. The HP 8990A peak power analyzer has two 500-MHz-to-40-GHz microwave input channels and two 100-MHz video input channels. It makes calibration-free power measurements on pulsed microwave signals and can display the video driving signals simultaneously. It operates with the HP 84812/13/14A family of peak power sensors.

Currently, three sensor versions cover input frequencies from 500 MHz to 18 GHz, 26.5 GHz, and 40 GHz, respectively. Sensors are specified from +20 dBm to −32 dBm and are usable down to −40 dBm.

Sensor Amplifier. The problem of slow video response must be addressed right at the detector output with the sensor amplifier. The system goal of <5-ns rise time requires special measures to deal with pulse flatness as well as drift and low-sensitivity issues. Conflicts between speed (video signals range from dc to 10-ns-wide pulses) and dc stability are resolved with a split-path design (see article, page 90). A microprocessor-controlled chopping circuit ahead of all dc-coupled circuits performs an autozeroing

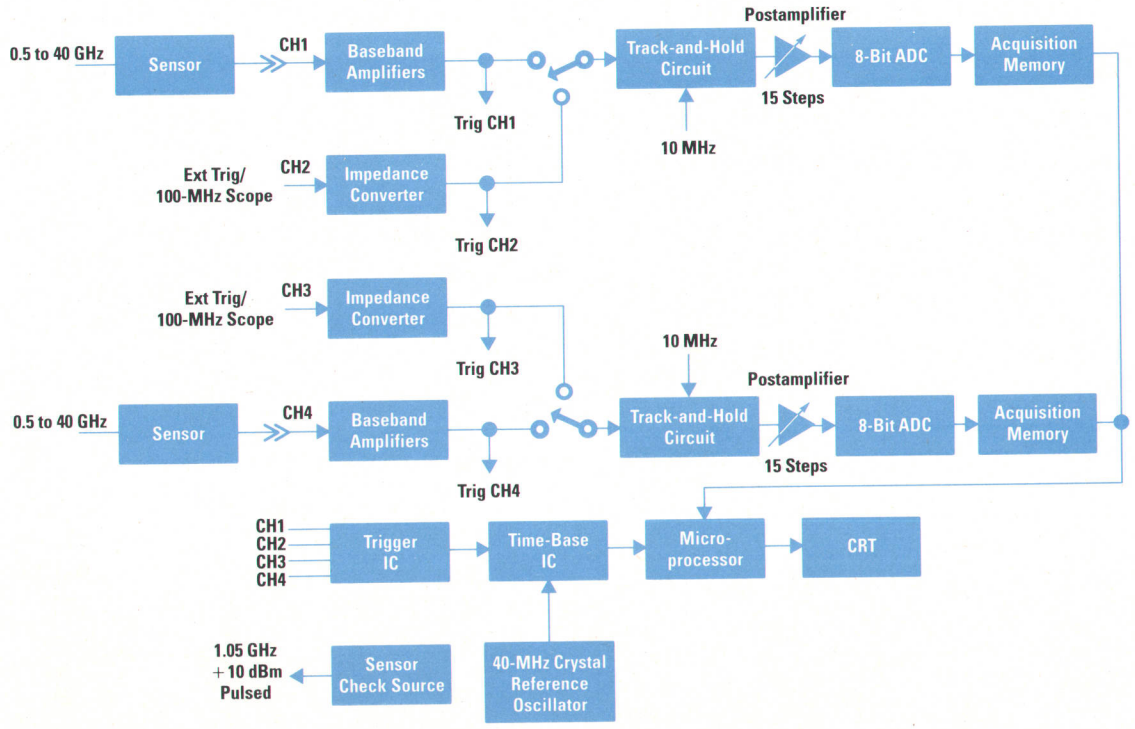


Fig. 3. Functional block diagram of the HP 8990A peak power analyzer.

Multilayer Shielding Protects Microvolt Signals in High-Interference Environment

Multilayer shielding is employed in the HP 8990A peak power analyzer to minimize interference from magnetic field sources. All three shielding mechanisms—reflection, absorption, and shunting—are used where appropriate (see Fig. 1).

The leveraging of an HP digital oscilloscope chassis, display, display driver, and power supply creates a highly constrained design problem because the field source and receiver geometry is fixed. Consequently, the interference cannot be reduced by orientation or physical separation of the sources and receivers. The most sensitive receivers are the baseband boards with their almost 100 dB of gain. Aluminum castings provide channel-to-channel isolation, gain stage isolation, and substantial electric field shielding. However, these onboard shields are ineffective in combating the magnetic fields emitted by several internal components. In addition to the switching power supply operating at 40 kHz, the CRT and the CRT driver circuitry emit magnetic fields at 60 Hz and 25 kHz, plus harmonics.

The low frequency and close proximity of these magnetic sources result in a low wave impedance, which is difficult to reflect with a conductive shield. Common practice suggests the use of a high-permeability shield enclosure to shunt the flux away from the baseband boards, and a mu-metal enclosure around the baseband boards was designed for this purpose. With an assist from the partial CRT enclosure, this enclosure lowers the 60-Hz interference to an acceptable level. However, the mu metal loses most of its permeability at the higher (> 25 kHz) frequencies, decreasing its ability to shunt the flux. An additional shielding mechanism, absorption loss, is caused by ohmic losses and is proportional to the square root of the product of frequency, permeability, and conductivity. Although the mu metal contributes some absorption at these frequencies, it is not enough. Even 1045 SAE steel would probably do better.

To achieve the necessary shielding, an additional layer of 0.014-inch copper is placed on the baseband shield on the sides facing the sources. This layer has a negligible effect on the low frequencies, but it is increasingly effective as the frequency increases. For example, although the copper does not shunt the fields, it reflects an estimated 18 dB and absorbs an estimated 8 dB at 25 kHz. Moreover, at the 100-kHz harmonic of the fast CRT sweep, these values increase to 25 dB reflection and 15 dB absorption. This increasing attenuation of the higher harmonics lowers the peaks of the 25 kHz waveform in the time domain to an acceptable level.

The mu-metal CRT shield is used to lower the 60-Hz field further and has only a minor effect on the higher (> 25 kHz) frequencies. This shield has an unusual shape in that it does not fully enclose the yoke, which is the recommended method of shunting the field. Although this geometry seriously limits its performance, the CRT beam path is not distorted, and the yoke need not be magnetically adjusted to the shield. In addition, it is easily installed and removed. The installation of the baseband and CRT shields requires the addition of only two screws to the original chassis.

The multilayer shielding achieves the required 40 to 55 dB of near-field shielding while leveraging the inexpensive CRT and CRT driver components.

James L. Bertsch
Charles W. Cook
Development Engineers
Stanford Park Division

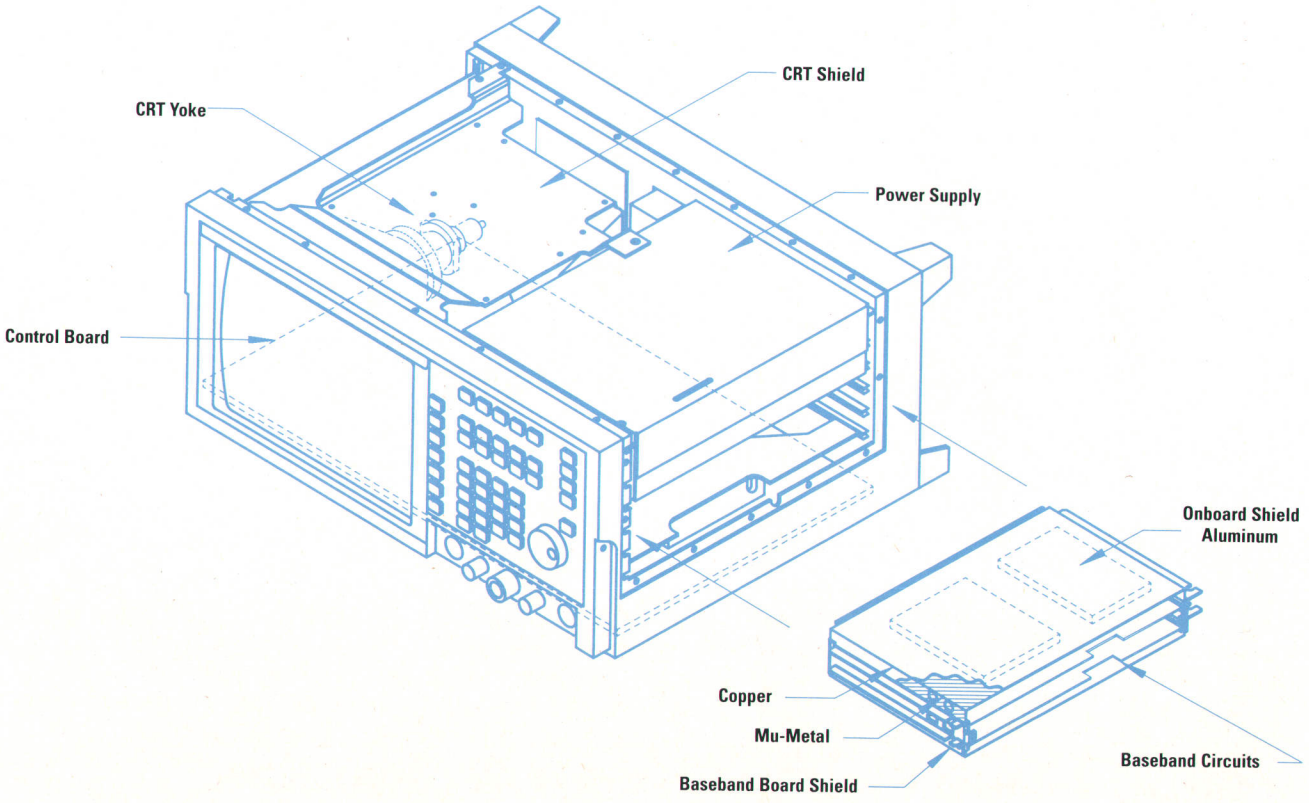


Fig. 1. Shielding in the HP 8990A peak power analyzer.

function at appropriate breaks in the acquisition process. This process, which is transparent to the user, allows the system to correct offsets and offset drift with temperature along the dc-coupled signal path. The dc and fast paths are reunited before leaving the sensor head. Both ends of the standard 5-foot sensor cable need to be well-matched to avoid video signal reflections.

Baseband Circuits. Moving from the sensor to the instrument, the video signal undergoes 94 dB of switchable gain. This gain resolves the problem that the limited sensitivity of an oscilloscope would present with microvolt signal levels. The dc-coupled amplifiers, switched with GaAs switch ICs, fulfill a dual task. First, they project the sensor video signal in coarse gain steps into the limited dynamic range of the track-and-hold circuit and later the analog-to-digital converter (ADC). Second, they limit broadband noise. The bandwidth of the amplifiers changes from >150 MHz for higher signal levels to 2.5 kHz for the lowest power decade. Correspondingly, the system rise time increases from a specified <5 ns to <250 μs (see article, page 90).

Acquisition and Digital Signal Processing Circuits. The acquisition circuits following the baseband circuits and the digital signal processing circuits are to a large degree leveraged from the digital oscilloscope technology of the HP 54500 family. The pulse envelope signal, roughly scaled by the baseband circuits, is subsequently sampled by a track-and-hold circuit at a rate of 10 MHz (see "Acquisition Process" later in this article). The hold level of the track-and-hold output then passes through a postamplifier with 15 fine gain steps. In combination with the baseband gain blocks, the postamp guarantees that any input level in the specified -32-to-$+20$-dBm range can fill the ADC window.

The 8-bit resolution of the flash converting ADC represents a bit range of 0 to 255. This translates to 24 dB of dynamic range referenced to the sensor input if the sensor were to operate solely in the square-law range, or to 48 dB for operation strictly in the linear range. The wide transition range between square and linear operation causes most applications to fall between these numbers. The ADC output is captured by the 2K-byte-wide circular acquisition memory.

The task of accurate time placement of the received pulse envelope is carried out by the powerful trigger and time-base ICs. A 40-MHz crystal oscillator is responsible for the time-base accuracy of 0.005%. A 68000 microprocessor controls the signal processing and the monochromatic 9-in display. The control code resides on a separate memory board and takes up roughly 600K bytes of ROM space.

Dual Sensor and Trigger/Oscilloscope Channels. The peak power analyzer is equipped with two sensor channels: channels 1 and 4. This facility allows pulse comparisons and delay measurements at different probing points along a microwave path or between systems. With the capability of displaying ratios of the channel inputs, the HP 8990A can measure pulsed gain and pulsed return loss (using external directional couplers). The sensor channels have internal triggering down to -30 dBm and a trigger bandwidth of 1 MHz.

Multiplexed with each sensor channel is a video input channel. The purpose of the video channels (channels 2 and 3) is twofold: they can be inputs for external trigger signals when fast triggering (bandwidth <100 MHz) is required, and they also serve as oscilloscope inputs with 100-MHz bandwidth and limited sensitivity (100 mV/div to 500 mV/div). With these channels, the HP 8990A can simultaneously display control signals and the resulting microwave pulse envelope and measure delay times between them. It can also measure transfer characteristics like the power-versus-control-voltage sensitivity of a pulse modulator.

Sensor Check Source. A built-in source provides a pulsed or CW signal of +10 dBm ± 0.5 dB at 1050 MHz. This serves two purposes. First, it acts as a source to verify the operation of a sensor. Peak power sensors are frequently used around high-power signals. The sensor check source is a convenient signal for checking the sensors if the user suspects sensor damage after an inadvertent connection to a high power level (damage level is specified at 1W peak power for 1 μs, not to exceed 200 mW average). Second, the check source supplies a signal for time calibration of the trigger circuits and the timing between the four channels.

Acquisition Process

The time-base portion of the acquisition process is a duplicate of that found in the HP 54500 family of oscilloscopes, while the vertical hardware processing was modified to accommodate the greater dynamic range requirements of the HP 8990A.

The sampling method used in the peak power analyzer is random repetitive sampling. In contrast to real-time sampling, where the sampling rate must be at least twice the highest frequency of the digitized signal (Nyquist rate), random repetitive sampling can sample at less than the Nyquist rate and still avoid aliasing.[2] Consequently, lower-speed circuits can be used in random repetitive sampling to achieve the same nominal bandwidth as real-time sampling.

The transition from a higher-bandwidth circuit requirement to a lower bandwidth is achieved in hardware through the track-and-hold diode bridge. Ideally, the bridge is modeled as an SPST switch operating at 10 MHz, closed during the track mode for 50 ns and open during the hold mode for 50 ns. In the track mode, the capacitor tracks the input signal, and in the hold mode the capacitor is isolated from the input signal. The charge residing on the capacitor during the hold time is a remnant of the input signal immediately before the bridge opened. During this 50-ns hold period, the 8-bit flash ADC digitizes the waveform. Conversion is initiated at a time when the sampled signal has settled.

The practical bandwidth limitations in the track-and-hold process are a nonzero hold capacitance and a finite diode switching time. For the HP 8990A, this translates to approximately 2-ns rise and fall times or a 175-MHz

bandwidth. Random repetitive sampling provides 100-ps resolution at the fastest time-base settings. This theoretically translates to an effective sampling rate of 1/100 ps = 10 GHz.[2] Of course, this bandwidth is not realized because the track-and-hold circuitry limits the speed and thus acts as a surrogate anti-aliasing filter.

The 100-ps resolution results from the time-base IC and the fine interpolator circuit, which time-stretches the uncertainty of one 40-MHz clock cycle. The 40-MHz clock is used to count the separation between the trigger event and the nearest data sample. Hence the possible error is from zero to one full clock cycle or equivalently from 0 to 25 ns. Instead of truncation, this residual time is fed into a time-stretcher circuit which accurately expands the time duration of the signal. The stretched signal is then counted with the same 40-MHz clock. The resultant uncertainty of one full clock cycle is divided by the stretch ratio, which is 250 in the HP 8990A. Hence the equivalent uncertainty (one clock cycle) is 1/(40 MHz)/250 = 100 ps.

The input signal is continuously sampled at 100-ns intervals for the fastest time-base settings. The sampled data is then successively placed into a circular acquisition RAM that has 2048 frames (see Fig. 4). At the end of an acquisition cycle, the RAM data's location relative to a trigger event that occurred during the acquisition cycle is determined. The samples are taken at exactly 100-ns intervals and therefore only one data sample's position relative to the trigger event need be determined to place the data appropriately in time. Both pretrigger and post-trigger data is gathered for each trigger event. The proportions of pretrigger and post-trigger data are determined by counters, which are set according to the time-base range and delay settings and the choice of left, center, or right screen placement.

The sampling signal and the input signal are asynchronous, so eventually all of the time slots, or buckets, will be filled with samples. Since the samples are randomly skewed in time between successive trigger events, no missing data or holes result in the recreated waveform. Thus a requirement of the input signal is that it be repetitive with a stable trigger event. For slower time-base settings the time buckets will be greater than 100 ps wide, reflecting the display's finite number of horizontal bits.

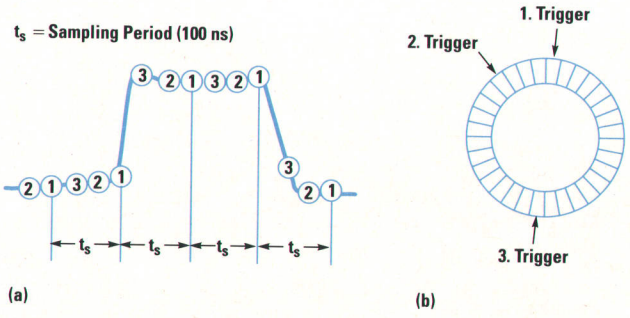


Fig. 4. (a) Random repetitive sampling. The number at each sample indicates the trigger event used. (b) Circular acquisition memory with 2048 time "buckets".

Continuous random repetitive sampling offers major advantages over sequential sampling. In sequential sampling only one sample per trigger is taken, with each successive trigger having an increased sample delay. In random repetitive sampling, data is continuously acquired at the sample rate, thus achieving a much faster display and providing pretrigger and post-trigger data as well.

While the oscilloscope design is mainly aimed at the analysis of repetitive signals, it is also capable of capturing single-shot events. The 10-MHz sampling rate records the event with sample points every 100 ns. With a criterion of 10 sample points per event, the HP 8990A offers a single-shot bandwidth of 1 MHz.

Microwave Pulse Measurement Features

Digital signal processing makes full use of the powerful timing and trigger ICs and the properties of random repetitive sampling. The resulting features have already found wide acceptance in the HP 54500 digital oscilloscope family, whose feature set was heavily leveraged in the HP 8990A. The following are some of the capabilities that are most important for microwave pulse measurements.

Time Windowing. The user of a peak power analyzer often needs to analyze a detail on a single pulse while keeping the full pulse train in view. Time windowing provides this horizontal zoom capability and allows measurements within the time window.

Trigger Conditioning. Many microwave applications present complex trigger situations that require more than a simple edge trigger function to achieve a stable display. Trigger holdoff prevents recurrent triggering on the subsequent edges of nonperiodic pulse trains, pulse packets, or bursts. Pattern trigger helps to specify a particular pulse within a frame of pulses to be triggered on, a useful feature when chasing sporadic misfires of a microwave transmitter, for example. Trigger delay, specified in time or pulse count, can be useful on long pulse trains to zoom in selectively on a particular pulse.

Persistence and Envelope Mode. Radars often operate with pulses that are extremely narrow compared to their pulse repetition interval. The duty cycle can be 0.01% or less. These signals are quite a challenge to find since most of the time the pulses fall between sample times and are not captured in the limited number of time buckets. With infinite persistence and envelope mode, they will eventually appear in a single pixel width and can then be expanded by using time windowing. Random repetitive sampling is a great advantage in this situation; a system based on sequential sampling would take a prohibitively long time to plot out such a low-repetition-rate signal.

Averaging. As lower-level signals are detected, broadband noise increasingly widens the trace of the amplified signal. Choosing a narrower bandwidth, if possible, cuts down on noise, but also slows down the system rise time. Averaging in the context of random repetitive sampling means averaging sample points associated with the same point in time with respect to the trigger event, but from different acquisitions. With increasing averaging, pulses

hidden in noise emerge and take shape. The digital averaging process filters noise like a low-pass filter with one important difference: it doesn't result in a rise time degradation.

Ratioing Channel Inputs. The Waveform Math menu allows the user to display the ratio of any two channel inputs, in addition to performing many other useful functions. For example, the ratio of the two sensor channels can conveniently show pulse compression when probing the input and output of a limiter, or the transfer function (mW/V) of a pulse modulator can be displayed as a ratio of a sensor channel to a video channel.

Amplitude@Time Markers. The HP 8990A provides not only amplitude and time markers, but also amplitude@time markers. These denote power (or voltage for channels 2 and 3), power difference, and power ratio for a start time and a stop time. The feature is useful in determining power or gain variations along a pulse, such as pulse droop.

Thirteen Measurements. Automatic level measurements on microwave pulses, including pulse peak, average, or pulse-top power, and time measurements on pulses, including rise time, pulse width, and duty cycle are implemented as simple blue-key shift functions. The reference levels for these measurements, that is, pulse top (100%) and pulse base (0%), are histogram-based according to IEEE standards.

Applications

The combination of two microwave sensors and two video inputs, able to make complex microwave and video pulse-measurements in one instrument and relate them in level and time, promises broad applications in many areas.

Radar Components and Systems. Starting out with a traditional area of peak power measurements, the characterization of the power transmitter is central to radar performance. Peak and average power, rise time, overshoot, and droop are standard measurements at the output of the transmitter. With the use of couplers, the dual microwave channels facilitate measurements of pulsed gain, gain compression, and pulsed return loss. Coupler losses can be compensated numerically in the HP 8990A.

On the receiver side, the receiver protection limiter needs to be characterized in terms of spike leakage and spike compression. Fig. 5 displays the output of a limiter. The time windowing feature is used to focus on the spike detail, and markers spell out the spike leakage in dB. The ratio of the input of the limiter to its output would show the power compression along the pulse.

At the system level, the peak power analyzer performs delay measurements between, for instance, pulse drive and transmitter output. The concurrent and time-calibrated display of the drive signal and the microwave pulse avoids cumbersome calibration tasks.

Analyzing the transfer characteristic of the pulse modulator is another example of the combined use of sensor and video inputs. The transfer characteristic can be displayed as a ratio of the microwave channel to the video channel and examined for linearity.

Complex Communication Signals. The powerful trigger capability of the HP 8990A comes into play when signals like the pulse bursts of a TDMA (time division multiple access) system are to be evaluated. For instance, trigger holdoff stabilizes the display by inhibiting recurrent triggering on subsequent edges. Trigger delay allows the user to select, say, the 231st pulse of a long pulse train. Trigger pattern lets the user trigger on specific pulses, like glitches, within a train or burst. The dual time base is useful for observing and measuring time or power from burst to burst and simultaneously on individual pulses within a burst (see Fig. 6).

Not so obvious is the application of the peak power analyzer to communication signals keyed in frequency or phase. The peak power analyzer can be used here to measure accurately the power glitches and changes that

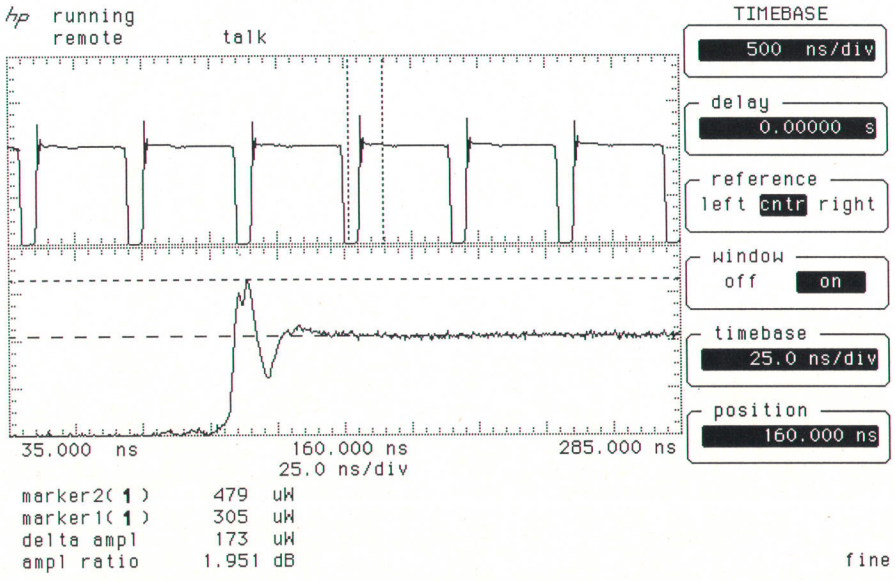


Fig. 5. Spike leakage measurement at the output of a radar receiver protector.

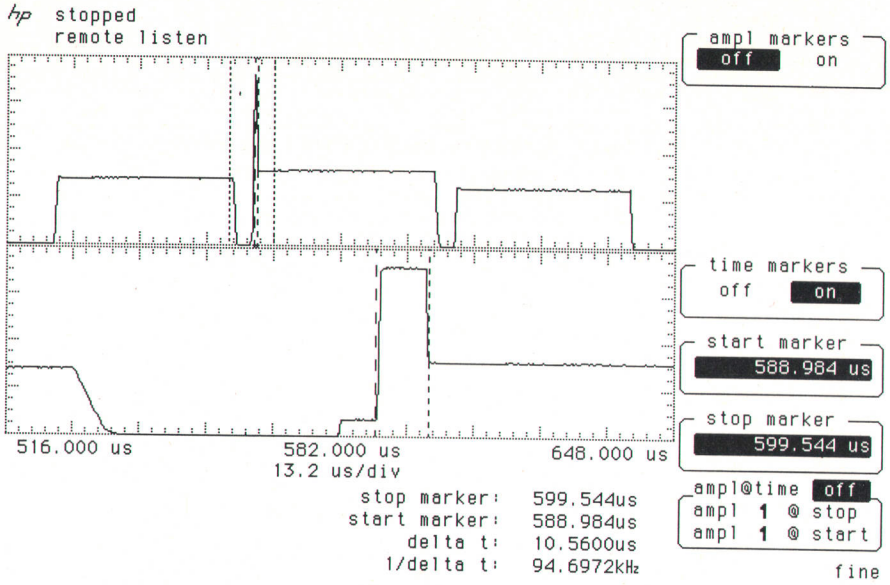


Fig. 6. GSM signal showing three frames of different levels representing maximum, minimum, and typical frame specifications.

accompany phase and frequency switching. More complex digital modulation formats, like the 16QAM signal of a digital radio, can be examined with regard to level transitions, overshoot, and power (amplitude) compression. Infinite persistence is useful for recording multiple traces (see Fig. 7).

Transient Response of Components. Transient measurements on microwave components usually require cumbersome calibration to account for control signal and instrumentation delays. The time calibration provided in the HP 8990A between the sensor channels and the video input channels and the simultaneous display of both the control stimulus and the microwave response make this task easy.

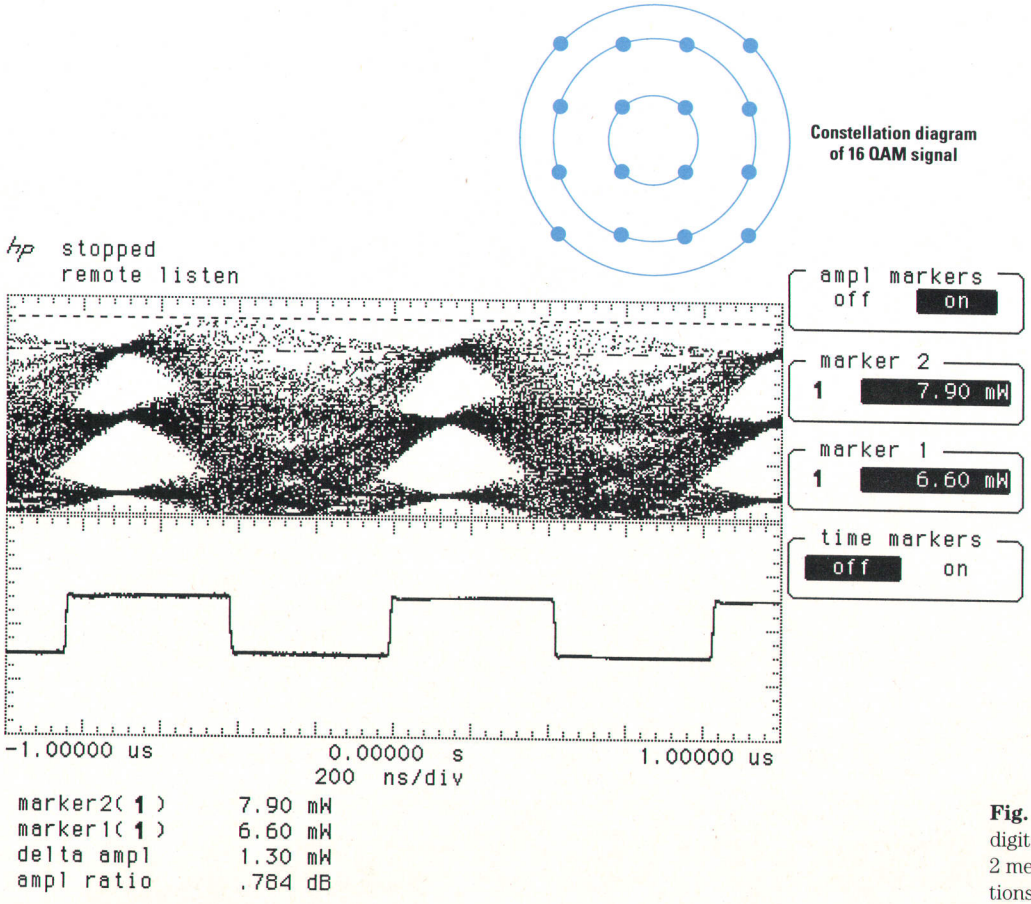


Fig. 7. Amplitude signature of a digital radio signal. Markers 1 and 2 measure the overshoot of transitions to the maximum amplitude.

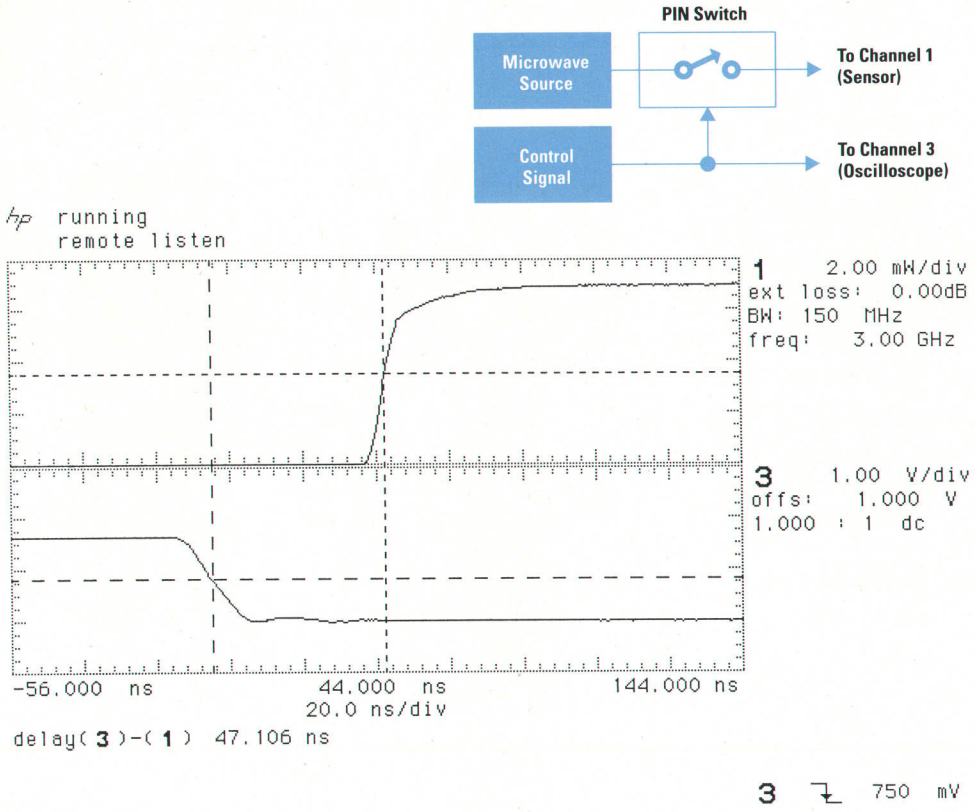


Fig. 8. Delay time measurement on a pin switch.

The pin switch in Fig. 8 is turned on by the falling edge of the control signal. A single blue-key operation measures the delay between the mesial points of the falling control signal (lower display) and the rising microwave pulse at the pin switch output. The wide pretrigger range makes it possible to view the full turn-on region of the transition.

Conclusion

The applications listed above give an indication of the versatility of the HP 8990A peak power analyzer. The integration of advanced sensor and calibration technology with the trigger and signal processing power of HP's advanced digital oscilloscopes results in an instrument that delivers carefree and accurate measurements on microwave pulses and driving control signals.

Acknowledgments

We would like to thank the HP Colorado Springs Division's HP 54500 team for their time and technical support. The enthusiastic collaboration of Helen Muterspaugh and her firmware team, and on the hardware side, Laura Whiteside and Yvonne Utzig, was key to the successful leveraging of digital oscilloscope technology and gave a powerful initial boost to the peak power analyzer development. Acknowledgments to the production engineering team, headed by Jay Anderson. Their efforts were critical for the fast-paced introduction of the project. Credit also goes to Darren McCarthy who was the driving force in reliability testing, to Allen Edwards for his technical contributions and his support as section manager, and to Jim Jensen who led the firmware effort.

References

1. D. Scherer, "Designing Sensors to Read Peak Power of Pulsed Waveforms," *Microwaves & RF*, February 1990.
2. R.A. Witte, "Understanding Digitizing Oscilloscopes," *RF Expo Digest*, 1989.

GaAs Technology in Sensor and Baseband Design

In the HP 8990A peak power analyzer design, the detector diodes for the sensors are GaAs planar doped barrier diodes, and the switches in the switchable-gain baseband amplifier use GaAs FETs.

by Michael C. Fischer, Michael J. Schoessow, and Peter Tong

The hardware design of the signal path of the HP 8990A peak power analyzer presented the conflicting requirements of wide bandwidth, high dynamic range, excellent dc performance, and a controllable gain range of more than 100 dB.

The signal path begins at the microwave detector, which is followed by a preamplifier, both residing in the sensor. The detected (baseband) signal then goes to a switchable-gain amplifier located within the analyzer. The stringent performance and reliability goals set for these circuits were met with the help of recent advances in gallium arsenide technology.

GaAs Balanced Detectors

Three different sensors are offered. They differ in their upper frequency limits of 18, 26.5, and 40 GHz, and correspondingly in their RF input connectors, which are type N, 3.5-mm, and 2.4-mm respectively. All three types use the same circuit configuration, but each is tested and calibrated only over its specified range.

The coaxial input connector interfaces with the circuit via a small cylindrical bellows that acts as a spring to accommodate mechanical tolerances between the input connector and the following sapphire substrate. The bellows also carries the input dc blocking capacitor. This capacitor sets a lower limit on the input frequency.

The coupling capacitor is pushed against the input end of a coplanar waveguide on the substrate by pressure from the bellows. A coplanar waveguide has the advantage of single-layer simplicity with convenient ground paths and also provides a good match to the coaxial input line because the field patterns are similar. The coplanar line includes a 3-dB pad to improve the input match at some cost in sensitivity. This pad drives the GaAs detector diodes and a 65-ohm shunt resistor to establish a net RF input impedance of 50 ohms for the detector circuit.

The MMIC (microwave monolithic integrated circuit) chip containing the detector diodes is bonded in a flip-chip manner onto the thin-film sapphire substrate to connect it to the coplanar structure. This integrated approach extends the upper frequency limit of the sensor to 40 GHz.

Gallium arsenide planar doped barrier diodes are used in the detector. The average saturation velocity of GaAs is much greater than that of silicon, and this allows the construction of diodes having the necessary very small junction capacitance with low junction series resistance. A low junction resistance lowers the RC time constant of the junction and raises the diode cutoff frequency. A planar doped barrier diode is less frequency-sensitive than a normal pn junction diode because of the intrinsic layer. In a pn junction diode the equivalent capacitance of the junction changes with power level, but in the planar doped barrier diode, junction capacitance is determined by the intrinsic layer, which remains almost constant as a function of power.

A specialized GaAs IC process allows custom tailoring of the doping to control the height of the Schottky barrier. This control makes it practical to operate the detector diodes in the current mode for all its advantages (see "Harmonic Errors and Average versus Peak Detection," page 94), while keeping the video resistance low enough to maintain high sensitivity.

The detector circuit employs two diodes, deposited symmetrically about the center of the coplanar line. They are driven in a push-pull manner to effect full-wave rectification. This scheme secures a number of advantages:

- Common-mode noise or interference riding on the ground plane is canceled at the detector output.
- Thermoelectric voltages resulting from the joining of dissimilar metals, a serious problem below –30 dBm, are canceled.
- Measurement errors caused by even-order harmonics in the input signal are suppressed.
- A 3-dB signal-to-noise improvement is realized by having two diodes. The output signal is doubled in voltage (quadrupled in power) while the noise output is doubled in power, since the dominant noise sources (in the input stages of the amplifiers) are uncorrelated.

Each diode is connected in series with a small integrated resistor at its output, followed by an RF bypass capacitor to ground. These resistors further improve the input match of the detector circuit. The impedance at the input side of the diodes consists of the 65-ohm resistor to ground in parallel with the loads presented by the two diodes. The junction resistances vary quite widely as a function of input power level, and the series resistors are there to reduce the effects of this variation. The choice of

resistance is a compromise between minimizing the impedance variation and reducing detector sensitivity.

The output of the detector circuit drives a transresistance amplifier (essentially a wideband current-to-voltage converter) with a low input resistance, forcing the diodes to operate in current mode rather than voltage mode. This prevents peak detection and ensures that the detector remains substantially average-responding, even at high power levels. Average-responding power detectors are significantly more accurate than peak detectors when the input RF carrier contains harmonics (see "Harmonic Errors and Average versus Peak Detection," page 94).

Diode saturation current and the thermal voltage kT/q are both strongly temperature dependent. This shifts the diode V-I curve and affects detection efficiency. A chip thermistor, mounted adjacent to the GaAs detector chip on the sapphire substrate, provides temperature data. The data is used to adjust firmware calibration coefficients automatically, thereby continually maintaining the analyzer's accuracy over temperature.

Thousands of hours of testing at maximum power and maximum operating temperature have confirmed the long-term stability of the detector sensitivity.

Sensor Video Amplifier

The amplifier circuit board in the sensor performs the initial processing of the rectified signals from the detector diodes. It also carries the EEPROM that contains the calibration data for that particular sensor. The sensor amplifier forms the interface for the balanced output of the detector diodes, provides chopping for dc stability at low levels, provides wideband gain from dc to over 300 MHz, and drives the 50-ohm cable to the analyzer.

The diode outputs must be loaded with a low impedance to keep the detection in a current mode rather than allow peak detection or voltage mode to occur. Signal-to-noise ratio is the limiting factor for low-level detection, so the amplifier also has to have low noise.

To provide submicrovolt-level dc stability along with hundreds of megahertz of bandwidth, the amplifier is constructed with separate paths, each optimized for its respective task (see Fig. 1). This requires that the initial circuit in the signal path be a diplexer to split the signals into dc/low-frequency and high-frequency paths.

Associated with the diplexer in the low-frequency path is the chopping stage, which disconnects the detector diodes

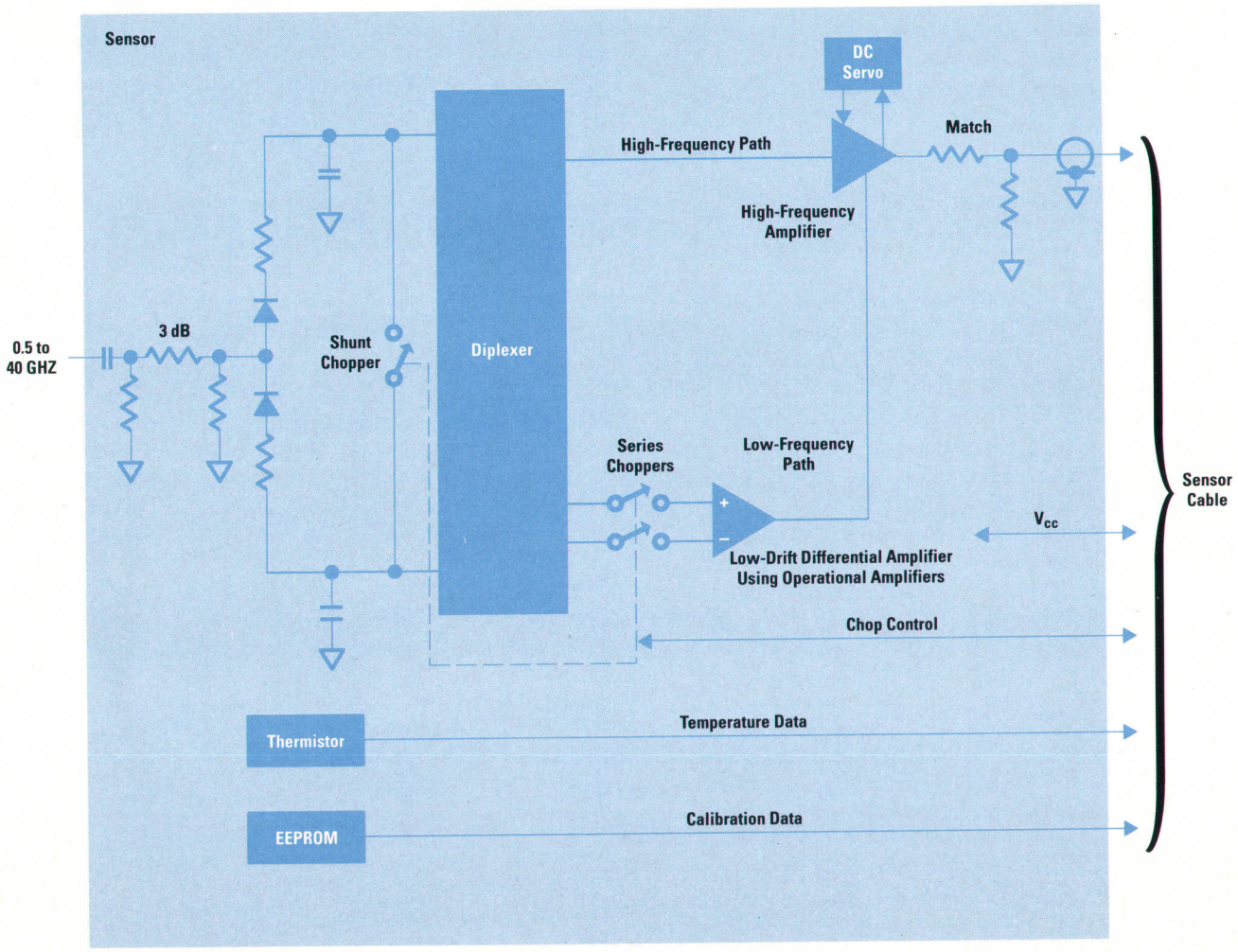


Fig. 1. Sensor amplifier design.

from the amplifier while the amplifier dc offset is being sensed and corrected using a combination of hardware and software techniques in the host analyzer. A bipolar junction transistor is used as a shunt chopper. It nearly shorts together the two opposite-polarity signals from the two diodes. The bipolar junction transistor offers an on resistance of very few ohms and an off capacitance of only a few tens of picofarads, both important for this function. This shunt chopper is driven through a pair of junction FETs, which provide the turn-on current for the chopper, and have low leakage and offsets when the chopper is turned off.

The most important part of the chopping is performed by the next devices on the signal paths, a pair of junction FETs in series with the signals from the detector diodes. These FETs have on resistances of only a few ohms to preserve the low input resistance of the amplifier. When these FETs are turned off, the low-frequency amplifier circuits are open-circuited, allowing their offset voltages and any other offsets in the entire active signal processing chain to be zeroed automatically.

After the chopper stage in the low-frequency path come the transresistance amplifiers. These are operational amplifiers, each having a feedback resistor that converts an incoming current to a corresponding output voltage. A pair of these amplifiers processes the balanced signals. These amplifiers are referenced to the diode and RF connector shell ground for frequencies below 10 Hz, and to the ground plane on the circuit board for higher frequencies. This separation of grounds allows the management of ground currents that might otherwise superimpose dc errors on the signal to be measured. One typical source of such interference is the potential difference between the chassis of various pieces of electronic equipment.

The outputs of the transresistance amplifiers are next combined differentially in the following stage by an operational amplifier with balanced inputs and a single-ended output. This single-ended signal is then recombined with the high-frequency component of the rectified envelope.

The recombined signals are fed to the high-frequency amplifier, a silicon bipolar junction transistor connected with collector-to-base feedback resistance to make it function as another transresistance amplifier with low input resistance. This fast amplifier must also handle the dc component of the signal. To stabilize it, an operational amplifier is connected to compare the input current with the output voltage of the stage and drive the base to force correspondence, with accuracy determined by the dc accuracy of the operational amplifier. The feedback in the fast amplifier stage lowers its output impedance significantly, so a series resistance is inserted to back-terminate the coaxial cable to the instrument properly. Combined with this back termination is a shunt resistance. Together, these two resistors standardize the overall sensitivity of each sensor so that it closely matches all others.

Several adjustments allow compensation for component tolerances to yield a very flat time-domain pulse response.

Postsensor Gain Requirements

The signal voltage level at the output of the sensor ranges from roughly 2 μV to 300 mV, corresponding to an RF input range of 100 nW to 150 mW.

An input power range of 62 dB results in an output voltage range of 104 dB. Most of this "range expansion" occurs at power levels below −15 dBm, where the detector diodes operate in the square-law region. Here a 1-dB input power change results in a 2-dB output voltage change. This effect substantially increases the gain requirements of the succeeding circuits.

Baseband Board Circuitry

The video processing circuitry of the HP 8990A requires at least 50 mV of signal, so considerable amplification of the sensor output is necessary in most cases before the signal can be processed. The baseband amplifier provides this amplification, along with a variable-bandwidth capability to optimize the noise performance.

The baseband board also contains additional circuitry to facilitate gain calibration and offset correction.

Fig. 2 shows a functional block diagram of the baseband board. Five separate low-pass amplifier blocks can be switched in, cumulatively, as more gain is required. The first three blocks use off-the-shelf operational amplifiers, while the last two are discrete wideband amplifiers with dc servos to ensure low offset.

When no amplifiers are switched in, there is typically 12 dB of loss along the signal path because of losses in the switches. Baseband gain can be set to six different values from −12 dB to +82 dB. Finer control of the overall gain is provided downstream in the video processing circuitry.

In any high-gain amplifier chain, circuit noise is an issue that must be addressed. In this case the noise originates primarily in the sensor amplifier, and if unchecked, will saturate the video cicuitry at the higher gain settings. The standard method for controlling noise is bandwidth limitation, and that is part of the solution here, along with digital processing of the video signal. There are always trade-offs, however, and in this case the trade-off is between rise time and dynamic range. At low sensitivity settings (for example 500 μW/div) noise is not a problem and the maximum bandwidth of 150 MHz is available, while no noise is visible on the screen. However, as sensitivity is increased a band of noise begins to appear about the trace, limiting the precision of the display.

Digital signal processing allows the user to reduce the displayed noise through the use of trace averaging for repetitive waveforms. This is a powerful tool, but it does have limitations. First, the waveform no longer appears in real time because a certain number of averages of the

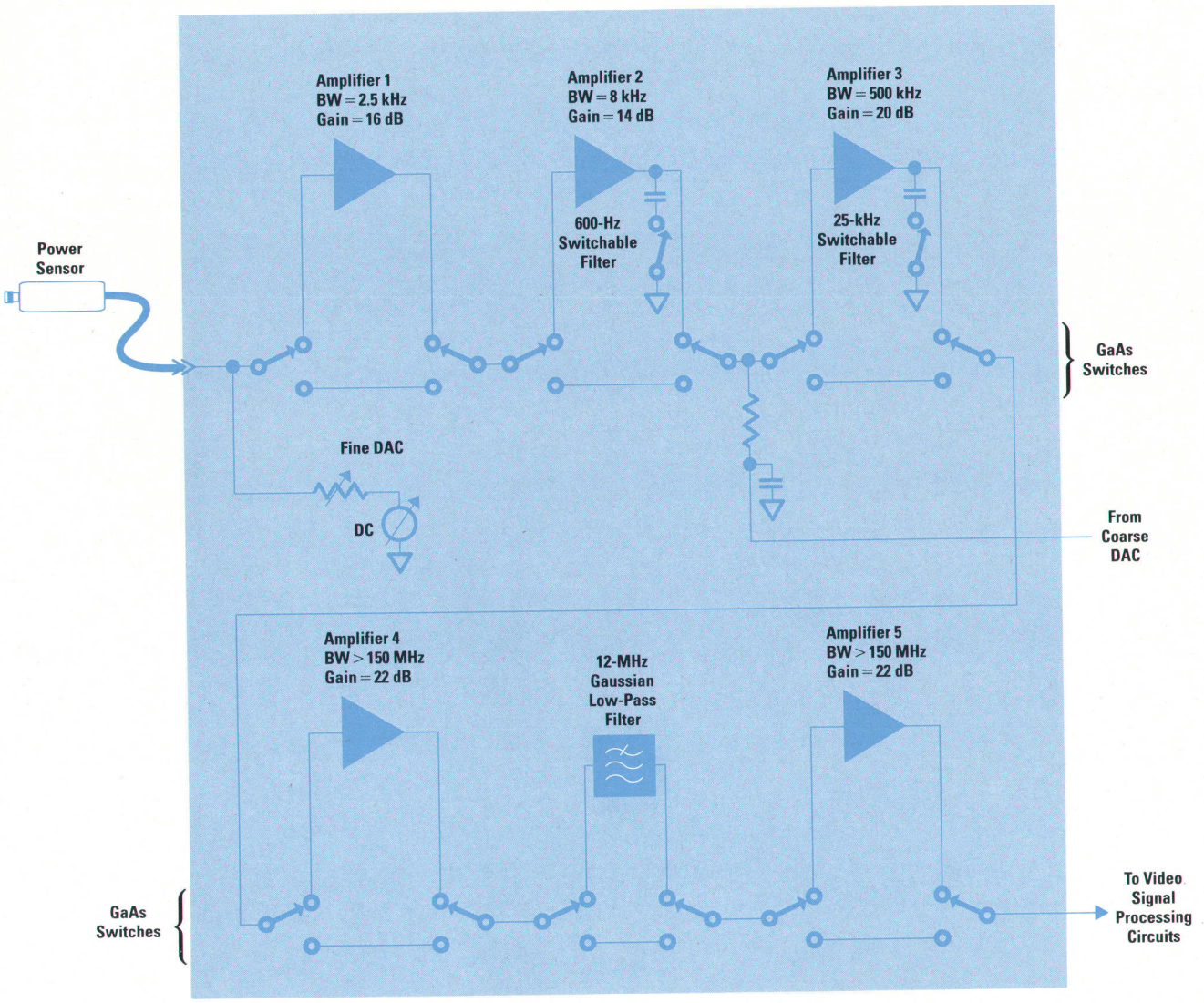


Fig. 2. Functional block diagram of the baseband board.

same waveform must accumulate before the (uncorrelated) noise is canceled out. Second, the digital processing captures only signals appearing on the display. Noise peaks beyond the boundaries of the display are ignored during averaging and this can lead to an offset error in the averaged display for cases where the noise is excessive or where the trace comes close to the top of the display in the presence of noise.

The limitations of digital trace averaging don't apply to the bandwidth limitation method of noise control. However, there is an adverse effect on rise time that always comes with lower bandwidth.

In the HP 8990A a judicious application of both techniques provides optimum performance over a wide range of test and measurement applications.

As the baseband board gain is increased by switching in more amplifier blocks, the bandwidth is reduced in steps to limit noise to a moderate level while not sacrificing rise time too much. The bandwidth is set by the amplifiers themselves or in some cases by switchable filters, as shown in Fig. 2. The use of switchable filters, plus 30 dB of fine gain control downstream in the video circuits, provides flexibility in the setting of baseband gain and bandwidth. This flexibility allows the user to select the bandwidth best suiting a particular application via a **Low/Auto/High** switch. For example, a rise time of 5 ns is available at 5 μW/div with the bandwidth set to **High**, but the dynamic range is limited by the high noise level, which must be reduced through averaging. The dynamic range reduction results from the requirement that the noise must be contained within the borders of the display before averaging, as explained above. With the bandwidth set to **Low**, a negligible noise level is achieved at the same sensitivity without the use of averaging, but at the expense of a 25-μs rise time. The (default) **Auto** bandwidth setting falls between these two cases.

A critical consideration in the design of the baseband board was the performance of the switches used to select the amplifiers. The performance requirements include low on resistance, low capacitance, small size, low power dissipation (low heat generation), and good dc accuracy. The requirement for simultaneous low resistance and low

Harmonic Errors and Average versus Peak Detection

Envelope detectors can be operated in a voltage mode by loading them with a high impedance, or in a current mode by loading them with a low impedance. In the voltage mode, at high signal levels, detectors perform peak detection, that is, their output response is proportional to the peak voltage amplitude of the input waveform. Historically, most microwave envelope detectors have been designed to operate as peak-responding detectors, often to maximize their voltage sensitivity.

Current-mode detectors (as in the HP 84812/13/14A sensors) keep the diodes in conduction over most of each half-cycle and this results in a detector that is substantially average-responding, even at the highest input power levels.

Detectors that operate in the peak mode in their linear range suffer from the transition between square-law and linear response, and from the change from an average response in the square-law region to a peak response in the linear region. Peak-mode detectors also have much slower rise and fall times than average-mode detectors, and the peak-responding detector incurs the full error penalty from harmonic content, an error that is only partially improved by balanced full-wave detection with differential amplification.

The mode of operation (average or peak) of a detector is established by controlling the ratio of diode video resistance to video load resistance, taking into consideration the RF bypass capacitance and any other reactive components of the video load seen by the diode.

At low input power levels (< − 10 dBm), both voltage-mode and current-mode detectors are in their square-law region. A detector in its square-law region is inherently average-responding. This square-law average response combines the energy in the fundamental and any harmonic components present to give a true total average power result. However, at high signal levels, a voltage-mode detector responds to the instantaneous peak of the waveform, the vector sum of the components. The following table shows the errors caused by harmonic content for different detection methods

Worst-Case Error Caused by −20 dBc Harmonic Content

Detector Type	Second Harmonic	Third Harmonic
Peak-Responding (Most Detectors)	21%	21%
Peak-Responding with Balance	7%	21%
Average-Responding	3%	7%
Average-Responding with Balance (HP 8990A)	1%	7%

A signal having a second-harmonic content of − 20 dBc can suffer a 21% error in a peak-mode detector, while the average-mode detector can reduce this to 3%, even at high power levels in the linear region. Balanced average detection can further reduce this error to 1%.

Third-harmonic content can cause the same magnitude of error in a peak detector, while average detection gives a factor of 3 improvement. Balancing gives no improvement for odd harmonics.

Michael C. Fischer
Development Engineer
Stanford Park Division

capacitance ruled out the use of JFETs and the dc accuracy requirement eliminated the possibility of diode switching. The choice finally was between mechanical relays and GaAs FET switches. Monolithic GaAs switches were chosen for their advantages in size and power consumption. There was also concern about the reliability of mechanical relays in some ATE applications in which the switches could see more than a million cycles in a relatively short period of time.

The on resistance of the switches is low but not negligible—about four ohms. With twelve such switches in series, the impedance of the signal path could deviate significantly from 50 ohms, depending upon the switch settings, and this would cause gain and frequency response errors leading to poor pulse response. The problem was prevented by placing resistors or RL networks between ground and each switch pole to equalize the impedance along the signal path in a distributed manner.

Two digital-to-analog converters (DACs) feed into the baseband signal path, as shown in Fig. 2. The fine DAC at the input of the board can be programmed to output specific dc voltages or currents with very high accuracy. This DAC is used to measure the dc gain and input impedance of the board during vertical calibration. The coarse DAC is used primarily to adjust the dc level of the signal path during zeroing.

The very high gain of the baseband board and subsequent circuits makes interference from electric and magnetic fields a major design consideration. This is especially true with a switching power supply and a CRT display residing in the same box. Mu-metal shield structures are employed around the board and around the CRT components (see "Multilayer Shielding Protects Microvolt Signals in High-Interference Environment," page 84). In addition, the printed circuit layout of the amplifier chain and the routing of input cables are configured to keep signal loop areas small. The input cables pass through common-mode chokes that reduce ground-loop currents at the 25-kHz CRT sweep frequency. These extreme measures ensure that interfering signal levels at the input to the baseband board remain below 100 nV peak to peak.

Acknowledgments

The authors would like to acknowledge the contributions of Russ Riley and Lee Colby for sensor development, Bill Strasser for the initial baseband design, and Sandy Dey, Rusty Myers, and Marc Tognaccini in solving the many intricate problems of assembling, testing, and establishing the calibration of these devices.

Automatic Calibration for Easy and Accurate Power Measurements

Changes in input power, carrier frequency, and sensor temperature are automatically compensated for. The user is not required to disconnect the sensor from the device under test and connect it to a calibration source.

by David L. Barnard, Henry Black, and James A. Thalmann

The HP 8990A peak power analyzer is designed to measure the power of pulsed signals accurately over a wide dynamic range. A well-designed calibration strategy was required to achieve the specified accuracy over all of the specified operating conditions. HP 8990A calibration includes both calibration of the power sensor and calibration of the analyzer.

Calibration of the Sensor

The sensitivity of the sensor is strongly influenced by the operating conditions. Calibration of the analyzer and sensor to compensate for changes in sensor sensitivity over these varying conditions became a significant design issue. There are three parameters that affect the sensitivity of the sensor: the incident power level, the carrier frequency of the applied signal, and the operating temperature of the sensor diode (see Fig. 1). These needed to be carefully considered in the development of the calibration scheme.

Besides the accuracy requirement, other goals affected the selection of a calibration strategy. One of these was ease of use. We wanted to minimize the effort on the part of the user to perform a calibration. For example, we didn't want to require the user to disconnect the sensor from the device under test and connect it to a calibration source. This would be especially objectionable if frequent calibrations were required. Another goal was to avoid the need for a lot of specialized calibration hardware that would add significantly to the cost of the analyzer. We didn't want the analyzer to perform excessive computation during normal use, which could make operation slow.

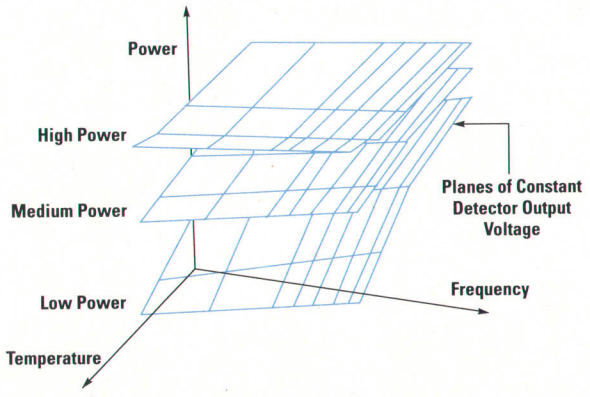


Fig. 1. Power sensor sensitivity is affected by the incident power level, the carrier frequency of the applied signal, and the temperature of the sensor diode. Automatic calibration of the HP 8990A peak power analyzer compensates for changes in these parameters.

Sensor Characteristics

The sensitivity of the sensor diode is a nonlinear function of the incident power. At low power levels, the diode response has a square-law characteristic, so the voltage appearing at the output of the sensor is proportional to the power of the applied signal. At high power levels, the response approaches linear operation. In this operating region, the sensor's output voltage is approximately proportional to the signal voltage. A broad transition region starts to appear above 10 μW and is still evident in the high-power region, so linear operation is never fully achieved. For this reason, no simple mathematical model exists to describe the voltage-versus-power transfer function of the sensor over the specified power range.

The sensor is required to operate over a broad frequency range, which brings into play the frequency dependent characteristics of the diode. The dominant effect is a roll-off of sensitivity that occurs at higher carrier frequencies. This roll-off is strongly dependent on the incident power level. In addition, there are some minor ripples in sensitivity starting near the middle of the frequency range. These frequency dependencies are much more noticeable at low incident powers (see Fig. 2).

The sensitivity of the sensor shows a significant temperature dependency. The sensitivity changes quickly at low temperatures and flattens out at high temperatures. Again, this effect is most noticeable at low incident powers (see Fig. 3).

Calibration Alternatives

We concluded that each sensor would have to be calibrated over its specified power range at a given operating point of carrier frequency and temperature. No way was found to model the physical processes in the sensor diode with sufficient accuracy to reduce this requirement. Several calibration alternatives were considered.

One solution would be to use a calibration source capable of producing accurate power levels over the power and frequency range of the sensor. The sensor could then

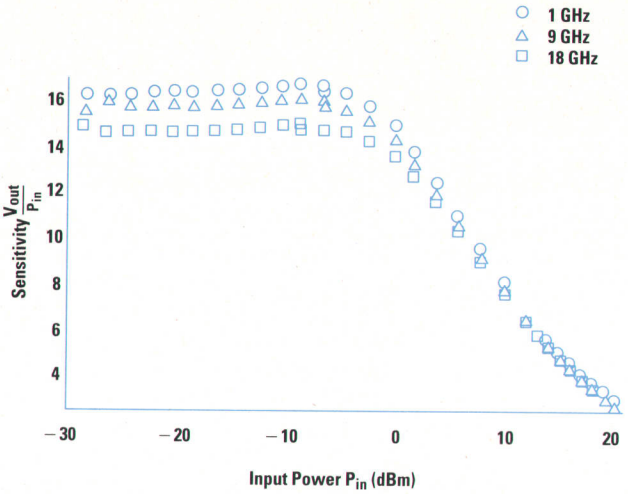


Fig. 2. Sensor sensitivity as a function of power for three carrier frequencies.

be calibrated before use, regardless of its current operating temperature. However, such a calibration source would be prohibitively expensive. Also, this approach would require the user to disconnect the sensor from the source under test to perform the calibration, and the calibration would have to be repeated whenever the operating temperature of the sensor changed.

A more economical solution would be to have a single-frequency calibration source. As before, the calibration would be performed over the sensor's input power range. Frequency calibration data might be stored in the sensor to correct for the change in sensitivity over frequency. This approach would be less accurate, since the frequency response of the sensor diode is not independent of power and temperature. It would also suffer from the need for manual intervention by the user.

HP 8990A Approach

The disadvantages of these calibration approaches led us to consider a dramatically different approach—characterizing the sensor over temperature, frequency, and power, and storing this information in the sensor for use by the analyzer. This scheme relies on the long-term stability of the sensor diode technology, which was shown to be excellent in the course of extensive reliability testing.

To make use of the data characterizing the sensor, the analyzer must know the operating conditions of the sensor. The operating power is not a problem since the analyzer always knows what range it is set to. Since the analyzer has no way of determining the carrier frequency, the frequency must be specified by the user, but this would also be true for any of the previously mentioned calibration schemes. The one new item of information required by this scheme is the sensor's operating temperature.

To support this calibration scheme, the sensor is designed with a thermistor located in close proximity to the sensor diode. The analyzer can read the thermistor with an analog-to-digital converter to learn the operating temperature. This, along with the carrier frequency supplied by the user, gives the analyzer sufficient information to interpret the sensor data to perform a calibrated power measurement.

Implementation

To implement the selected calibration scheme, we needed to find an efficient way to represent the sensor's behavior. As previously mentioned, we concluded that the sensor's nonlinear relationship of input power to output voltage precluded the use of a simple mathematical model. This led us to test each sensor over the specified power range. Since the effects of temperature and frequency influence each other and are not easily modeled, it became apparent that the power-to-voltage transfer function of each sensor would have to be measured at a number of different temperatures and frequencies.

A sensor test system was designed to perform the needed measurements. It measures the sensor output voltage as a function of power over the specified range of the sensor. The sources are broadband, allowing any test frequency in the specified range of the sensor to be used. The sensor under test is placed in a temperature chamber so it can be characterized over temperature.

A grid of temperatures and frequencies is constructed for each sensor model. For each temperature, power-versus-voltage data is collected at each test frequency. After correction for mismatch errors, the resulting test data forms a three-dimensional matrix of power and voltage pairs.

Processing the Sensor Test Results

The matrix of measurements delivered by the sensor test system is too bulky to store directly in the sensor. We decided to try processing the data with numerical curve-fitting techniques to yield a more compact representation. We were concerned that the representation be usable by the analyzer without a lot of time-consuming floating-point processing. Such processing could cause large delays whenever the analyzer recomputed the sensor's response.

This ruled out the use of logarithmic or exponential functions. Instead, we chose polynomials. To cover the large dynamic range, we broke the power-to-voltage curve into four segments. For each segment, a curve is fit to

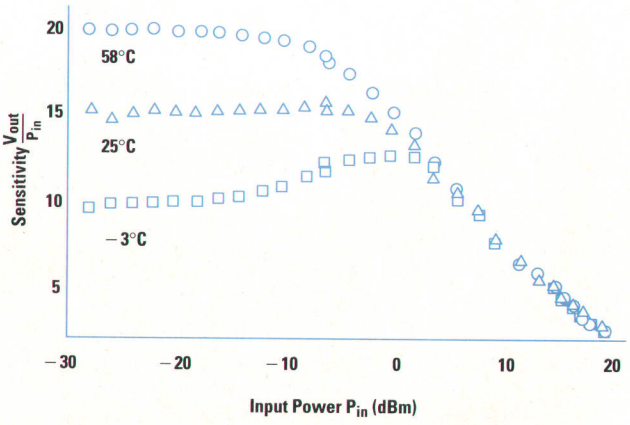


Fig. 3. Sensor sensitivity as a function of power for three temperatures.

the data by the least squares method with the added constraint of making the endpoints match the adjacent segments. The resulting polynomial coefficients are much more compact than the original data, making storage in the sensor's internal EEPROM practical. Also, polynomials can be efficiently calculated in the analyzer.

Analyzer Calculations

The analyzer reads the coefficients and associated data from the sensor's EEPROM at power-up or when the sensor is plugged in. The analyzer then uses the coefficients to calculate the polynomials, which give power as a function of sensor output voltage. Each set of coefficients, and therefore each polynomial, applies at a single point in a two-dimensional grid of temperature and frequency. Thus, the power at a given operating point is calculated by interpolating between the powers at the nearest test frequencies and temperatures. This interpolation is implemented by a spline surface-fitting algorithm[1] and is included as part of the overall voltage-to-power function.

The voltage-to-power function is applied in two ways. The first step is calculating the amount of internal gain that the analyzer must insert to amplify the detected voltage to match the selected full-scale power sensitivity. This requires the inverse function, power to voltage, which is calculated from the original function by a version of Newton's method.[2] Once the gain is set, we are assured that applying the power corresponding to the top of the screen to the sensor will result in the amplified output voltage corresponding to the top quantization level of the analog-to-digital converter (ADC). Independently, the offset leveling, which is automatically performed, ensures that the lowest ADC reading corresponds to zero power.

The final step is to calibrate the rest of the ADC levels. A table maintained inside the analyzer translates the ADC levels to calibrated power. The voltage-to-power function is used to calculate the values in this table.

This process must be repeated whenever a new analyzer sensitivity is selected, when a new carrier frequency is entered, when the sensor temperature changes, or when a new sensor is plugged in.

The analyzer continually monitors the sensor thermistor to check for temperature changes. If the temperature deviates more than a certain amount, the calibration procedure is automatically performed. This relieves the user from the worry of manually recalibrating the analyzer when the sensor's operating environment changes.

Calibration of the Analyzer

The calibration of the analyzer is complicated by a number of constraints. These include circuit nonlinearities, the large dynamic range, and the nature of the signal in the HP 8990A. A systems design approach was required, including both hardware and software design.

Offset Voltage

Even ignoring the effects of the GaAs switches in the HP 8990A signal path, offset voltage is a design issue, since the amplifiers in the sensor produce an offset that depends upon ambient temperature and other factors and so tends to drift slowly in operation. The HP 84810 Series sensors incorporate a "chop" line, which commands the sensor circuitry to simulate a condition of no incident RF power. This permits offset leveling without operator intervention. The offset is periodically releveled to prevent any drift out of the specified accuracy over time. It doesn't matter where in the signal processing path the offset variation originates; a single swift automatic leveling effectively compensates for the offset.

Channel Resistance

The channel resistance of the GaAs switches is known to vary with operating temperature, so it is important to be able to compensate for the effects of channel resistance variation during analyzer self-calibration. One reason channel resistance effects are so important is that they affect the impedance match between amplifier stages and so influence the overall gain of the analyzer. Consequently, the voltage gain is measured during self-calibration.

Another effect of channel resistance variation is that it directly affects the (nominally 50-ohm) input resistance of the HP 8990A. The sensors are calibrated in terms of their output to a load of exactly 50 ohms. During self-calibration the HP 8990A measures its own input resistance to determine the match between the sensor and the analyzer.

Offset DAC Circuits

In a sense, the HP 8990A self-calibration subsystems are built around the precision fine-DAC circuit, which is constructed of highly stable precision components. The circuit can operate as either a low-impedance source (voltage mode) or a precise medium-impedance source. The HP 8990A also has a coarse DAC, which is not a precision DAC. The coarse DAC injects into the signal path downstream of some amplification or attenuation from the point of injection of the precision fine DAC (see Fig. 4).

Automatic Offset Leveling

To level the offset in the general case, the analyzer is first set up as follows. With the sensor chop line "pulled" and the precision fine DAC set to its starting position in medium-impedance mode, the stick DAC, which provides the reference voltage to the flash ADC (see Fig. 4), and the coarse offset DAC are set to midrange. Then the flash ADC is used to take many data samples. Typically the ADC output is at its upper or lower limit at this time since the coarse DAC is probably in the wrong position. A conventional binary chop search is made for the coarse DAC setting that will bring the signal level within the range of the ADC. Then, because the fine DAC is extremely linear, an extrapolation can be done using two pairs of fine DAC settings and ADC readings to calculate the fine DAC setting that just gives an ADC reading of zero. This completes offset leveling for the simple case.

The coarse DAC has a relatively long settling time, so when offset leveling is performed, care is taken to avoid changing the coarse DAC setting unless absolutely necessary. Since different vertical ranges typically require different coarse DAC settings, changing the range can result in a delay because of the binary search needed to

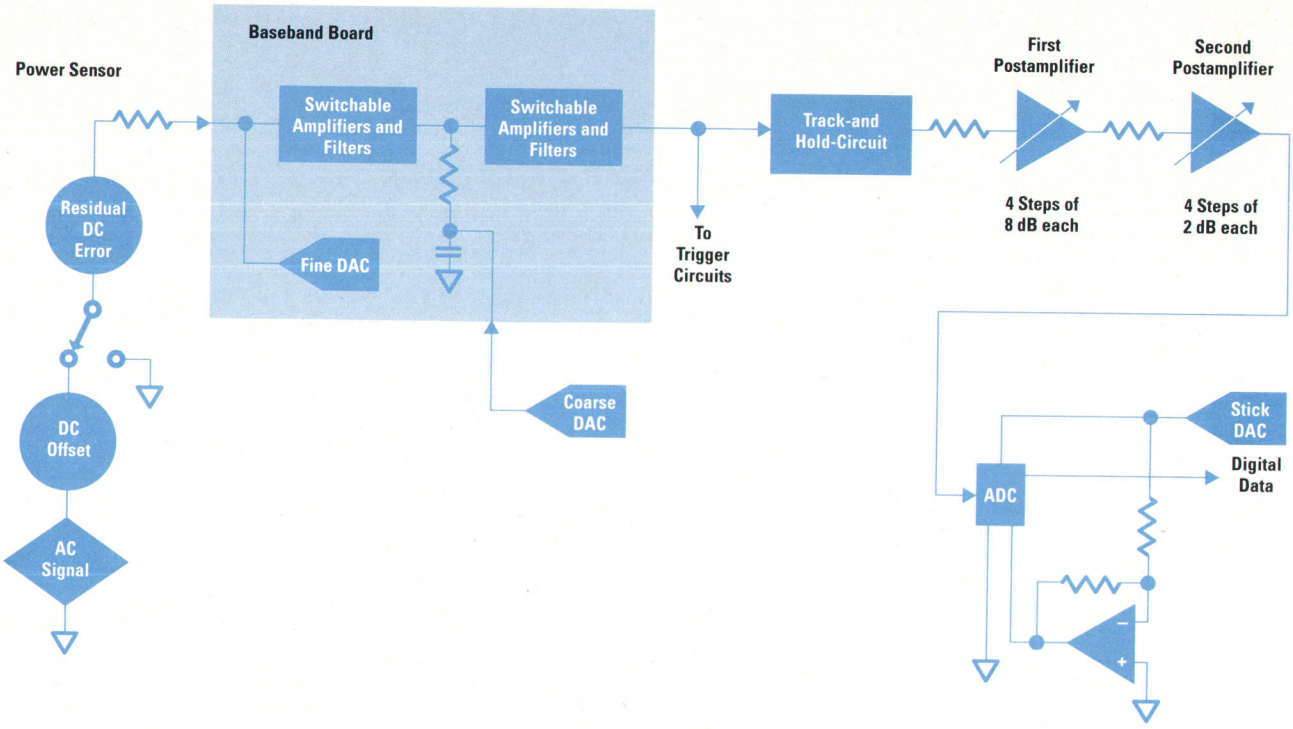


Fig. 4. Analog model of the HP 8990A used in calibration.

find the new coarse DAC setting. To avoid this delay, once offset leveling is performed in a particular vertical range, the coarse DAC setting for that range is stored. When that range is revisited, which might happen after the user makes a series of range changes, the stored coarse DAC setting is used. This eliminates the need for the time-consuming binary search except when the offset voltages have changed significantly.

A manual zero feature analogous to that of Hewlett-Packard average power meters is provided. This feature can be time-consuming to use but is available for best accuracy when the signal is below −30 dBm. It is capable of correcting for offsets that precede the chop switch.

Vertical Calibration

Essentially, vertical calibration of the HP 8990A answers the question: How much signal from the sensor corresponds to full scale at the ADC?

Within its operating range, the sensitivity of the HP 8990A is, for practical purposes, continuously adjustable. Thus the purpose of vertical calibration is to provide data so the vertical setup subsystem can select the analyzer hardware settings that will provide the desired sensitivity. The vertical calibration data includes:

- Vertical sensitivity expressed in ADC counts per volt.
- Input resistance expressed in ohms.
- Compensation coefficients for correcting analyzer nonlinearity.

This data is repeated for each combination of vertical amplifier settings.

The actual measurement of vertical sensitivity is relatively simple: the precision fine DAC is placed in its low-impedance mode (also known as voltage-source mode) and its count is changed. The ratio of the change in the fine-DAC count to the corresponding change in the ADC count, together with the absolute sensitivity value k of the precision fine DAC, provides the needed value:

$$\text{Sensitivity} = \frac{\Delta(\text{fine DAC})}{\Delta(\text{ADC})}\,k\,.$$

Mathematical techniques such as linear regression and drift modeling are used to minimize execution time and maximize accuracy.

Input resistance can be measured with or without a sensor attached. The input resistance calibration is performed using the precision fine DAC as a stimulus. If a sensor is connected to the input of the analyzer, the sensor output impedance forms a load in parallel with the

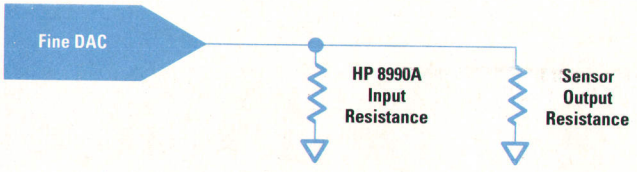


Fig. 5. Input resistance calibration equivalent circuit.

Testing the Peak Power Analyzer Firmware

The firmware quality assurance plan for the HP 8990A firmware had the following objectives:

- Extensively test all HP 8990A functionality to meet shipment criteria
- Develop a comprehensive automated test procedure to simplify verification of development firmware revisions and postshipment firmware releases
- Leverage as many test tools as possible.

An internally developed tool called the HP-IB Interactive Test System (HITS) was leveraged from previous projects and used for automated testing. The program was modified to add various new features and commands to test HP 8990A functionality. HITS is a BASIC program that runs in the RMB-UX environment on HP 9000 Series 300 workstations. It takes in HITS input test files and creates corresponding signature files. The HP-UX **diff** utility is used to compare the output signature files against verified reference files. Any discrepancies point to a change with the new firmware revision which should be investigated as a potential problem.

The HITS test files contain a series of commands, usually one per line. The command format is as follows:

```
CC XXXX...X YYYYYY...Y
```

The CC field contains a two-character command that specifies how to interpret the other two fields. For example:

```
CQ CHANNEL1:RANGE 10mw
```

The **CQ** tells HITS to send the command **CHANNEL1:RANGE** 10mw and then send the query **CHANNEL1:RANGE?**. The command, query, response, any error messages, and status information are logged to the signature file.

HITS provides the following capabilities:

- Send a command
- Send a query and get a response
- Send a command followed by a query
- Perform a measurement and limit-check the result
- Perform random key testing for a specified number of key presses
- Randomly set parameters in a specified range a specified number of times
- Perform a series of tests in sequential order
- Repeat a series of tests in random order a specified number of times
- Test IEEE 488.2 functionality
- Test digitization functionality of the HP 8990A
- Test autoscale functionality of the HP 8990A
- Interact with the DUT to facilitate development of test cases and other functions
- Log commands and queries to a file (no responses, errors, or status messages are logged).

All the commands and the resulting queries, responses, error messages, and status information are logged to signature files.

The HP 8990A is a fairly complex analyzer with over two hundred and fifty functions. To test this large set of features, both subsystem and scenario-based testing were used. As a first step, subsystem tests were written to verify the basic functionality of each subsystem. Then scenario tests were written to test various measurement scenarios and the interactions and couplings between the various functions. About 80% of the automated test development time was spent generating the scenario tests.

Time was also devoted to manual testing of front-panel operation, the display subsystem, calibration scenarios, interaction with different controller platforms, analyzer options, and features that cannot be tested in an automated fashion. In addition, all code was run through the C program checker/verifier tool lint and the C syntax checker tool **inspect**.*

In addition to HITS, a second BASIC program called INTERP was used for normal interaction with the analyzer and for testing new functionality. The INTERP program was written by the project's HP-IB engineer as a development tool. This program takes an HP-IB command, sends it to the analyzer, and automatically reads the responses to queries. It prints error messages and status information, and provides support for reading and sending block data and for recalling previous commands. The program can also read commands from a file. This feature was used to recreate problems, with input provided by the file created by the HITS log feature.

The automated tests proved to be very useful. After changes to the firmware a successful test run helped us verify that a bug fix had not introduced any new problems. When bugs were found that were not detected by the automated tests, the tests were updated to check for those specific problems. This improved the coverage of the automated tests. The automated tests were also run to verify that hardware changes had not had any unexpected impact on firmware functionality.

The firmware shipment goal for the HP 8990A peak power analyzer was to reach a defect rate of <0.05 defects/hour of test time. To stop testing, the defect rate had to exhibit a trend of <0.05 defects/hour and the product had to go through a period of 40 hours of test time without discovering any defects.

During the final phase of quality assurance testing an automated/manual test cycle was used. Once the firmware passed the automated tests the product was released to a group of marketing and R&D engineers. This group performed application-specific and function-specific testing for a period of 24 hours. Any defects found were fixed and the cycle was repeated until the shipment criteria were met.

The HP 8990A has a total of 99,351 lines of noncomment source statements (NCSS). Since this was a leveraged product, a better metric is the number of lines of code that were modified or added. Approximately 40,000 NCSS were added or modified. During the product development cycle, 355 defects were logged. This gives a defect rate of 8.88 defects per thousand lines of code.

The testing process worked well in helping the HP 8990A firmware team meet all of its quality assurance objectives.

Acknowledgments

I would like to thank Tatsuo Yano and Mark Johnston for their help with HITS, Jim Thalmann for writing INTERP, and the Colorado Springs Division's HP 54500 team for their help in resolving various defects.

Jayesh K. Shah
Development Engineer
Stanford Park Division

* **inspect** is a trademark of AT&T.

analyzer input resistance (see Fig. 5). In essence the resistance calibration consists of finding two precision fine-DAC stimuli, one in low-impedance mode and one in medium-impedance mode, that produce the same effects at the ADC converter. The Thevenin equivalent source resistance of the sensor is available from the sensor's own EEPROM coefficients. The sensor's parallel conductance is corrected for mathematically.

The compensation coefficients correct for small changes in analyzer sensitivity that occur if the calibration hardware setup is not the same as the measurement hardware setup. One such coefficient corrects the input resistance for the difference in voltage gain between calibration and measurement. Another coefficient accounts for the effects of offset voltage feedback. The error component is only a small part of the gain, so a very simple model of it is good enough.

Vertical Setup

To see how the various calibration factors interact, it is illustrative to look at what happens when a new vertical sensitivity is selected. In part the process involves making some apparently arbitrary decisions at the outset to set an overall gain or sensitivity. The process then converges to a solution. It begins when the user selects a desired sensitivity in microwave power level per screen division. This determines the full-scale power level. From this level, the sensor data, the temperature, and the carrier frequency, a full-scale Thevenin equivalent circuit of the source (open-circuit voltage and source resistance) can be determined. Making an approximation for analyzer input resistance, the full-scale power level is converted to an input voltage at the analyzer front panel and the amount of baseband board amplification required to produce a roughly correct signal level at the flash ADC converter is selected. Recall that the baseband board amplification is available in steps of approximately 20 dB. Once the baseband board amplifiers and appropriate low-pass filters are selected, the input resistance and the full-scale input voltage can be determined accurately. Since the full-scale input voltage is known, the lookup table (which expresses the nonlinear relationship between power and voltage) can be constructed. The postamplifier settings can also be selected. Having converged this far, the setup is within about ±1 dB of the desired sensitivity. After a final iteration of the calculation, the final value of the stick DAC setting is deduced so as to arrive at precisely the desired sensitivity. The stick DAC provides the reference voltage to the flash ADC converter and is the final adjustment of sensitivity. Offset leveling is then all that remains to be done to complete the vertical setup of the analyzer.

Acknowledgments

Tom Menten developed and refined the curve-fitting algorithm that is used to process the sensor data. Sandy Dey provided production support of the sensor calibration process. Kari Santos helped develop the vertical calibration firmware.

References

1. G. Dahlquist and Å. Björk, *Numerical Methods*, Prentice-Hall, 1974, p. 134.
2. *Ibid*, p. 5.

An Advanced 5-Hz-to-500-MHz Network Analyzer with High Speed, Accuracy, and Dynamic Range

A three-processor design provides a measurement speed of 400 microseconds per point, fast enough to keep up with manual adjustments. Maximum frequency resolution is 0.001 Hz. Dynamic accuracy is ±0.05 dB in amplitude and ±0.3 degree in phase. Sensitivity of the three receiver channels is −130 dBm, and dynamic range is 110 dB or 130 dB, depending on the sweep mode.

by Koichi Yanagawa

Higher productivity with lower cost is an eternal theme in all industries, not only in production but also in the laboratory. Several of the design objectives for the HP 8751A network analyzer were based on this observation. For example, a faster network analyzer directly increases productivity. In production, it increases throughput by reducing test time. If the line has a tuning process, such as for LC filters, then a network analyzer fast enough to give a real-time response improves tuning efficiency. Powerful analysis capabilities built into a network analyzer improve the efficiency of component evaluation, thereby contributing to reduced development cycles in laboratories and reduced test time in production.

Customer requirements for network analyzer functions and performance are becoming more varied and demanding because these customers have an increasing need for component evaluation and testing to ensure higher quality for their own products. For example, a customer might want to measure the passband characteristics of high-Q quartz crystal or ceramic filters with less than 0.1-dB resolution and relatively coarse frequency resolution, the stop-band spurious below −100 dB with very fine frequency resolution, and the overtone characteristics of these devices, all in the same sweep measurement on the same screen. Just as the filter or resonator characteristics to be evaluated are many and complex, powerful analysis capabilities for ripple, insertion loss, Q, bandwidth, center frequency, resonant frequency, and so on are desired as built-in functions of a network analyzer.

HP network analyzer customers also want a common user interface with earlier HP products for reasons of familiarity, ease of use, and a quick introduction to their working environments. The user interface includes the softkey operation, the HP-IB control commands, and the programming language.

New Network Analyzer

The HP 8751A network analyzer (Fig.1) provides solutions for these customer requirements. The HP 8751A is designed to improve the testing of filters and resonators for telecommunications and commercial products, and to simplify the design and evaluation of circuits, function blocks, and discrete complex devices in the development laboratory. It inherits the look and feel of HP 8752/3 analyzers, making it easy to become familiar with and introduce into a production line. Among its new features are:

- Simulation of impedance matching networks for a device under test
- A list sweep mode for measuring at various user-defined frequency points, power levels, and IF bandwidths
- An order base display mode useful with list sweep for making simultaneous high-speed and high-accuracy measurements in separate frequency ranges
- Simultaneous display of gain, return loss, group delay—the three key parameters for filter applications.

In addition to these functions, the HP 8751A offers measurement performance as high as existing HP products or higher. Its frequency range is 5 Hz to 500 MHz. It has three tuned receiver channels with −130-dBm sensitivity and a maximum frequency resolution of 0.001 Hz. Since dynamic accuracy is one of the most important parameters for network analyzers, the receiver section is designed for typical dynamic accuracy several times better than the specifications of 0.05 dB and 0.3 degree. Typical dynamic accuracy has been measured as less than 0.01 dB and 0.1 degree as described in "Dynamic Accuracy" later in this article.

The HP 8751A has a maximum sweep rate of 80 milliseconds for 200 points with a 4-kHz IF bandwidth. This is fast enough to give a near-real-time response. In a tuning process, production personnel can see the measured characteristics varying as fast as they make adjustments.

A 130-dB dynamic range with list sweep (110 dB for other sweep modes) gives highly accurate spurious measurements of high-Q crystal filters. The wide dynamic range makes lower-level spurious detection possible. With the order base display capability, the measurements can be indicated in a natural form on the CRT screen.

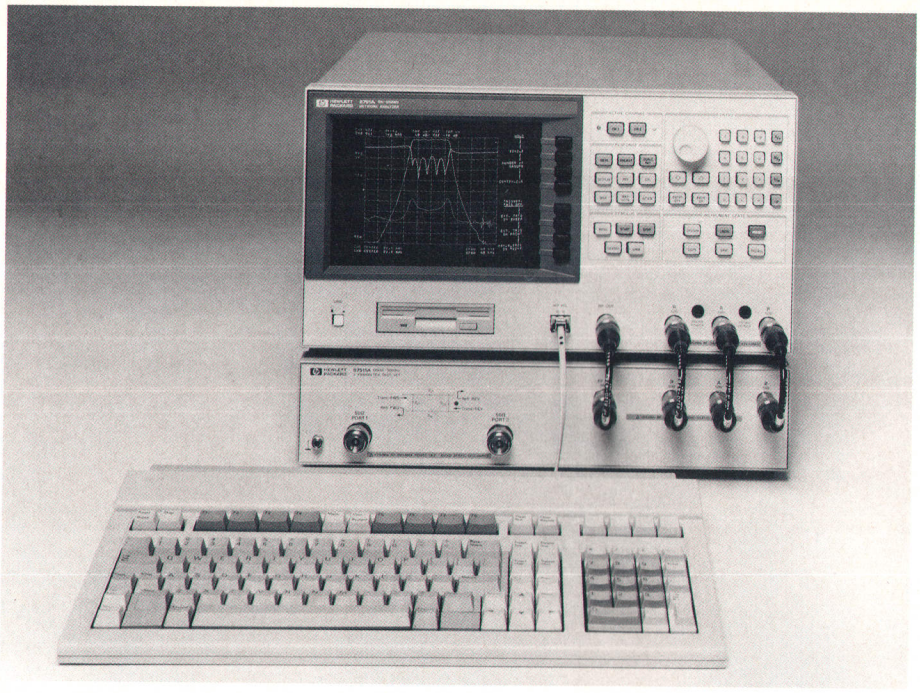


Fig. 1. The HP 8751A 5-Hz-to-500-MHz network analyzer with the HP 87511A 100-kHz-to-500-MHz s-parameter test set and a keyboard for the optional HP Instrument BASIC. Designed for both the production line and the laboratory, it offers a maximum measurement speed of 400 μs/point, list sweep, conjugate matching capability, and many dedicated functions for resonator and filter manufacturers.

The HP 8751A's 25-dB to 35-dB power sweep range is useful for evaluating the gain compression of amplifiers or measuring the drive-level dependency of resonators. No mechanical relays are used, so reliable and faster power sweep measurements can be made with no discontinuity of the applied signal. Continuity of this signal is important for smooth measurements on devices with drive-level dependency such as crystal resonators.

A built-in conjugate matching capability provides circuit designers with a powerful design tool that makes impedance matching problems easy to solve. This capability is also useful for simulating the effects of inserted LC elements. An example is the simulation of the effects of load capacitance in resonator measurements.

Many built-in functions, such as ripple and bandwidth analysis for filter applications or resonant parameter analysis for resonator applications, provide high-speed analysis capability.

A built-in 3½-inch flexible disk drive is standard. MS-DOS® format is supported for easy data transfer between the HP 8751A and personal computers. Users can manipulate measured data using available PC software, such as LOTUS® 1-2-3®.

The HP Instrument BASIC option provides control of measurement sequences, data manipulation, and external devices, making it possible to build a low-cost system without an external computer. I/O capabilities include the HP-IB (IEEE 488, IEC 625), the HP-HIL for an external keyboard, and a general I/O port that provides four bits for input and eight bits for output and is used to control external apparatus such as sequencers, handlers, or scanners.

*MS-DOS is a U.S. registered trademark of Microsoft Corporation.
*Lotus and 1-2-3 are U.S. registered trademarks of Lotus Development Corporation.

Measurement Examples

The main applications for the HP 8751A are testing and evaluation of components and circuits such as filters and amplifiers. Fig. 2 shows a three-trace display for a crystal filter obtained in a single swept measurement. Fig. 3 shows wideband and narrowband characteristics of a crystal filter on the same screen using list sweep. Stopband rejection at lower frequencies is greater than 130 dB. Fig. 4 shows another application of list sweep. The fundamental and overtone responses are measured in one sweep and displayed on the same screen using the order base display mode. Fig. 5 shows the conjugate matching capability used in an LC filter application. The simple simulation capability of conjugate matching can be used, for example, to simulate the load capacitance of a resonator.

System Overview

The major sections of the HP 8751A are the signal source, the receiver, the digital control section, and the power supply.

In the analog sections, many of the 3800 electrical components are mounted on 2-layer or 4-layer boards of 3500 cm^2 area using surface mount technology to save board area. Through-hole components are also mounted on these boards. Fig. 6 shows a typical analog board, the RF and local oscillator (LO) board.

Most of the filters on the printed circuit boards were designed using the HP Microwave Design System (HP 85150B).

In the digital control section, about 600 electrical components are mounted on 4-layer or 6-layer boards of about 1600 cm^2 area using standard through-hole technology and leaded components. Both sides of the digital boards

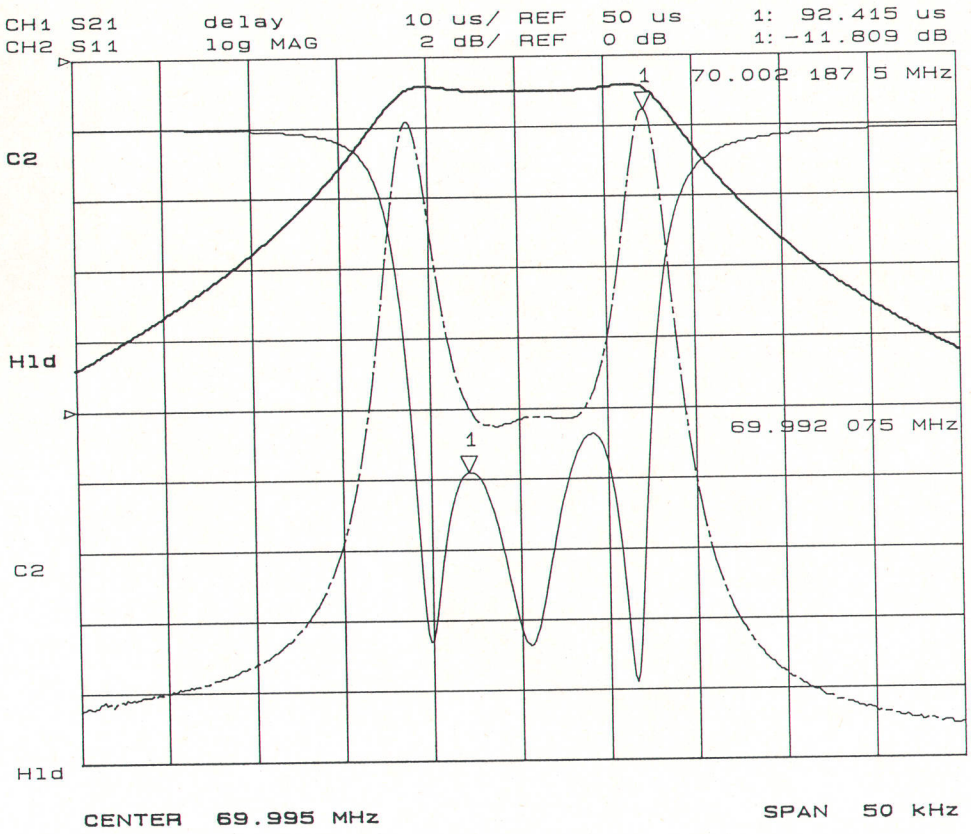


Fig. 2. Simultaneous measurement of gain, return loss, and group delay makes filter evaluation easier and faster. Here the heavier solid line shows the gain of a filter, the thinner solid line shows its return loss, and the dashed line shows its group delay characteristic.

are covered by the fixed-potential planes, that is, the ground plane and the +5V plane, to minimize electromagnetic interference from the printed circuit traces.

As a member of the HP 8752/3 family, the HP 8751A has the same footprint as its predecessors, but is 1.75 inches taller to accommodate the flexible disk drive.

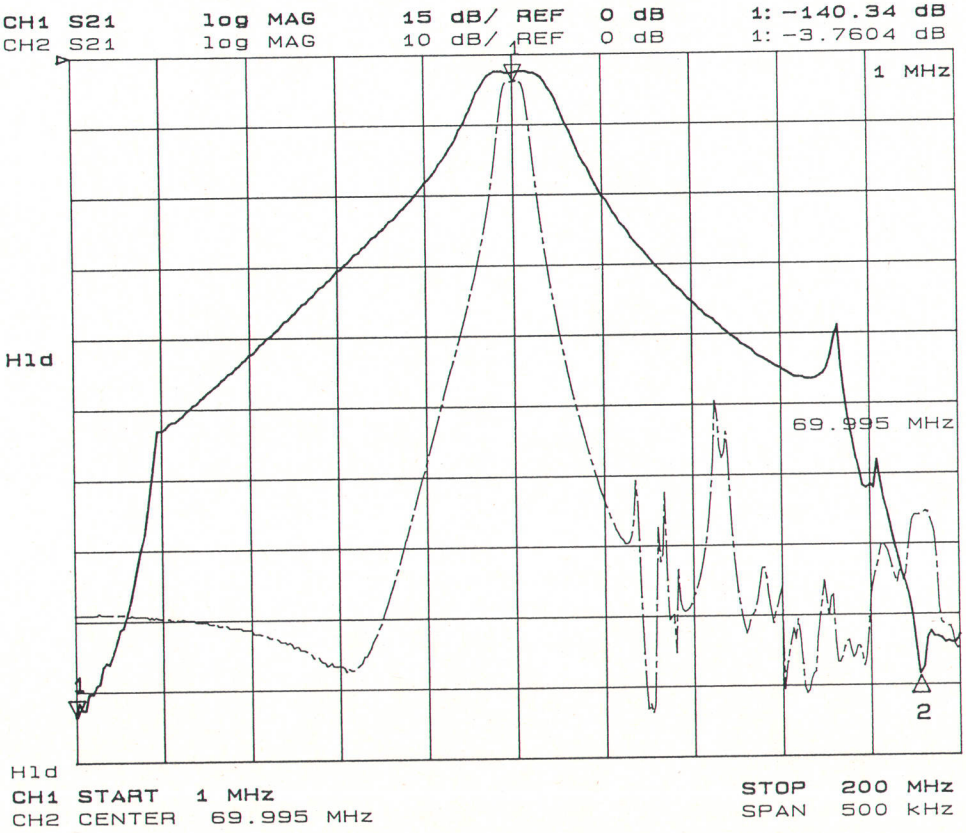


Fig. 3. In the list sweep mode, the user can set the power level and IF bandwidth for each frequency list sweep segment. Dynamic range is 130 dB. Shown here is the magnitude of the gain characteristic of a 70-MHz crystal filter. The solid line shows the list sweep with the frequency range of 1 MHz to 200 MHz divided into three segments: 1 MHz to 69.92 MHz, 69.92 MHz to 70.07 MHz, and 70.07 MHz to 200 MHz. The dashed line shows a normal linear sweep with 69.995-MHz center frequency and 500-kHz span.

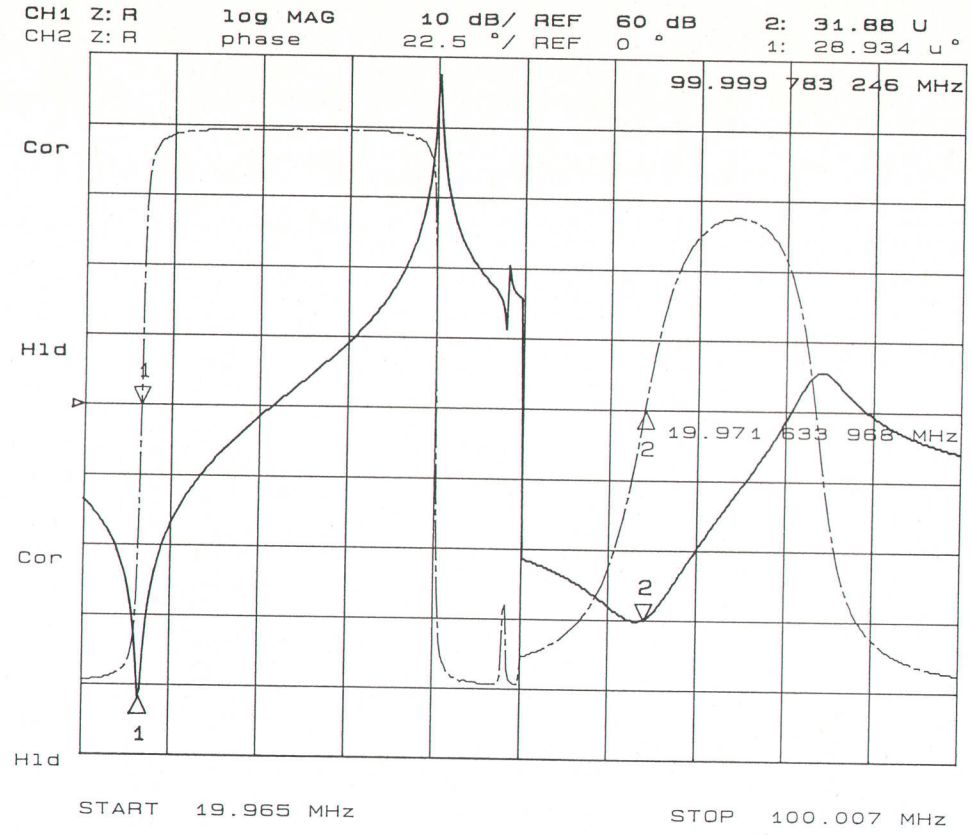


Fig. 4. Overtone characteristics can be evaluated in one sweep on a single screen using list sweep. Here the left side shows the fundamental charateristics (magnitude and phase) and the right side shows the fifth overtone characteristics of an AT-cut crystal resonator.

The signal source section consists of three oscillators. The RF oscillator generates a fixed frequency f_{RF} = 850 or 848.484848484 MHz. The local oscillator (LO) generates a fixed LO frequency $f_{LO} = f_{RF} - f_{IF1}$, where f_{IF1} = 1.5675 MHz. The variable oscillator generates a variable frequency $f_{RF} + f_r$, where f_r is the measurement frequency at the front-panel **RF OUT** connector and varies from 5 Hz to 500 MHz.

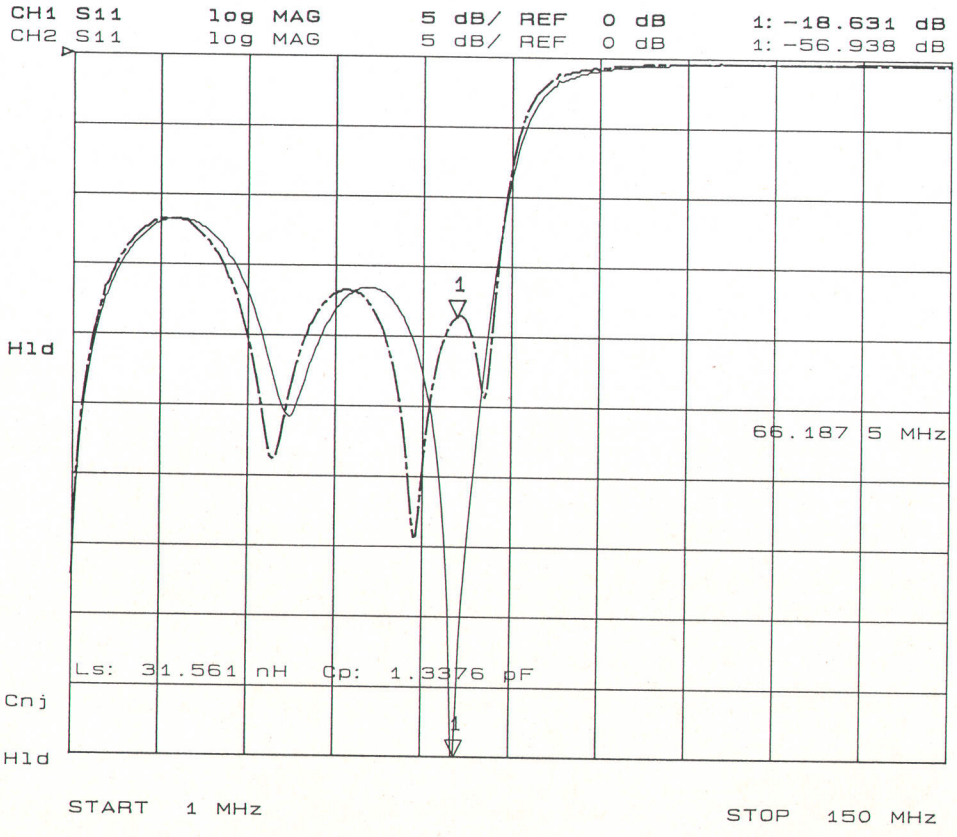


Fig. 5. The conjugate matching capability shows that the return loss characteristics can be improved by adding a 32-nH inductor and a 1.3-pF capacitor.

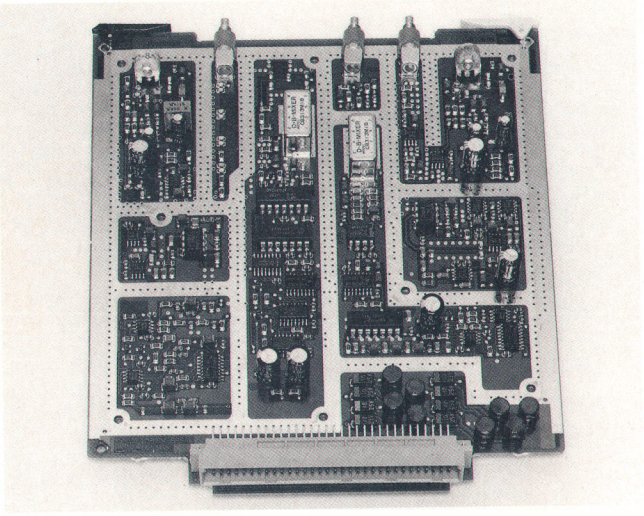


Fig. 6. Most of the analog boards contain both through-hole and surface mount devices. The trace inductors for filters were designed using the HP Microwave Design System.

The HP 8751A contains three identical receiver sections for the R, A, and B input ports of a conventional network analyzer. The signal from each input port is converted to the first intermediate frequency f_{IF1} by a double balanced mixer. This signal is amplified and filtered and then converted to the second intermediate frequency f_{IF2}, which is 5 kHz. The f_{IF2} signal is processed by the ranging and filtering circuits and then converted into digital form by an analog-to-digital converter (ADC).

The digital control section consists of the master CPU and the analog control circuits. The master CPU is a 12.5-MHz MC68HC000 microprocessor with a 25-MHz MC68882 coprocessor. It controls the HP-IB, the display, the keyboard, the flexible disc drive, the test set, and the I/O port at the rear panel. The analog control section also has a 12.5-MHz MC68HC000 CPU and controls the analog section through bus lines to the analog boards. This configuration was determined by trading off cost and measurement speed.

Signal Source

Fig. 7 shows a block diagram of the signal source. It contains five phase-locked loops.

The RF loop generates the fixed frequency called f_{RF}. Normally 850 MHz is selected as f_{RF}, but in 37 special frequency regions, 848.484848484 MHz is selected to avoid residual responses caused by spurious signals generated at the LO mixer. However, this frequency change is canceled if the IF bandwidth is 4 kHz because an increased noise level hides the spurious responses. The f_{RF} frequency change is made by changing the division number of the reference divider, which divides its 50-MHz input by 32 or 33 to output 1.5625 MHz or 1.515151515 MHz, respectively. The f_{RF} signal is phase locked to this reference frequency and to another 25-MHz reference.

The LO frequency f_{LO} is divided by two and phase locked to $f_{RF}/2$ and to half of the first intermediate frequency f_{IF1} to generate $f_{RF} - f_{IF1}$. The first IF of 1.5675 MHz is

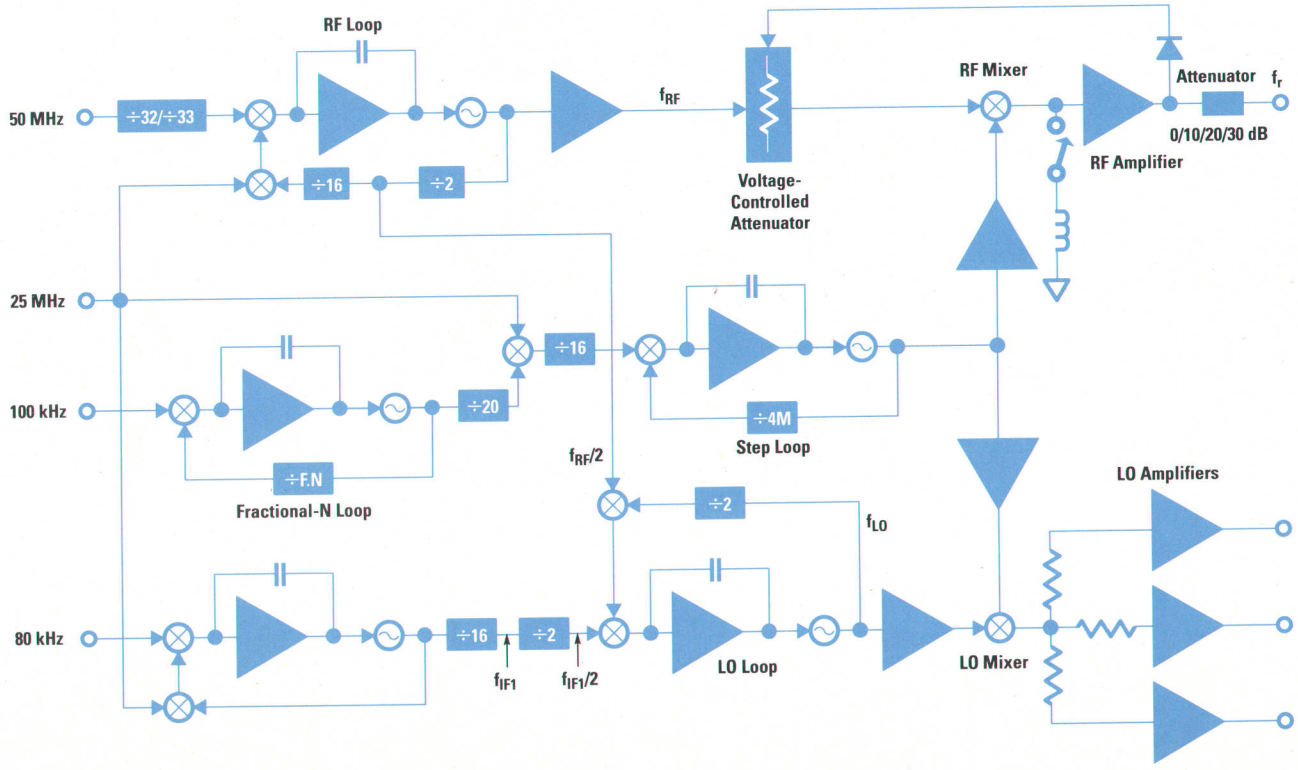


Fig. 7. Block diagram of the signal source section of the HP 8751A network analyzer. Fast settling and low spurious outputs were the key performance goals.

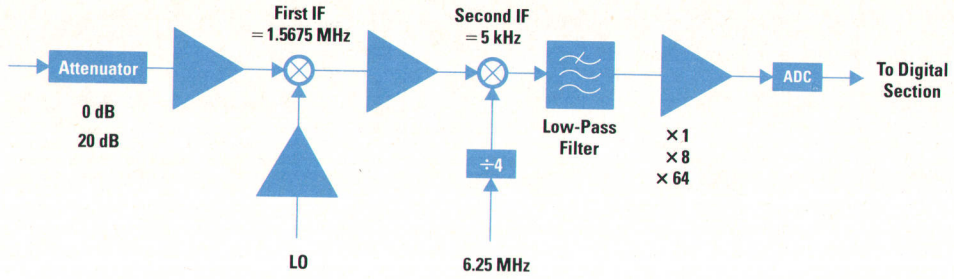


Fig. 8. HP 8751A receiver section. The dynamic accuracy depends mainly on this section.

generated by dividing 25.08 MHz by 16. The 25.08-MHz signal is created in another phase-locked loop.

The variable frequency f_r from f_{RF} + 5 Hz up to f_{RF} + 500 MHz is generated by the step phase-locked loop and the fractional-N phase-locked loop. The output frequency of the fractional-N phase-locked loop varies from 56 MHz to 69 MHz depending on the frequency f_r. This signal is divided by 20 and summed in a mixer with 25 MHz to generate a signal from 27.8 MHz to 28.45 MHz. This frequency divided by 16 is the reference input to the step phase-locked loop. The output frequency of the step phase-locked loop varies from 850 MHz to 1350 MHz as the division number (4M, where M is an integer) is incremented. Each step is one fourth of the reference frequency, so as the reference frequency from the fractional-N phase-locked loop varies, the output frequency of the step phase-locked loop varies with 1-mHz resolution. In other words, the output frequency of the fractional-N phase-locked loop varies with enough resolution to give 1-mHz resolution at the output port of the HP 8751A.

The output signals of the RF phase-locked loop and the step phase-locked loop are the inputs to the RF mixer. The RF mixer output, after being amplified by the RF amplifier, is the measuring signal of frequency f_r. The outputs of the LO phase-locked loop and the step phase-locked loop are the inputs to the LO mixer. The output of the LO mixer is split into three branches, one for each receiver, and is amplified and leveled to generate the local oscillator signals for the first mixers of the receivers. Since the residual responses of a network analyzer are caused by the spurious signals appearing on the LO signal, the isolation between the RF signal path and the LO signal path is a key design parameter. The reason for using $f_{LO}/2$ and $f_{RF}/2$ instead of f_{LO} and f_{RF} in phase locking these two frequencies is to isolate these signals and to reduce the cost of mixing. Buffer amplifiers at the outputs of phase-locked loops and at the input ports of mixers are also important for isolation.

The voltage-controlled attenuator at the RF input port of the RF mixer has a 35-dB attenuation range. This performance is essential for the power sweep and list sweep modes. A capacitive divider configuration is used for attenuator stability.

Combined with the 0-dB-to-30-dB fixed attenuators, which are switchable in 10-dB steps, the 35-dB voltage-controlled attenuator gives an output power level range of –50 dBm to +15dBm. With a 0-dB fixed attenuator setting, the output power range is –20 dBm to +15 dBm. With a 10-dB fixed attenuator range, it is –30 dBm to +5 dBm, and so on.

The maximum power sweep range varies from 25 dB to 35 dB, depending on the stop power setting. If the stop power is set to +5.0 dBm, the fixed attenuator is set to 10 dB and the maximum power sweep range is –30 dBm to +5 dBm, or 35 dB. If the stop power is set to +5.1 dBm, the fixed attenuator is automatically set to 0 dB, and the maximum power sweep range is –20 dBm to +5.1 dBm, or 25.1 dB.

At the output of the RF mixer are two active-L circuits for absorbing dc offset of the mixer output, which occurs when a coarse frequency change is made. Two inductors can be switched in, depending on the frequency settings.

The RF amplifier at the final stage of the RF path provides +15-dBm output power. A level detector feeds back the output level to the voltage-controlled attenuator to level the output power at frequencies of 501 kHz and above. (Below 501 kHz, the frequency response of the output amplifier is very flat, so for faster settling, the output power is unleveled.) The LO amplifiers at the final stage of the LO path are for splitting the local oscillator signal to the first mixers of the three receivers and for isolating the receivers.

Receiver

Fig. 8 shows a block diagram of a receiver section. The receiver is a double-conversion type. The received signal at the input port is introduced to the first mixer through an input attenuator and a buffer amplifier and is converted to the first IF. The attenuation of the input attenuator is either 20 dB or 0 dB; it is selected by the user to set the maximum input level to 0 dBm or –20 dBm, respectively. The first mixer is a commercially available double balanced diode mixer. Its input level affects the upper-range dynamic accuracy of the analyzer. The first IF output from the first mixer is amplified and filtered to reject the unwanted upper sideband and is introduced into the second mixer. The second mixer, a monolithic analog switch, converts the first IF signal to the second IF. The second IF of 5 kHz was selected as a trade-off among measurement speed, the conversion rate of the following ADC, the simplicity of the ranging amplifier, and the ease of generation of the second LO signal. The ranging amplifier amplifies the second IF by 1, 8, or 64 depending on the signal level. Because the amplifier's linearity and ranging stability affect the midrange dynamic accuracy, much care was taken in the design of the amplifier to optimize the signal levels and phase changes at the range switching points. The conversion rate of the ADC is 50 microseconds per point; thus, four data samples equally spaced in time are obtained in one cycle of

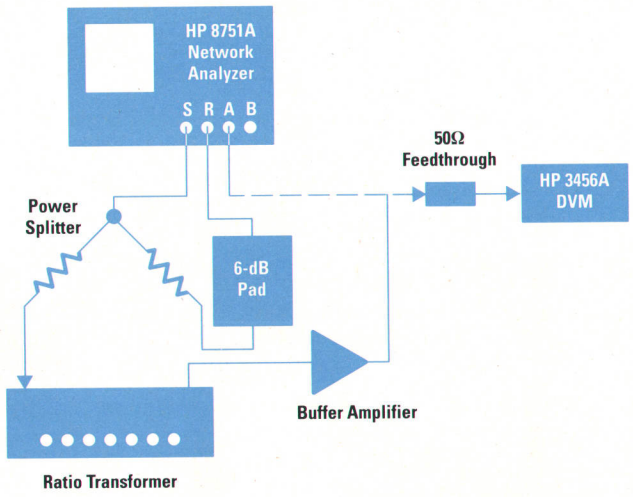


Fig. 9. Test configuration for step I of the dynamic accuracy measurement. Midrange performance was measured using a ratio transformer at 2 kHz. HP 8751A settings: power sweep with 6 dBm center and 0 dB span, A/R measurement, 21 points, A channel attenuation = 20 dB, R channel attenuation = 20 dB, IF bandwidth = 20 Hz, averaging factor = 64, CW frequency = 2 kHz.

the second IF. The ADC is a commercially available, 16-bit, successive approximation type.

Dynamic Accuracy

To verify the design, the dynamic accuracy performance of the HP 8751A network analyzer was tested in three steps during the development phase. Since the performance data obtained in these tests was well below the specified accuracy of 0.05 dB and 0.3 degree, the HP 8751A is verified in production using easier and less costly methods whose accuracy ratio to the specifications is still quite high. Sophisticated automated tests now under development will result in better specifications without degrading the accuracy ratio or the testing efficiency. The performance tests were as follows:

Step I. The first step was an overall performance test from the input port to the final ADC using a ratio transformer at 2 kHz as shown in Fig. 9. The change in the measured values relative to the value measured at the –20-dB (= 0.1000000) setting of the ratio transformer was measured for every 10-dB change of the ratio transformer. At the low power range, 64 measurements were averaged to reduce the error caused by random noise. The mean value and the standard deviation (σ) were calculated for the measured data from 18 receiver boards and it was found that the mean plus 1.3σ in the range from –60 dB to –10 dB was within 0.01 dB for the magnitude ratio measurement and within 0.1 degree for the phase measurement. The mean plus 1.3σ represents the range in which 80% of the samples are expected, assuming a Gaussian distribution. This result is shown as typical performance in the instrument specifications.

Step II. The second step was the testing of the frequency dependent portion of the signal path before the first mixer near the maximum input range. The linearity of this portion becomes better as the signal level decreases, so it is sufficient to test at around the maximum input level. Fig. 10 shows the measurement configuration for this test. A 20-dB fixed attenuator was placed at the reference channel input (e.g., the R channel) to apply a level 20 dB lower than that at the test channel input (e.g., the A channel). The power level applied to both channels was varied by a step attenuator ahead of the power splitter. The measured data taken in a swept-frequency measurement from 1 MHz to 500 MHz with a 0-dBm input level at the test channel was compared with data taken with a –20-dBm input level at the test channel to find the frequency at which the data changed most. Then the dependency on the signal level was measured at that frequency, using the power sweep mode up to the maximum input level. The mean value and the standard deviation were calculated for the measured data from 9 boards and the mean value plus 1.3σ was adopted as a typical value at full-scale input, that is, at 0 dB. The 1.3σ values for amplitude and phase were 0.014 dB and 0.44 degree, well below the specifications of 0.1 dB and 1.2 degree. Since a curve fit to the linearity data obtained in step II showed that the nonlinearity decreases quickly when the input level is 10 dB or more below full scale, the data obtained in step II was adopted as typical performance for the maximum input level.

Step III. The third step was a test of the dependency of the magnitude and phase measurements on the phase change. Fig. 11 shows the configuration for this test. The frequency of the HP 8751A source was set to the worst case found in step II above, and a signal from another signal source, which was frequency locked with the HP 8751A, was applied through a cable long enough to give some phase difference between the reference and test channels. With the frequency of the signal source slightly different from that of the HP 8751A, the phase difference between the input signal and the internal phase reference of the HP 8751A rotates at a rate equal to the frequency

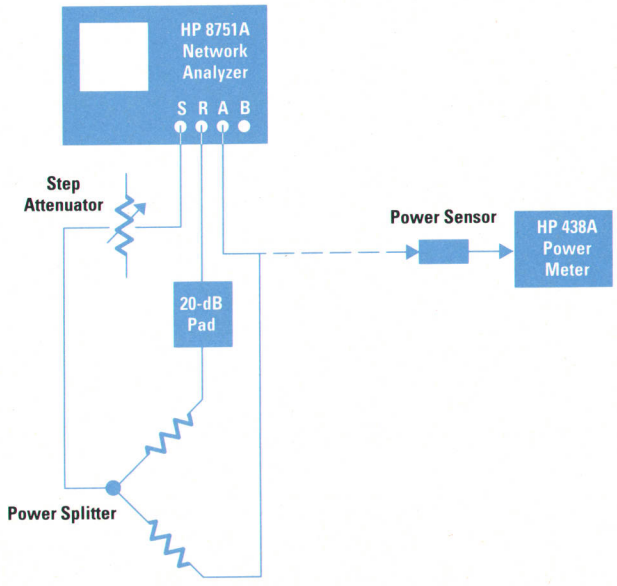


Fig. 10. Test configuration for step II of the dynamic accuracy measurement. High-range performance of the frequency dependent portion of the signal path was tested. HP 8751A settings: linear frequency sweep with 1-MHz start frequency and 500-MHz stop frequency, A/R measurement, power = 6 dBm, IF bandwidth = 20 Hz, A channel attenuation = 20 dB, R channel attenuation = 20 dB.

difference. If the receiver depends on the phase of the signal compared with the internal reference, the trace of the magnitude or phase will change sinusoidally. The peak-to-peak value of this trace gives a measure of this dependency. The resultant values are 0.002 dB peak to peak for the magnitude ratio and 0.01 degree peak to peak for the phase measurement. These results show that the dependence of the measurements on the phase change is small enough to be neglected.

Digital Section

Fig. 12 is a block diagram of the digital section of the HP 8751A. A 12.5-MHz MC68HC000 CPU called the master CPU controls the overall measuring sequence, the graphics system processor, the HP-IB, the disk drive, the test set, the keys on the front panel, the general I/O port, and the HP-HIL for the external keyboard of the HP Instrument BASIC option. A 25-MHz MC68882 floating-point coprocessor is used for the complex mathematical operations involved in correcting and formatting the measured data. A 1M-byte ROM contains the operating system and other control routines. The firmware for the HP Instrument BASIC option is contained in 1.5M bytes of ROM. A 16K-byte EEPROM contains the setup and correction data for the analog circuits. A 64K-byte SRAM backed up by a large capacitor holds the calibration data, the user's display colors, the date and time, and the RAM disk.

A 12.5-MHz MC68HC000 CPU called the slave CPU controls the analog sections. It exchanges commands and data with the master CPU through a 64K-byte communication RAM. A 128K-byte ROM holds the slave CPU firmware, and a 64K-byte SRAM serves as a work area.

To maximize the measurement speed, the master CPU is isolated from many control tasks and from the setting of the analog boards by the slave CPU. The master CPU is isolated from the display tasks by the graphics system processor. This configuration, along with pipelined processing of the measurement jobs and the correcting and

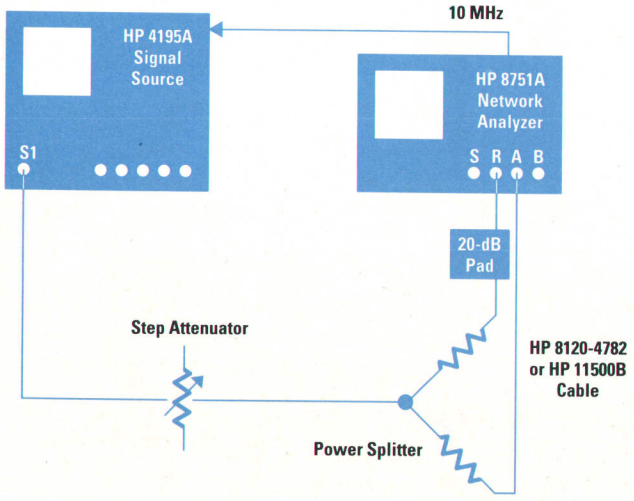


Fig. 11. Test configuration for step III of the dynamic accuracy measurement. Phase dependency of the accuracy characteristics was tested. HP 4195A setting: 340.32 MHz – 0.05 Hz. HP 8751A settings: frequency = 340.32 MHz, sweep time = 20 s. Step attenuator setting = 0 dB.

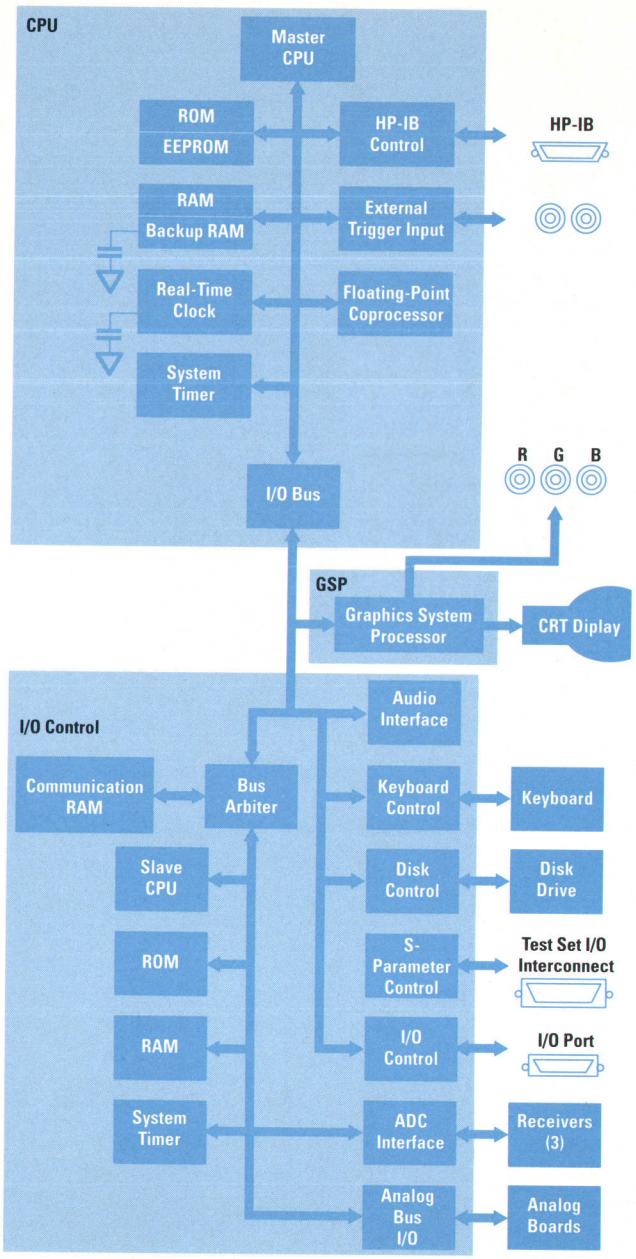


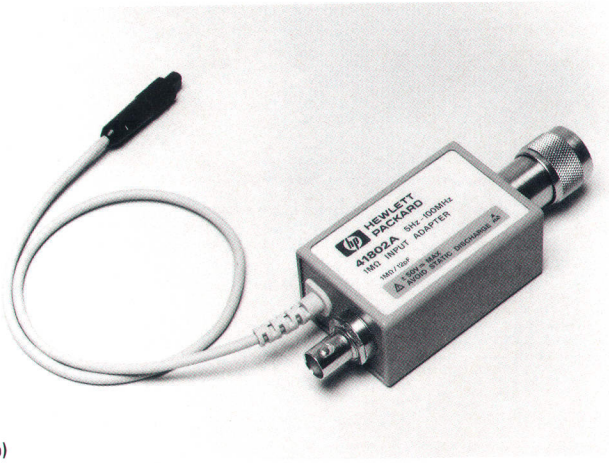
Fig. 12. HP 8751A digital section.

calculating jobs, doubles the measurement rate. The slave CPU sets the stimulus, reads the output data from the ADC in 50 μs after waiting for a settling time of 200 μs for the analog sections, transfers the data to the master CPU through the communication RAM, and then starts to set the stimulus for the next measurement. The master CPU, with the aid of the MC68882 coprocessor, corrects the raw data for internal errors and user calibration, formats the corrected data for the display, and transfers it to the graphics system processor. This process takes less than 350 μs per point. While the master CPU is processing, the slave CPU is collecting the data for the next measurement point.

Firmware Design

In developing the firmware for the HP 8751A, efforts were made not to specialize but to generalize. The

(a)



(b)

Fig. 13. (a) HP 87512A transmission/reflection test set. (b) HP 41802A 1-MΩ input adapter.

operating system is designed to provide a general environment for other firmware routines, similar to that in computers, rather than an instrument-oriented environment like that in earlier analyzers. Almost all of the firmware code is written in the C language, taking into account the portability of the code to future products. The code for the slave CPU and part of the code for the master CPU are written in assembly language for maximum speed. The compatibility of the user interface with that of existing products is taken into account in the design. The HP-IB commands for the same functions are the same as for the HP 8753A/B/C, and the softkey allocations and the structure of the service functions are almost the same.

Installing HP Instrument BASIC into the instrument environment required special considerations. One was how to share the I/O resources, such as the CRT, the keyboards, and the HP-IB port. Another was how to share the processing time between the instrument and HP Instrument BASIC. To resolve the first item, display allocation is designed to be user-selectable by softkey. The user can select all instrument, half instrument and half BASIC, or all BASIC to determine how the CRT screen is allocated. To resolve the second item, the processing time is sliced into 100-ms intervals and is shared almost equally between the instrument and HP Instrument BASIC. The interval of about 100 ms is tuned so as not to interrupt measurements during a sweep operation in the fastest sweep mode.

To keep HP Instrument BASIC programs in the HP 8751A as consistent as possible with those in external controllers, select code 8 is used for the HP 8751A, following the example set by other HP products.

Fixtures

Two types of test sets and an adapter for a high-impedance probe are available. The HP 87511A 50-ohm test set and the HP 87511B 75-ohm test set are 100-kHz-to-500-MHz s-parameter test sets for measuring device s-parameters in both forward and reverse directions. The HP 87512A 50-ohm test set and the HP 87512B 75-ohm test set are 5-Hz-to-500-MHz reflection/transmission test sets that are essentially resistive dividers for measuring reflection and transmission characteristics at low cost. For adequate directivity, these test sets must be calibrated using standards supplied with them. The HP 41802A is an adapter that converts the HP 8751A input impedance from 50 ohms to 1 megohm for the convenience of using the high-impedance probes matched to a 1-megohm termination that are widely available for oscilloscopes. Fig. 13 shows the HP 87512A test set and the HP 41802A adapter. The HP 87511A test set appears in Fig. 1.

Acknowledgments

The HP 8751A design team members who deserve special recognition are Hitoshi Imaizumi, Kazuhisa Utada, Satoshi Roppongi, and Troy Morin for the signal source section, Hiroaki Ugawa for the receiver, Kazuhiro Matsui for the power supply and test set, Jun Kadowaki, the leader of the software and digital group, Akira Nukiyama, Kohichi Takeuchi, Katsuhiro Hakamada, Atsushi Hattori, and Hideki Yamashita for software, Ken-ichiro Nakaya, Kotaro Yamauchi, and Mutsuhiko Asada for software and digital hardware, Akira Uchiyama, Koji Takeda, and Norio Nakano for mechanical design, and Kenichi Katoh for industrial design. Special thanks are also due Kazuyuki Yagi for his general management, design ideas, and encouragement to the team, Masahiro Yokokawa and Satoru Hashimoto who managed the software group, Hiroshi Shiratori who managed the mechanical and industrial design group, and Masao Noguchi for his general management.

A High-Performance Measurement Coprocessor for Personal Computers

This plug-in card brings test and measurement coprocessing power to ISA (Industry Standard Architecture) personal computers with greater calculation speed and better HP-IB performance than its predecessor. It also has DMA capability.

by Michael P. Moore and Eric N. Gullerud

The HP 82324A high-performance measurement coprocessor is a plug-in card for HP Vectra and compatible computers that turns an ordinary PC into a multiprocessing test and measurement workstation. The coprocessor is programmed within the DOS environment using HP BASIC, a de facto standard test and measurement programming language.

The HP 82324A high-performance measurement coprocessor is designed to meet customer needs for higher calculation speed and better HP-IB performance than its predecessor, as well as DMA for better overall system performance. To minimize duplicated effort and maximize reliability, the design of the measurement coprocessor is leveraged from the HP 9000 Model 332 computer. The Model 332 was chosen because of its low cost, high performance, and potential for fitting onto a single full-size PC I/O card.

Hardware Architecture

Fig. 1 is a block diagram of the high-performance measurement coprocessor. At the heart of the design is the 16-MHz MC68030 CPU with its integral memory management unit. The MC68882 floating-point coprocessor can be installed as a socketed option. A custom DMA controller allows rapid transfers of data between memory and devices. Plug-in RAM boards, similar to the Model 332 RAM boards, are available in 1M-byte and 4M-byte sizes. One or two RAM boards can be plugged into the main board, allowing RAM configurations of 1M, 2M, 4M, 5M, and 8M bytes. Built-in HP DIO input/output bus circuitry provides a connection to companion HP GPIO and HP SRM (shared resource manager) interface cards, and the on-board HP-IB interface allows direct connection to HP-IB (IEEE 488, IEC 625) instruments and devices.

Features eliminated from the Model 332 design include the display circuitry, keyboard controller, timer, speaker, and serial I/O. Those functions are performed by PC resources through an emulation process discussed later. Space constraints required the elimination of the memory parity circuitry. The boot ROMs were replaced by a scheme that uses the CPU's memory management unit to remap RAM downloaded from the PC into the boot ROM

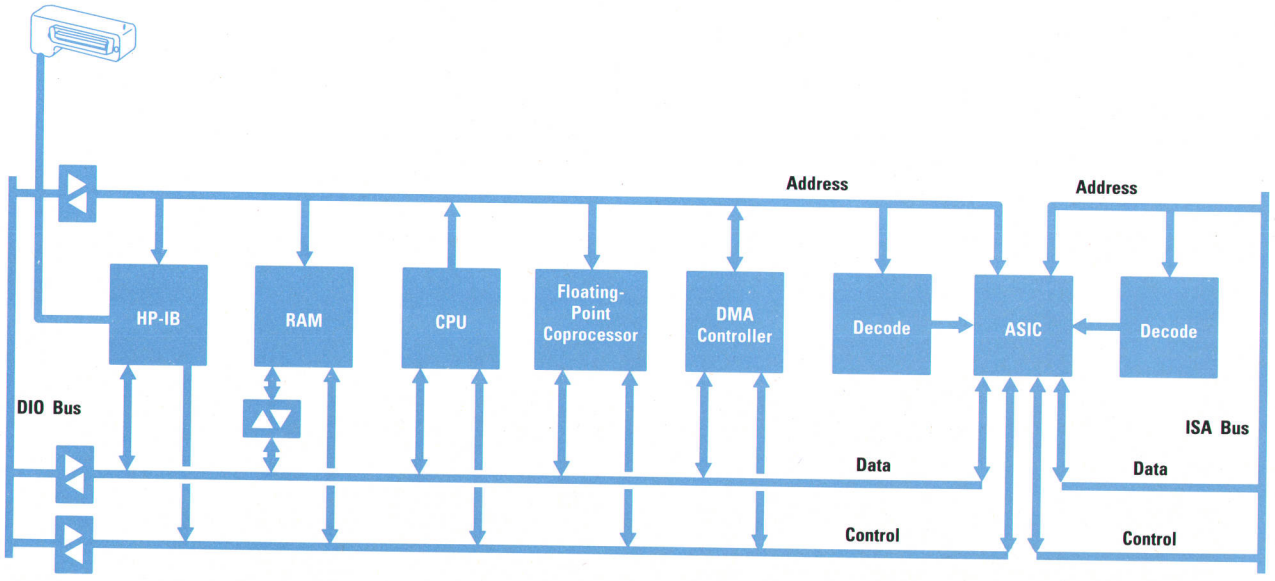


Fig. 1. Block diagram of the HP 82324A high-performance measurement coprocessor.

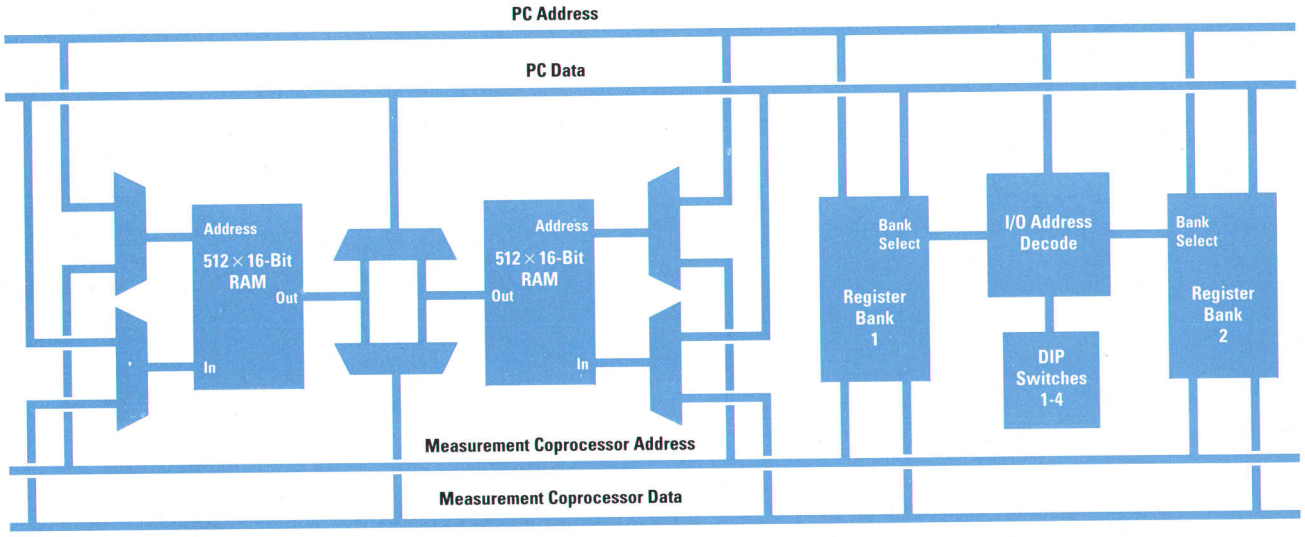


Fig. 2. Block diagram of the application-specific integrated circuit in the measurement coprocessor.

address space. Added to the Model 332 design is an ASIC (application-specific integrated circuit) that provides an enhanced interface to the PC backplane. It includes the same interface mechanism used by its predecessor (detailed in the section below on device emulation), as well as a new memory-based mechanism that allows a higher-bandwidth data path to the PC. Fig. 2 is a block diagram of the ASIC.

PC Backplane Interface

The interface to the PC backplane consists of two banks of eight 8-bit I/O registers, two 1024-byte memory buffers, and three interrupt sources. The base address of the I/O registers is configurable by a DIP switch at the top of the main board. The base address of the memory buffer and the interrupt level are both configurable through the I/O registers. The user can reconfigure the card without having to remove it from the computer and without turning off the computer. Four switches are used to select the base address of the I/O registers. Two of the switches select one of four address ranges within the PC's standard 10-bit I/O address range. These addresses are 250h, 280h, 330h, and 390h. The other two switches select one of four "alias" address ranges outside of the PC's 10-bit I/O address range, making the base address a 12-bit value. In other words, if the first two switches select a base address of 250h, then the other two switches further qualify the address as one of 250h, 650h, A50h, or E50h. This addressing scheme allows up to four HP 82324A cards to be mapped into a single 8-bit register bank, conserving scarce PC I/O resources. Finally, the second bank of registers is selected by the thirteenth address bit. Thus, if the first eight registers are at 250h through 257h, the second eight are at 1250h through 1257h. Again, this scheme conserves PC I/O resources.

One of the I/O registers selects one of eight possible PC interrupt lines and one of eight base addresses for the memory buffer. It also selects either the 8-bit or the 16-bit access mode of the memory buffer or disables it completely.

Another I/O register is used to generate interrupts to the 68030 from the PC. This enables the PC software to emulate devices that generate interrupts, such as the keyboard controller. This register is also used to enable or disable the three interrupt sources that generate PC interrupts. These interrupt sources are a 10-millisecond periodic interrupt, a PC mailbox interrupt, and an address match interrupt.

The 10-millisecond interrupt source is used as the time base for the keyboard controller emulation. In an HP 9000 Series 300 computer, the keyboard controller contains timers used for keeping the time and date, for timeouts, for periodic event generation, and for delays. These timers all have a resolution of 10 milliseconds. Since the PC's periodic system interrupt occurs only approximately every 55 milliseconds, the 10-millisecond interrupt source on the measurement coprocessor was necessary. The mailbox interrupt occurs when the 68030 sets the PC mailbox flag. The purpose of this flag is discussed later. The address match interrupt occurs when a bus cycle initiated by the 68030 is within an address range reserved for device emulation. This interrupt source can be used to implement interrupt-driven device emulation instead of polled device emulation.

Device Emulation

When the 68030 (or some other DIO bus master, such as the DMA controller) generates a bus cycle within a certain range of addresses, the ASIC freezes the bus cycle and waits for PC software to complete it. If the 68030 is performing a write operation, the PC software can read the value and release the bus cycle. If the 68030 is doing a read, the PC software can place a value on the data bus and release the bus cycle. To the 68030, nothing different is happening from writing to an actual hardware device, such as the built-in HP-IB interface, except that the bus cycles take much longer to complete. The PC has complete control over the termination of one of these "trapped" cycles, and can take its time determining whether to terminate the bus cycle normally or abort it with a bus error.

(continued on page 113)

Measurement Coprocessor ASIC

The custom application-specific integrated circuit (ASIC) in the HP 82324A measurement coprocessor design contains the interface between the Industry Standard Architecture (ISA) backplane bus of the PC and the CPU bus of the measurement coprocessor. The interface to the PC supports either 8-bit or 16-bit access modes while the measurement coprocessor interface is fixed at 16 bits.

The ASIC has outputs directly connected to six interrupt lines on the measurement coprocessor interface so that PC software can generate multiple interrupts. On the PC, each interrupting device must have a dedicated interrupt line. At boot time, three software-controlled outputs select one of eight possible interrupts to connect to the ISA bus by external circuitry.

Sixteen registers in the ASIC are addressable from the PC interface and six are addressable from the measurement coprocessor interface. In the PC interface the registers provide a view and limited control of the state of the measurement coprocessor bus. Data can be read from or written to the data bus during trapped accesses, and the cycle is then ended under register control. (For a description of trapped accesses, see "Device Emulation" in the accompanying article). The PC can also assert any of the six shared priority-level interrupts on the measurement coprocessor. A typical use of these interrupts is as follows. The PC receives a keystroke for the measurement coprocessor and asserts an interrupt as if it were the keyboard controller. The measurement coprocessor responds to the interrupt by addressing the keyboard controller register to read the keycode. The read cycle is trapped and recognized by the PC, which writes the keycode to the measurement coprocessor data bus and asserts the signal to end the read cycle.

There are three types of interrupts for the PC, which can be enabled separately. These are generated by a 10-millisecond timer, a semaphore flag in the measurement coprocessor interface, and address matches. Matching addresses are exter-

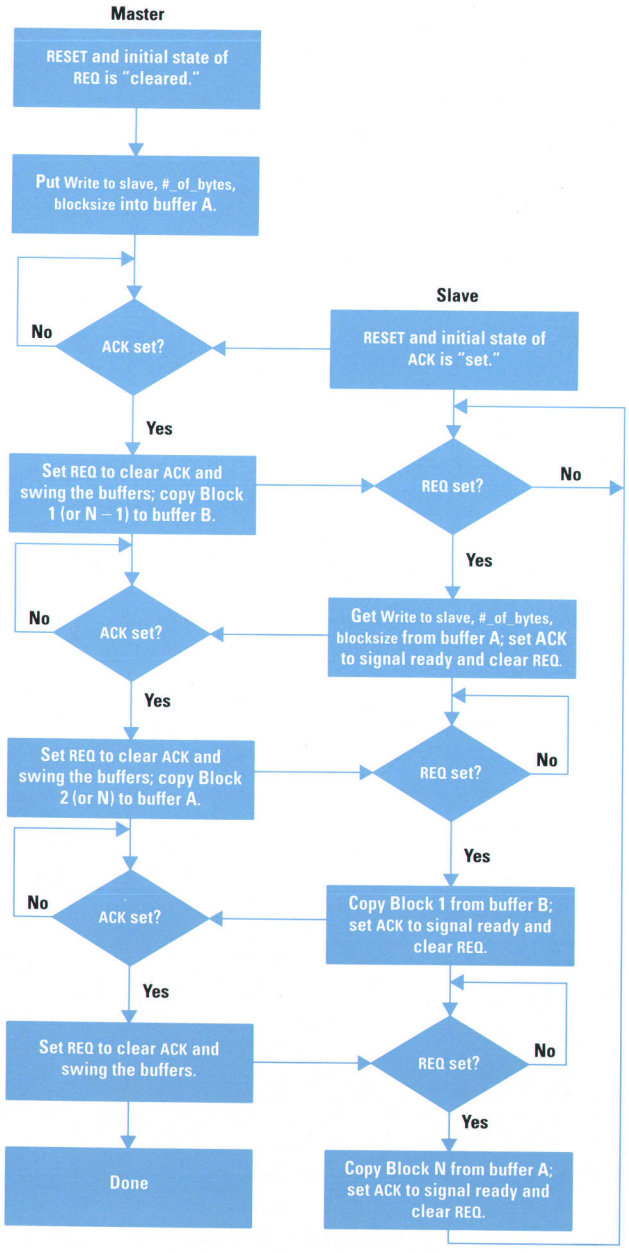


Fig. 1. Data transfer from master to slave.

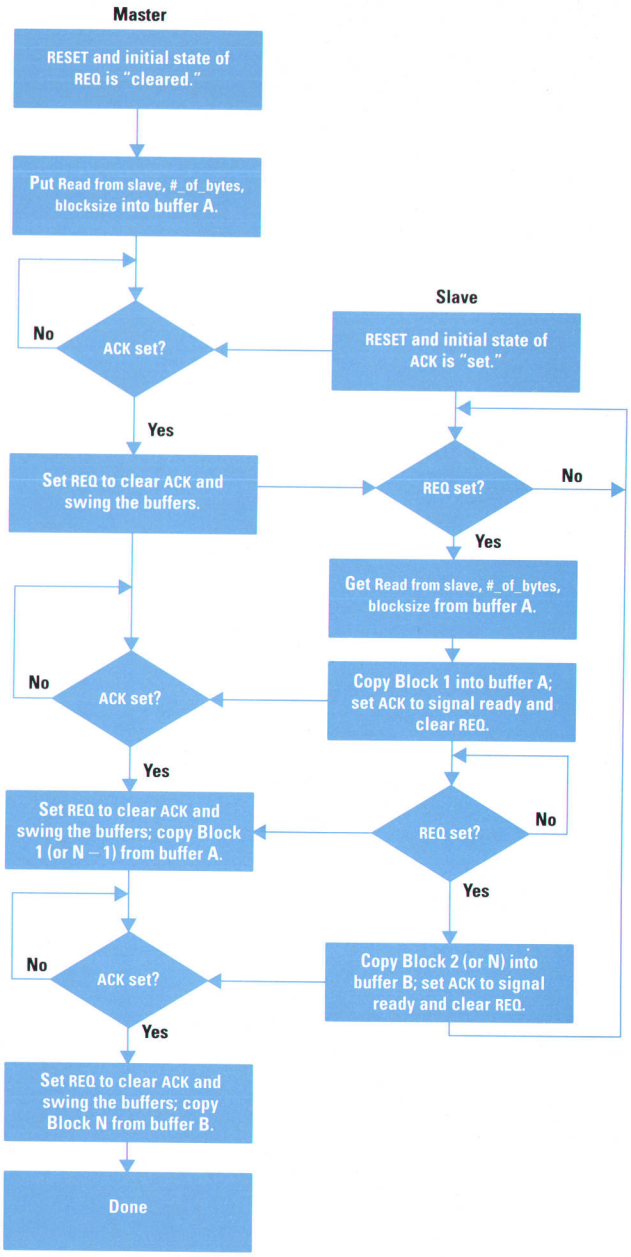


Fig. 2. Data transfer from slave to master.

nally decoded and categorized by three inputs to the ASIC: alpha accesses, graphics accesses, and all others. The PC can view these categories to determine what action needs to be taken.

The DMA controller on the measurement coprocessor relies on external resources to perform byte folding, that is, duplicating data from the low byte of the bus to the high byte when necessary for transfers involving 8-bit devices. Byte folding is handled within the ASIC on demand from the DMA controller.

Byte addressing by the Motorola CPU of the measurement coprocessor is different from byte addressing by the Intel CPUs used in PCs. The least-significant byte of a word has an odd address for Motorola and an even address for Intel. For files as well as for some data types, it is necessary to swap bytes whenever they are stored on the PC. The ASIC can perform byte swapping in hardware if enabled via a register in the PC interface.

Two banks of 512 × 16 bits of RAM are included in the ASIC to facilitate fast block transfers between the measurement coprocessor and the PC. The two RAM banks are configured as a "swing buffer" called the HyperChannel. Each bank can be accessed from either the PC interface or the measurement coprocessor interface but not from both simultaneously. While the PC is accessing one bank, the measurement coprocessor can be independently accessing the other. When both the PC and the measurement coprocessor have finished their respective accesses, the buffers can be swapped or "swung" between the two interfaces. This allows one interface to be continuously filling buffers while the other interface is continuously emptying them in a fully simultaneous process. In this way, a transfer can be accomplished at the full speed of the slower interface.

An identical set of registers is provided for each interface to synchronize transfers using the HyperChannel. Only one interface can initiate a transfer and is therefore considered the owner or master of the HyperChannel, while the other interface is considered the slave. Either interface can be master, but ownership can only be relinquished by the current master, not preemptively seized by the slave. Identical sets of four single-bit write registers and one read register exist in each interface to control the HyperChannel. The read registers include four bits indicating the state of each of the write registers. The write registers allow the assertion of the signals request (**REQ**), acknowledge (**ACK**), error (**ERR**), and change master. The master asserts **REQ**, which in turn clears **ACK** and swings the buffers. The slave asserts **ACK**, which in turn clears **REQ**. Figs. 1 and 2 show the protocol for transfers between master and slave.

The PC can handle the emulation of devices either by waiting for an address match interrupt, as mentioned above, or by polling the status of the 68030's bus. The polling method allows the PC to emulate devices faster than an interrupt-based method, but emulation software must be constantly polling the interface instead of waiting for an event.

While the trapped address mechanism simplifies the PC software and initially eliminated the need to modify HP BASIC software, it has two major drawbacks. First, the rate of data transfer between the PC and the measurement coprocessor is relatively slow. This often makes the transfer of large blocks of data the bottleneck in PC/measurement coprocessor performance. Second, no interrupts, even nonmaskable, can get through to the 68030 when it is in the trapped address state. This makes interrupt-driven I/O without hardware handshaking unreliable at best, and impossible in some situations.

Improved Interprocessor Communication

To circumvent these drawbacks, the PC interface has a secondary communication channel called the HyperChannel. This memory-based mechanism consists of two 1024-byte buffers and three handshaking flags. When one of the buffers is accessible to the PC, the other is accessible to the 68030, and vice versa. Whichever processor is designated the channel master controls the swapping of the buffers when both processors are ready. Since the buffers are being accessed simultaneously, the sustained data transfer rate is as fast as the slower of the two processors. In addition to the performance advantage, this protocol allows the measurement coprocessor and the PC to transact business while both processors remain completely interruptible by other tasks. See "Measurement Coprocessor ASIC," page 112, for a detailed description of the buffer protocol.

A third communication mechanism has been added to make the interface even more flexible. This communication path consists of two mailbox flags, one for the PC and one for the measurement coprocessor. Both mailbox flags can be read by either processor, but only the mailbox flag owned by a processor can be set or cleared by it. When the measurement coprocessor sets its mailbox flag, an interrupt can be generated to the PC. The two processors can then synchronize operations by waiting for both mailbox flags to be set, clearing their mailbox flags, and waiting for both mailbox flags to clear. This protocol allows the measurement coprocessor to generate a PC interrupt without having to be frozen in the trapped address state.

Software Architecture

There are two types of software for the measurement coprocessor: software that runs on the measurement coprocessor itself, and software that runs on the host PC. Both groups of software work cooperatively and concurrently to exploit the capabilities of the measurement coprocessor architecture.

The software that runs on the measurement coprocessor is HP's version of the BASIC language, which has been prevalent in the instrument control world for many years. Because the hardware architecture is leveraged from the HP 9000 Series 300 product line, existing HP BASIC programs can run on the measurement coprocessor, usually without modification. Also, the reference manuals are identical to those shipped with HP BASIC for workstations. Even the measurement coprocessor version of the HP BASIC system is compiled from the same source code that is used to generate the workstation version. (For the history of the development of HP BASIC for the PC, see "Measurement Coprocessor History," page 114.)

In addition to running the HP BASIC system, the measurement coprocessor emulates the system boot ROM in RAM, eliminating the requirement for ROMs on the measurement coprocessor. To achieve ROM-less operation, the PC loads a small boot loader program into the measurement coprocessor's RAM. After testing system RAM, this boot loader program copies the boot ROM image from the PC to the bottom of system RAM, initializes the 68030's memory management unit to map logical

Measurement Coprocessor History

In the early 1980s, the HP 9000 Series 200 computer was HP's premier instrument controller. With HP's version of the BASIC language, the Series 200 made automating part or all of the test and measurement task much easier. HP BASIC became a de facto standard test and measurement language.

The early 1980s also saw the introduction of the IBM personal computer (PC). Within a few years, the IBM PC and compatible machines replaced the Series 200 and 300 computers as instrument controllers for a large portion of the test and measurement market. However, there was no implementation of BASIC for the PC that offered the rich feature set HP BASIC customers had come to expect. HP's answer to this problem was the BASIC language processor—a plug-in card for the PC that temporarily turned the PC into an HP 9000 Series 200 instrument controller.

Two basic strategies for porting the HP BASIC environment to the PC platform were initially considered. One approach was to rewrite the HP BASIC software so that it would run directly in the DOS environment. Because HP BASIC was heavily optimized for the HP 9000 hardware environment, this approach required the commitment of many resources and involved unknown risks.

The other approach was to "port" enough of the HP 9000 hardware to the PC platform so that HP BASIC would run indirectly in the DOS environment. In other words, HP BASIC programs would run on a coprocessor card, while the PC emulated HP 9000 hardware that wasn't on the coprocessor card. While this approach eased the software effort, it required the customer to purchase additional hardware for the PC.

After weighing and considering the two approaches, HP chose the latter. This approach eliminated many of the technical hurdles involved in moving the HP BASIC programming environment to a completely different operating system while retaining compatibility with the original environment.

In the middle of 1987, the BASIC language processor was introduced. It featured an 8-MHz 68000 CPU, an HP-IB interface, boot ROMs, half a megabyte of RAM, an HP DIO bus interface, and special circuitry to interface to the PC's backplane. The heart of the interface circuitry was a mechanism that allowed a program running on the PC to emulate hardware not present on the BASIC language processor. Daughter boards were available to add more RAM or ROM. Sister boards were available to provide HP GPIO and HP SRM (shared resource manager) interfaces. The same HP BASIC software that ran on HP 9000 Series 200 computers ran unmodified on the BASIC language processor with an emulator program running on the PC. Even the boot ROMs were Series 200 boot ROMs.

The emulator software that ran on the PC had three major functions. First, it mapped the I/O resources of the PC onto Series 200 hardware. Second, the emulator software allowed HP BASIC programs to access the DOS operating system by sending data to and reading from an imaginary GPIO interface. Standard DOS commands, commercial applications software, and custom programs could all be invoked from the HP BASIC environment. Third, the emulator software allowed a limited form of background operation. The emulator could be placed in the background, allowing the BASIC language processor to execute the current HP BASIC program while a completely different application, such as an editor or spreadsheet program, ran on the PC.

Because of the hardware emulation scheme, most programs that ran on Series 200 computers could be run on the BASIC language processor with little or no modification. Even programs that directly accessed hardware (such as the graphics frame buffer) could run unmodified. However, this degree of compatibility had a price: severe performance degradation. Improving performance quickly became a priority.

The next major revision of the software included four major architectural changes. First, the boot ROMs were rewritten to speed up the boot process by an order of magnitude. Second, a new HP BASIC binary program was written that implemented the DOS file system. This provided a way for HP BASIC programs to access DOS files directly, without the clumsy emulation scheme described above. Third, internal alpha, graphics, and DOS file system operations were reorganized on a transaction basis rather than a hardware emulation basis. In other words, the HP BASIC binaries that implemented alpha and graphics video operations and the new DOS file system binary program generated transactions instead of register-level accesses. The emulator program on the PC was rewritten to service these transactions as well as its hardware emulation tasks, and the background mode of operation was enhanced to allow DOS file system operations in the background, enabling an HP BASIC program to log data to disk while another DOS application was running. Fourth, a mechanism for a background HP BASIC program to communicate with a foreground DOS application (such as a spreadsheet) was added. Because of these expanded foreground/background capabilities, the BASIC language processor was renamed the measurement coprocessor. It represented a great improvement in boot performance, graphics operations, and mass storage operations, and provided greater versatility in the DOS environment.

However, while the new software emulator made significant improvements in some areas of performance, the remaining issues could only be addressed by upgrading the hardware. These issues included increased computation speed, HP-IB I/O throughput, and DMA capabilities. From this effort came the high-performance measurement coprocessor described in the accompanying article.

references to boot ROM address space to the image in RAM, and transfers control to the RESET exception vector in the boot ROM image. From that point on, the boot ROM and the HP BASIC system operate as if a boot ROM were actually present.

Host PC Software

The software that runs on the host PC is grouped into four parts. The system software configuration part is handled by the CONF.EXE program. The boot process and workstation emulation parts are coordinated by the BASIC.EXE program. DOS/HP BASIC communication is handled partly by the BASIC.EXE program and the HPBLP.SYS device driver, and partly by the POPCOM.COM program.

The CONF.EXE program allows the user to alter the software configuration interactively so that the PC's keyboard, serial ports, and HP-IB cards can be remapped as desired. It also allows the user to specify how to map the PC's disk drives to appear as emulated LIF (HP logical interchange format) disk drives. Configuration information is stored in a file that is used by the emulation programs described below.

Boot Process

The BASIC.EXE program is the main control program for the boot and workstation emulation tasks. When BASIC.EXE is run, it first checks to see if the measurement coprocessor has been booted. If it has not, it starts the boot process. The boot process is handled by two separate programs, both of which are run from BASIC.EXE.

The first program, B0.EXE, is responsible for starting the measurement coprocessor and loading the boot ROM image into it. The second program, B2.EXE, is responsible for managing the boot process, which is an emulation of the HP workstation boot process. Both programs request hardware configuration information from a device driver (HPBLP.SYS), which is installed by CONFIG.SYS when DOS boots up.

When B0.EXE begins execution, it requests the hardware configuration information for the measurement coprocessor being booted from the HPBLP.SYS device driver and

verifies that there are no hardware configuration conflicts with DOS or other applications. It then resets the measurement coprocessor, which leaves the 68030 frozen in the trapped address state trying to fetch the stack pointer from address $00000000. B0.EXE then copies a boot loader program to the HyperChannel buffer, and emulates boot ROM accesses long enough to cause the 68030 to exchange buffers and run the program in the buffer. Once the boot loader program is running, B0.EXE copies the actual boot ROM image to the measurement coprocessor over the HyperChannel.

The B2.EXE program handles the boot process for the measurement coprocessor. This program cooperates with the boot code loaded by the B0.EXE program to simulate the Series 300 boot process. Systems can be booted from a variety of sources, including the optional HP SRM interface or external HFS (HP hierarchical file system) disks. To the user, the PC behaves like an HP 9000 Model 332 computer booting up.

Workstation Emulation

After successfully booting the measurement coprocessor, the BASIC.EXE control program, in conjunction with the B3.EXE program and the HPBLP.SYS device driver, emulates the portions of the workstation hardware not present on the measurement coprocessor. The emulation services provided include the emulation of the workstation's display, keyboard, beeper, and mouse, mapping of the PC HP-IB, serial, and printer ports to emulated workstation I/O cards, and mapping of PC disk drives to workstation LIF and DOS formatted disk drives. A mechanism for running DOS commands from the measurement coprocessor is also provided.

The workstation emulation task can be run either in the foreground or in the background. In foreground mode, the PC behaves as if it were an HP 9000 Model 332 computer running HP BASIC. In the background mode, the user runs DOS applications while the measurement coprocessor concurrently runs an HP BASIC program without a visible display or keyboard.

The BASIC.EXE program is responsible for emulation services common to both foreground and background modes. These services include access to the DOS file system, emulation of the timekeeping and beeper parts of the workstation keyboard controller, and the buffering of alpha video. When HP BASIC is placed in the background mode, BASIC.EXE remains resident in DOS memory while other DOS applications execute.

The full workstation emulation is provided by the B3.EXE program, which is started by BASIC.EXE. B3.EXE provides all emulation services not provided by BASIC.EXE. Communication between BASIC.EXE and B3.EXE is facilitated by code and data resident in the HPBLP.SYS device driver. The common data includes state variables and procedure entry points, and the common code includes interrupt service routines.

Because B3.EXE is not executing during background operation, all processing on the measurement coprocessor stops when an HP BASIC program tries to access emulated services available only in foreground mode. Processing resumes when the measurement coprocessor is switched back into foreground mode.

DOS/HP BASIC Communication

Because the measurement coprocessor runs concurrently with the host PC, there is a potential for dividing the test and measurement task between the processors. To take advantage of this potential, the measurement coprocessor must be able to communicate with DOS and DOS applications. The measurement coprocessor software supports this interprocessor communication with three basic mechanisms: shared file access, MultiCom, and PopCom.

When the measurement coprocessor is in the background mode, an HP BASIC program can write to or read from a DOS file, even if a foreground application is using the file system. While DOS is not a multitasking operating system, it is possible for a collection of interrupt service routines to give a background application access to the DOS file system when the foreground application is not using it. The background service routines in BASIC.EXE, in conjunction with the interrupt service routines in the HPBLP.SYS device driver, create this illusion of concurrent use of the DOS file system, which is the basis for this form of interprocessor communication. While this scheme can be used by any application that can access a DOS file, it is relatively slow.

The MultiCom mechanism for DOS/HP BASIC communication uses the existing I/O capabilities of DOS and HP BASIC compiled subroutine libraries to allow HP BASIC programs to send messages to and receive messages from DOS applications. With MultiCom, an HP BASIC program running in the background can generate keypresses to the DOS application running in the foreground by calling one of several compiled subroutines. Conversely, a DOS application running in the foreground can send messages to the HP BASIC program running in the background by writing to a measurement coprocessor device file (similar to the .PRN file for a printer). To illustrate this mechanism, assume that the user wants to run a test, then integrate the data into a spreadsheet to generate a printed report. The user programs the measurement coprocessor to handle the setup, initiation, and data gathering parts of the test, using MultiCom to receive parameters from and send data to the spreadsheet program. The user programs the spreadsheet program to start the test on the measurement coprocessor, receive the data, and graph the results on a form. The user uses HP BASIC for the test task and the spreadsheet program for the presentation task instead of trying to do both tasks in one environment or the other.

Another use of the MultiCom mechanism is to support multiple measurement coprocessor configurations. With MultiCom, a DOS application can communicate with up to three measurement coprocessors in a PC. In addition, one measurement coprocessor can communicate with another independently of the DOS application running in the foreground. In this manner, powerful multiprocessing test systems can be configured and controlled in one host PC.

Finally, a user may simply want to start a test running on the measurement coprocessor, then go do something else

in DOS, such as reading electronic mail or writing a document. However, if the user wants to be notified when the test finishes, or just wants to see how far along it is, it would be annoying to have to stop working, start up HP BASIC, type a couple of commands, exit HP BASIC, and restart the DOS application. The PopCom mechanism supports a kind of "pop-up" interface to HP BASIC running in the background. With PopCom installed, an HP BASIC program in the background can cause a dialog window to pop up over the foreground DOS application, temporarily taking control of the PC. After the user enters a response, the pop-up window disappears, and the foreground application continues execution. This type of pop-up interface is already popular with PC users, and fits naturally into the DOS environment.

Acknowledgments

The original HP 9000 Model 332 CPU board design was done by Mark Anderson of the former Fort Collins Systems Division. Carl Thomsen did the ASIC design before transferring to the Vancouver Division before the end of the project. Greg Parets joined the project toward the end and was instrumental in the final stages of development and release to manufacturing. The software development team for the HP 82324A's predecessor, the HP 82300A basic language processor, was managed by Andy Rood and included Terry Leeper (project leader), Everett Kaser, Leon Nelson, Roberto Orozco, and Tony Yokoyama. Roberto also took over as project manager from Andy during the second major revision of the BLP software. David Kepler managed the hardware design and software QA of both the HP 82300A and the HP 82324A. Support from the RMB/WS team at MSO was invaluable.

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, CA 94303-0724

ADDRESS CORRECTION REQUESTED

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

HEWLETT-PACKARD
JOURNAL

April 1992 Volume 43 • Number 2

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, California, 94303-0724 U.S.A.

Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

01029627 LOAD
MR. GEORGE PONTIS
SUITE 409
24 COVE LN
REDWOOD CITY CA 94065-1528

CHANGE OF ADDRESS:

To subscribe, change your address, or delete your name from our mailing list, send your request to Hewlett-Packard Journal, P.O. Box 51827, Palo Alto, CA 94303-0724 U.S.A. Include your old address label, if any. Allow 60 days.

5091-3792E