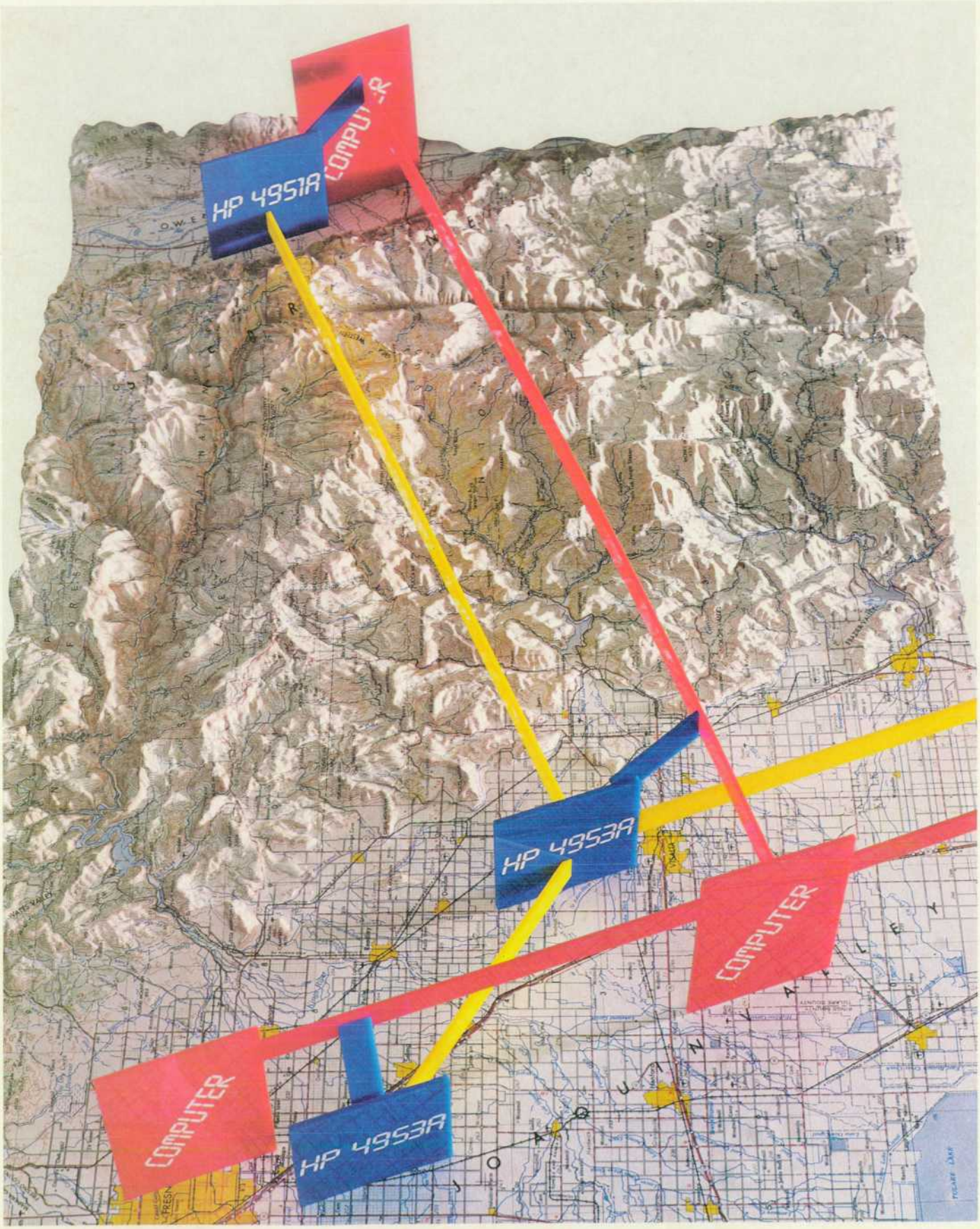
HEWLETT·PACKARD
JOURNAL

JULY 1985



HEWLETT-PACKARD JOURNAL

July 1985 Volume 36 • Number 7

Articles

In this Issue

Not everything that gets transmitted over a data communication network is data. Each block of data is surrounded by synchronization characters and is usually followed by error detection bits. Between blocks of data there are overhead transmissions necessary to initiate, maintain, and terminate computer transactions. Each network uses a particular protocol, which is a definition of the synchronization and control characters and the format of data blocks and control messages. Each network also uses a particular data code. Fortunately, the number of protocols and data codes isn't as large as the number of networks. There are a handful of standard protocols and most networks use one of them. The same holds for codes.

When there's trouble on the network, it isn't necessarily a hardware failure, and its effect might only be observable as a deviation, perhaps intermittent, from the required protocol. Protocol analyzers are instruments that are specially designed to monitor in-service data communication lines, trap errors, and provide enough information to isolate the source of an error to an individual component of the network. This issue gives you the design story of two Hewlett-Packard protocol analyzers. The HP 4953A, the subject of the articles on pages 4, 12, and 18, is for analyzing high-speed networks and is used mainly at computer centers, although it can also be used in the field. The HP 4951A, described in the article on page 24, is a lightweight, portable, field service analyzer for medium-speed networks. Both analyzers are capable of not only monitoring but also simulating data transmissions. This makes them useful for checking components of a network individually, or for verifying the operation of a component before it's installed in the network. Both analyzers are also capable of remote operation. One HP 4953A Protocol Analyzer can control and receive data from several HP 4953As or HP 4951As at remote sites. This means that one engineer or technician can test an entire network from a central site. Our cover illustrates this capability. Two special HP 4951A features help the field service technician: Autoconfigure determines the network's protocol and data code automatically, and bit error rate testing can eliminate the need to carry a separate instrument for this purpose.

Hewlett-Packard's Semiconductor Productivity Network (SPN) is a group of integrated modules, primarily software, that are designed to automate the production of integrated circuit chips. The aim is to keep the yield of good chips high by correlating data from the entire factory to minimize yield fluctuations and discover their causes. By far the most important part of the operation is the process equipment, where temperatures and chemical concentrations are so critical that their control has always been something of an art. In the article on page 30, you'll meet the SPN module called PC-10, which is designed for direct monitoring and control of process equipment such as furnaces, ion implanters, and plasma etchers, as well as monitoring and measuring equipment. The outgrowth of work begun at HP's central research laboratories (see "A Process Control Network" in our June 1981 issue), PC-10 improves the repeatability of the production process, providing the predictability needed to get yields up and keep them there.

-R. P. Dolan

What's Ahead

In August, we'll have an article on the objectives and basic principles of HP's next-generation computers, which are now under development. Four articles will cover the design of the HP 3326A Two-Channel Synthesizer, a signal generator that produces either a pair of independent signals or a complex combination of two signals in the frequency range of dc to 13 megahertz. Another article will discuss TC-10, a module of HP's Semiconductor Productivity Network (like PC-10 in this issue). TC-10 takes data from various incompatible sources and makes it accessible wherever it's needed.

A Protocol Analyzer for EDP Centers and Field Service

It's the latest member of a family that also includes a low-cost portable analyzer for field service and a high-speed BASIC-programmable analyzer for data communications research and development.

by Aileen C. Appleyard, Roger W. Ruhnow, William Grant Grovenburg, and Wayne M. Angevine

THE DEVELOPMENT OF COMPUTER NETWORKS has led to the need for reliable data communications, not only between elements of the network such as terminals and the CPU, but also between networks—for example, between several computer systems and a mainframe system.

Since these systems may not be at a common site, data needs to be transmitted from one site to another. This can be done using either dedicated lines or data networks. For two systems to exchange data, the data must be in a common format for both systems, and must contain overhead information—for example, the address of the unit to which the data is to be sent, what type of data is being sent, and error checking information. It is usual for the receiving system to transmit an acknowledge message in response to information received so that if some data is lost in transmission it can be retransmitted.

OSI Model

Networks are structured in levels, each level having a specific function. One commonly used method of structuring a network is the ISO Open Systems Interconnection model, which has seven levels, as shown in Fig. 1.

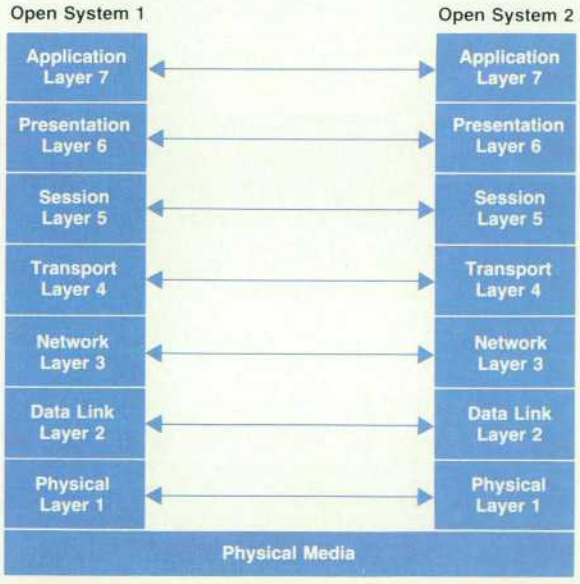


Fig. 1. *In the ISO Open Systems Interconnection (OSI) model, networks have seven layers. Communication takes place between like layers in different systems.*

The physical level transmits the digital data over the physical communications link. For example, the standard RS-232-C link uses a voltage of less than −3V for a mark (logical level one) and greater than +3V for a space (logical level zero). Other commonly used physical layer standards are RS-449 for systems operating at higher data rates, and X.21, which is commonly used in Europe on public data networks.

The function of the data link level is to ensure that error-free data is received at the network level. This is achieved by partitioning the data into frames which are transmitted

Fig. 2. *The Hewlett-Packard family of protocol analyzers includes (top to bottom) the HP 4955A, a 72-kbps BASIC-programmable analyzer for datacom R&D, the HP 4951A, a 19.2-kbps low-cost portable analyzer for field service, and the HP 4953A, a 72-kbps analyzer for field service and datacom center applications.*

sequentially, and requiring a frame acknowledgment signal from the receiving system. If this signal is not received within a fixed time, the frame is retransmitted. This level also controls the rate of frame transfer, so that systems operating at different speeds can exchange data.

The network layer combines frames into larger groups known as packets and routes the packets to their destinations. This route would be fixed in a simple network, but in a more complex network the routing would vary according to system loading.

The transport layer accepts data from the session layer and splits it into smaller units for efficient transmission to the network layer. It also isolates the session layer from any impact of hardware changes in the system.

The session layer is the user's interface to the network. The user establishes a connection with a process on another machine in this layer, and billing information is contained in this layer.

The presentation layer performs text compression, encryption, and file conversion functions.

The application layer's function is determined by the individual user and typically is used to control the interface of two user programs on different machines.

Protocol Analyzers

The HP 4955A, HP 4953A, and HP 4951A Protocol Analyzers (Fig. 2) have been developed to aid in the design and maintenance of these networks.

The HP 4955A is primarily used in data communications research and development, where the features of high-speed operation to 72,000 bits per second and the BASIC programming language are required. The HP 4953A is used in the high-end field service area and in EDP centers, where speeds of 72,000 bits/s may be required, but in a less expensive and more portable instrument. The HP 4951A is used in field service, operates to 19,200 bits/s, and is a low-cost, easily portable instrument.

This article and the articles on pages 12 and 18 describe the capabilities and the design of the HP 4953A Protocol

Fig. 3. *The HP 4953A Protocol Analyzer connects to the network under test through a network-specific interface pod. It has a menu-driven softkey user interface and a built-in cartridge tape unit for reading and storing data and applications programs.*

How Protocol Analysis Can Help

A problem situation that occurred at HP's Colorado Telecommunications Division illustrates how the HP 4953A Protocol Analyzer can be used to diagnose data communications difficulties. When a file was to be printed from an HP 3000 Computer, the response time of the printer was very slow. A line would be printed, then there would be a wait of several seconds before the next line was printed, which was unacceptable.

An HP 4953A was installed to monitor the activity between the HP 3000 (DCE) and the printer (DTE) during printer dump routines. It was discovered that the DTE was not returning an acknowledge character AK in response to the enquiry character EQ sent by the DCE. This was causing a timeout of the DCE of two seconds before the next block of text was sent to the DTE. This was measured using the HP 4953A's cursor timing feature.

To verify the problem, the HP 4953A was then configured to simulate the response of the DTE (Fig. 1). The printer dump routine was run with no timeouts occurring. The time from EQ to the start of the next block of text from the DCE was only four milliseconds, and the time to print a page of text was reduced by 59%.

```
Hewlett - Packard   4953A   Protocol Analyzer

                                        Triggers left = 62

Device Emulating  DTE

Transmission Mode is  FDX

Interface Lead Control is  User Defined

Block  1:

     When DCE %   then goto Block  2

Block  2:

     Send    %
       and then
        Goto Block  1
```

Fig. 1. *Simulate menu.*

Analyzer, the newest member of the family (Fig. 3). The HP 4951A is described in the article on page 24. Most operational characteristics are common to all three analyzers. For example, all three instruments use the same menu-driven softkey approach in guiding the user through the selections necessary for correct operation.

A protocol analyzer has two main functions: to monitor data and to simulate data. In the monitor mode, the analyzer decodes data either from the internal buffer memory or from a line and displays it in a readily understandable format. Data in the internal buffer may have been stored there at an earlier time or transferred there from a DC100 cartridge tape. Data may be stored to tape and subsequently analyzed on any of the instruments.

The Setup menu is used to establish the characteristics of the system being used, including the protocol, the data rate, the data code, and the error checking scheme. These are all parameters that are defined by the network under test. The HP 4953A is able to operate with the SDLC, HDLC, X.25, X.75, SNA, DDCMP, Bisync, and asynchronous protocols.

The Monitor menu allows programs to be written to trigger the start and stop of the display and storage functions so that a large quantity of irrelevant data need not be scanned. For example, one of a group of terminals may not respond to data from the mainframe. This particular terminal's data flow can be examined by triggering on the address of that terminal and examining the data after this address. There are also five timers and counters, which can be used to select data to store or display—for example, by storing data to tape for a specified time after a trigger event or after a selected number of specified events (see Fig. 4).

The Simulate menu allows the user to select whether to simulate a DTE (Data Terminal Equipment)—for example, a keyboard or computer—or a DCE (Data Circuit-Terminating Equipment)—for example, a modem—and whether the transmission is to be full or half duplex. Programs are then written using the softkey-directed syntax to generate data in the format of the selected protocol and transmit it via the analyzer's interface pod to the line (Fig. 5). The leads of the interface pod can be set to simulate handshake sequences between DTE and DCE, and data strings can be specified so that a complete message can be transmitted.

The Display menu allows several different display formats to be selected depending on the protocol, so the user can easily look at the data of interest. DTE data, DCE data, or both can be shown on the display and, if required, the data can be separated into columns to decode the protocol header functions. In addition, data may be shown in conjunction with the logical state of selected interface leads.

The Run menu is used to execute the selected Monitor or Simulate menus. In the monitor mode it lets the user select the source of data, either on-line or from the internal data buffer, and in the simulate mode the analyzer is either allowed to free run or made to stop when the buffer or the tape (if selected) is full. The Execute key in this menu starts the analyzer running.

The Examine Data menu allows detailed analysis of the data that has been stored in the buffer using the selected monitor menu. In this mode it is possible to perform timing measurements between characters in the display.

The Mass Store menu allows control of the DC100 tape unit. The tape can be used to store buffer data and its associated menus, nonstandard data codes, and application programs.

The Printer menu allows the screen to be printed to an RS-232-C ASCII printer. Both menus and buffer data can be printed.

The Remote menu allows configuration of the HP 4953A as either a controller of up to 16 slave instruments or as a slave. After the configuration of the remote system, the operation to be performed remotely is selected. Functions available are uploading and downloading of data and menus and application modules. The HP 4953A may control or be controlled by an HP 4951A or an HP 4955A with certain limitations.

The Application menu is used to load and execute application programs written for the instrument. Typical programs include additional protocol decodes, tape edit programs, and high-level emulation programs that cause the instrument to simulate a network.

The Extended Tests menu is used for troubleshooting the instrument and directs the technician to a problem area.

Remote Operation

The features of the HP 4953A suit it to troubleshooting problems in data communications. These problems might arise from improper configuration of hardware or software, or from the failure of hardware components. The environ-

```
Hewlett - Packard   4953A   Protocol Analyzer

                                                    Triggers left = 51

Device Emulating  DTE

Transmission Mode is  HDX

Interface Lead Control is  User Defined

Block  1:

     When DTE  1  8  ########  ##01####  ########  ########
       or
          DCE  1  8  ########  ##01####  ########  ########0   then goto Block  2

Block  2:

     Increment Counter  1
       and then
        Beep
       and then
        Goto Block  1
```

Fig. 4. *An HP 4953A program to use one of the five built-in counter/timers to count data packets in DTE and DCE data.*

```
            Hewlett - Packard   4953A   Protocol Analyzer

                                                        Triggers left = 63
Device Emulating  DTE

Transmission Mode is  HDX

Interface Lead Control is  User Defined

Block  1:

     Increment Counter 1
       and then
        Send 1 % 10000011 EB!
       and then
        Send 1 % 00000000 THIS IS AN I FRAME EB!

     If Counter 1 is >    4   then goto Block  3

Block  2:

     Goto Block  1

Block  3:

     Stop Tests
```

```
            Hewlett - Packard   4953A   Protocol Analyzer
```

DTE ADDRESS	DTE TYPE	DTE N(S)	DTE P/F	DTE N(R)	DTE INFO	DTE FCS	DCE ADDRESS	DCE TYPE	DCE N(S)	DCE P/F	DCE N(R)	DCE INFO	DCE FCS
%	SNRM		0			G							
%	Info	0	0	0	THIS	B							
%	SNRM		0			G							
%	Info	0	0	0	THIS	B							
%	SNRM		0			G							
%	Info	0	0	0	THIS	B							
%	SNRM		0			G							
%	Info	0	0	0	THIS	B							
%	SNRM		0			G							
%	Info	0	0	0	THIS	B							

BLOCK NUMBER = 1

Fig. 5. *An HP 4953A program (top) to transmit HDLC SNRM (set normal response mode) and the resulting display (bottom).*

ment may be an existing system, an expansion of an existing system, or an installation of a new system. In most cases, the first person to troubleshoot the system will not have an extensive data communications background. Most likely, the troubleshooting procedure will be confirmation of software and hardware configurations and confirmation of hardware integrity. If these procedures are unsuccessful, then a system specialist or data communications specialist must get involved.

The specialist needs to examine the activity on the communication link. Tools available might include diagnostic trace listings provided by one of the communicating devices and/or protocol analyzers. Unfortunately, the data of interest is not immediately available. Either data must be collected at the site and delivered, or the site must be visited. Either case involves added expense and downtime for the customer. If data is delivered to the specialist, there is a danger that the data does not reveal the problem. On the other hand, site visits reduce the availability of the specialist. The remote feature set of the HP 4953A is tailored to reduce or eliminate these problems.

Using a local HP 4953A the specialist can remotely configure, execute, and obtain data from a remote HP 4953A. All that needs to be done at the remote site is to connect the appropriate pod on an HP 4953A to the link under test and attach the RS-232-C connector on the back panel to a leased or switched line via an asynchronous modem, matching the bit rate of the port to that of the modem. The specialist now has full control of the remote instrument. Any number of tests can be run and the results obtained for local analysis.

The remote feature set of the HP 4953A allows manipulation of all basic functions. From a controller, the following operations can be performed on a remote unit (slave). Upload and download are defined with respect to the controller, i.e., a menu is downloaded from the controller to the slave.

- Movement of menus
  - Upload Setup, Monitor, Simulate, Run, and Display menus
  - Download Setup, Monitor, Simulate, Run, and Display menus
- Manipulation of mass storage
  - Slave's mass store catalog
  - Upload Mass Store menu
  - Download Mass Store menu
  - Execute slave's Mass Store menu
- Movement of data
  - Upload timers and counters
  - Upload captured data

- Download captured data
- Manipulation of applications
  - Slave's application module catalog
  - Upload application module
  - Download application module
  - Execute slave's application module
  - Delete slave's application module
- Information on slave
  - Identify slave
  - Slave's status
  - Slave's memory use
- Miscellaneous control
  - Soft reset slave
  - Set slave's buffer size
  - Lockout slave's keyboard
  - Execute slave's Run menu.

Both point-to-point and multidrop remote configurations are possible, as shown in Fig. 6. In the multidrop configuration, a controller HP 4953A can work with up to sixteen slaves. A particular slave is selected by an address field in the controller's Remote menu.

Remote User Interface

The HP 4953A's user interface is designed to be user friendly through the use of softkeys and menus. An HP 4953A is configured as a controller or a slave depending on which Remote menu (Controller menu or Slave menu) was last selected. The Controller menu allows modification of the remote link data rate, selection of the target slave, and selection of the desired slave operation. Once these selections have been made, the operation can be performed. Errors encountered by the controller during execution are

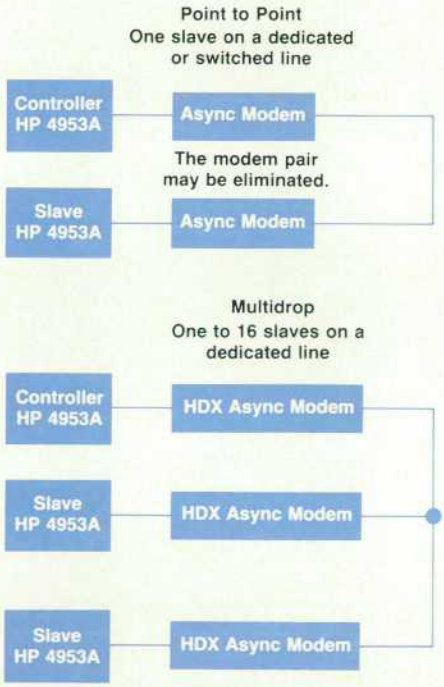


Fig. 6. *Two HP 4953A remote configurations. In the multidrop configuration, a controller HP 4953A can work with up to 16 slaves.*

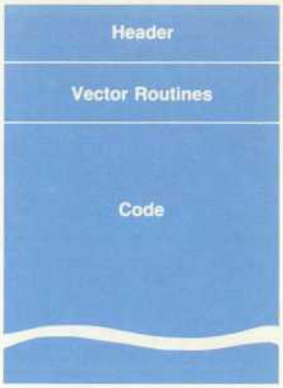


Fig. 7. *Basic structure of an HP 4953A application module.*

reported to the user as system errors. An example of a controller error might be, "The tape is out," following an operation to obtain the slave's mass store catalog.

The HP 4953A is a slave following power-up, a hard reset, or selection of the Slave menu. Once a slave, the instrument will respond to commands from a controller. The Slave menu allows modification of the remote link data rate. In addition, if a user at a slave wants to prevent a controller from modifying the state of the slave, controller operations can be disallowed via the Slave menu. The slave will then return an error for most operations received from a controller.

Remote Implementation

Remote operations use a half-duplex asynchronous protocol that guarantees data integrity across the communication link. A transaction sequence is the basic building block for remote tasks. There are four parts to the transaction sequence. First, the controller issues an operation code to the target slave. Second, the controller transfers data to the slave if required by the operation. Third, the slave returns a completion status to the controller. Fourth, the slave sends data to the controller if required by the operation. The first and third steps are required, while the second and fourth steps are optional. The following three examples illustrate these steps.

Operation code	Data to slave	Completion status	Data to controller
Reset remote	None	Operation accepted	None
Upload menus	None	Operation accepted	Menus
Download menus	Menus	Operation accepted	None

The slave can reject an operation following the first or second step. For instance, if controller operations are disallowed by the slave and the controller initiates a download menus operation, the slave need not wait for the menus (step 2) to be received before it rejects the operation. On the other hand, the slave could accept the menus (step 2) only to find the menus are not compatible with it. If an operation is rejected, the reason is always sent to the controller in step 4. The reason is then presented to the user at the controller.

Some remote tasks, such as the examples above, require only one transaction sequence. Other tasks, such as upload-

ing an application module, require several transaction sequences. This allows even complex tasks to be implemented using simpler operations.

Applications Software

The data communications test market is a very difficult one. The market is very fragmented, with each communications equipment vendor using a special protocol or variation of a standard protocol. To make matters worse, often a number of slightly different versions of a protocol exist because the standard changes over time. This fragmentation makes it impossible to design a completely general-purpose protocol analyzer.

The solution to this problem is applications software. An application is a program that is not built into the firmware of the instrument, but adds a function to the instrument when it is present. By the use of applications, the instrument can be customized for a specific new or different protocol without the cost and difficulty of a firmware change.

If a system is to be capable of supporting applications, some method must be devised to hook the new software modules into the system. Particularly in a softkey-driven system like the HP 4953A, new softkeys must appear when an application is present. Even more important functionally, when those new keys are pressed, the new software must be executed.

A complication peculiar to the HP 4953A is the multiprocessor architecture (see article, page 12). Applications may need to change the functionality of more than one processor. This means that there must be a way in which code brought into the unit through the system processor can be downloaded to and executed on, for instance, the front-end processor.

Fig. 7 shows the basic structure of an application module. The first block of data is a header, which contains information such as the name of the application and security flags. The header also contains four vectors, which are the offsets from the header to four functions within the application. These functions handle loading, deleting, resetting, and executing the module. The reset vector is called when a system soft reset occurs. The execute vector points to the top level of the application program if the application is executable. The load and delete vectors are discussed later.

An application may be executable or not. An executable application is one that can be run from a softkey in the applications menu. A nonexecutable application generally adds functionality to an existing menu, and therefore is not executed directly.

Installing an Application

The problem of integrating applications into the resident firmware is solved through the use of jump tables (see box, page 10). The jump tables contain the addresses of every major routine in the system. The developer of an application simply chooses the functions that need to be changed to display a new softkey or call a new routine and replaces those functions with new versions by changing the entry for each routine in the jump table. Fig. 8 shows the replacement of a function.

The characteristics of system functions have a great deal of impact on the ease of writing applications and the size of the applications produced. Good software engineering practice requires individual functions to be small and to implement a well-defined piece of the solution to some problem. These characteristics pay off handsomely when an application must modify the solution of a particular problem, because fewer and smaller functions have to be replaced.

This is where the load and delete vectors come into play. The load vector points to a routine that changes the jump table entries as needed by the application, and the delete

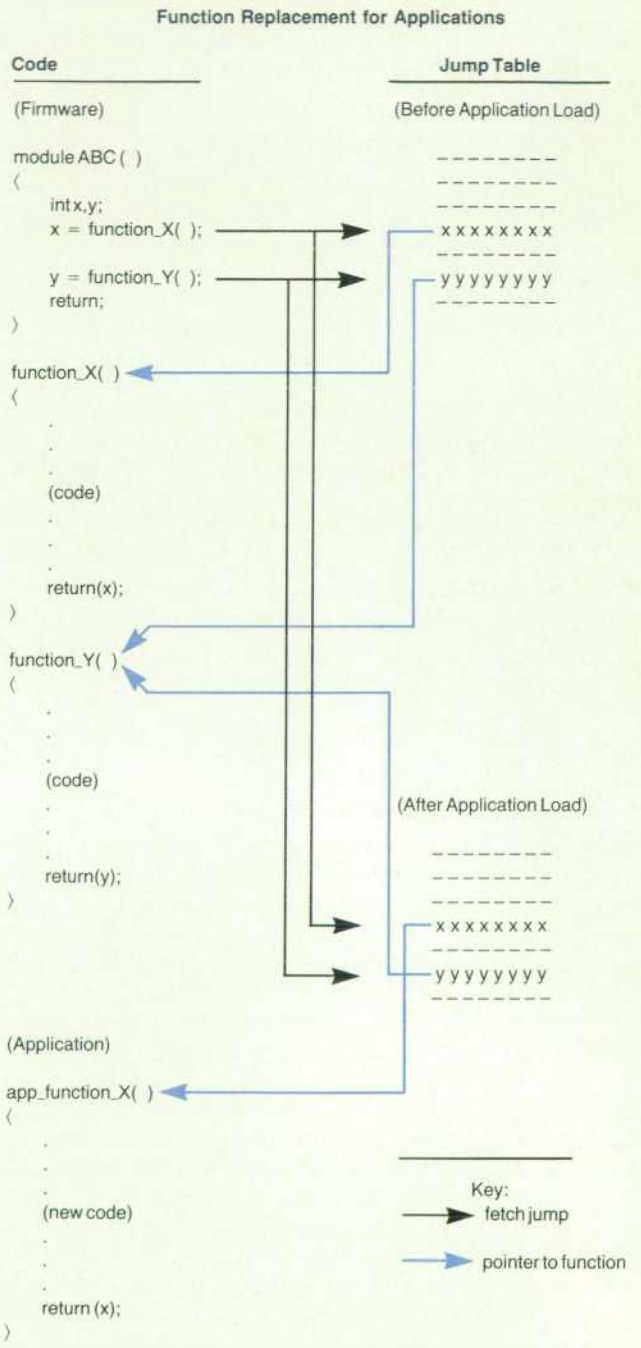


Fig. 8. *Application programs change the functions of softkeys or call new routines by changing entries in jump tables.*

Protocol Analyzer Software Development

Programming in the instrument environment provides opportunities that are somewhat different from programming in a mainframe environment. One of the historic constraints in a ROM-based instrument is the amount of code space available to the software designer, although this is beginning to change as the cost of memory drops. In the HP 4953A this obviously wasn't a severe problem, since we ended up with almost 350K bytes of code, but even so, we were beginning to push the limits of the available space. This space constraint caused the project team to employ slightly different methodologies from those we would have chosen given unlimited space and processor speed.

We realized that we were developing a large software system that would require several software designers. Since the user interface was organized as a set of menus, each of which was relatively decoupled from the others, each designer was given responsibility for a different menu. This resulted in the software being partitioned into a set of separately linked modules. This separate linkage of the modules in the system decoupled the designers from each other and made the possibility of undesirable interactions between modules less likely. In general, each module was in charge of modifying a set of global variables, which other modules would use as values but not modify. As an example, the Setup menu modifies the protocol specific parameters. The Display, Examine Data, and Run menus use these values but do not modify them. The modules range in size from 15K bytes to 45K bytes. As large as some of these modules are, they were much more controllable than a single module of 350K bytes would have been.

At power-on the kernel module calls the performance verification module, which does a thorough test of the hardware. Assuming that the hardware is healthy, the performance verification code returns control back to the kernel which then calls the initialization routines located in each of the modules. Once the machine has been initialized, the main-level display is presented and a user response is awaited. When the user decides to enter a particular menu, control is transferred to the appropriate module. Within the menu, the user can modify the parameters associated with that menu. For example, modifying Display menu parameters changes the way the Examine Data or Run menus present the data to the user. The changes to the data base are usually under softkey control and protect the user from mistakes. A high level of help is provided through this softkey interface, since each softkey selection automatically relabels the softkeys presenting the user with the next set of selections.

Since all of the software designers had a need to access the I/O devices of the HP 4953A, such as the display and the keyboard, a standard library was implemented. All of the user interface software was implemented in the C programming language, so it made sense to provide a C standard library for the use of the software designers. Because of the position dependent nature of the user interface menus, a superset of printf and scanf was implemented that allowed cursor and attribute control of the display. Since the HP 4953A allows the user to operate in several data codes, such as ASCII and EBCDIC, scanf was modified to allow the input from the keyboard to be returned in the requested data code. The implementation of the standard library allowed the software designers to gain command of the I/O functions in the HP 4953A very quickly, since the commands closely follow the conventions established by C.

The design team acknowledged early in the project that a system of the size we were developing was highly unlikely to be error-free, so the team implemented jump tables through which all routines are accessed. These jump tables are located in an EPROM by themselves, the theory being that, if and when bugs are found, the jump vector of the errant routine can be replaced with a vector to a replacement routine located in the space left in the jump table EPROM. In this way, a field update can be effected by replacing only the existing EPROM with a new one that contains the updated jump table and the patched code.

William Grant Grovenburg
Development Engineer
Colorado Telecommunications Division

vector points to a routine that reverses these changes. The one major liability of this system is its inability to handle more than one application changing a given jump table entry at a time. Obviously, if application A is expecting its copy of function X to be executed, while application B, loaded later, also changes the entry for function X, application A will be gravely disappointed. The system prevents this conflict by not allowing the loading of a second application conflicting with one already present.

Applications that need to modify the way that data is brought into the analyzer must change the code on the front end, which uses a separate processor. Hooks are provided by the front-end program to allow the application code to be downloaded into the front-end processor's memory. This code is then accessed through a command table, also downloaded into the front-end processor's memory, which replaces the existing table. The command table is very similar to the jump tables on the system side. The new command table may access any functions normally resident in the front end as well as new functions defined by the application code.

The Application Menu

The user interfaces with the applications handler of the HP 4953A in the same way as with any other function of the instrument: through a menu. The Application menu looks like Fig. 9. There is a catalog of the applications that are loaded, showing the name, size, and description of each. The user can scroll through the catalog using the cursor keys. At the bottom of the screen are memory use statistics, showing the total size of the applications loaded, the available space, the data buffer size, the largest block of contiguous space available, and the number of blocks of data present in the data buffer.

Softkeys available in the Application menu depend on the application selected by the cursor in the application catalog display. The Load application, Buffer size, and Exit keys are always present. If the selected application is storable, a Store application key appears. If the application can be deleted, a Delete application key appears. If the application is executable, an Execute application key appears. In line with the general philosophy of the instrument, invalid choices are not presented to the user.

The mass storage and remote facilities of the HP 4953A

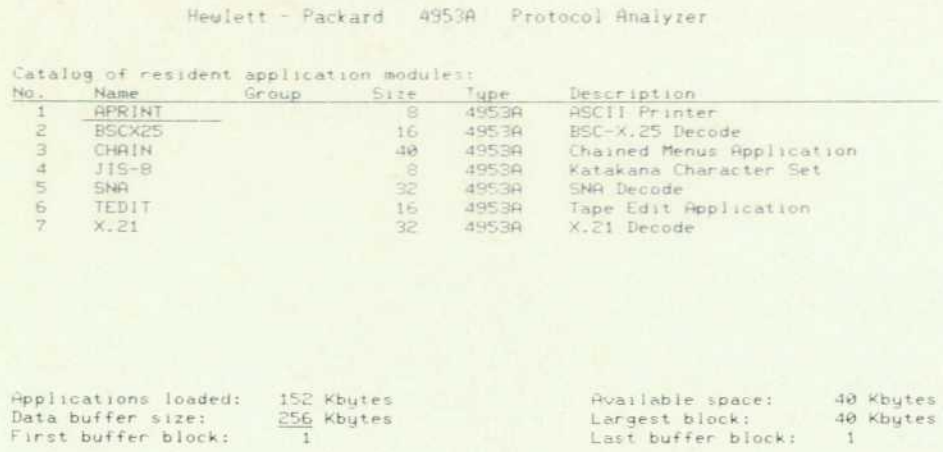


Fig. 9. *HP 4953A Application menu.*

are used for applications storage and transfer. Applications may be stored and loaded from tape and uploaded or downloaded to or from a remote unit. A simple security scheme is implemented on tape or remote transfers. An application may be storable, nonstorable, or a master application. A master application can be copied, but its copies cannot be copied. Most application programs sold for the HP 4953A are masters. This security scheme is maintained by the tape editing program that comes with each HP 4953A.

Memory to store applications is traded off with memory for the data capture buffer. In an HP 4953A with Option 01 (expanded memory) there is a total of 448K bytes to be divided. The data capture buffer must be a minimum of 16K bytes, leaving 432K bytes for applications. If the maximum data capture buffer size of 256K bytes is needed, 192K bytes is left over for applications.

Performance

Performance in a protocol analyzer is a function of several factors. First, there is front-end speed, or how fast the machine can receive and transmit data from the data communications line under test. The HP 4953A can communicate at data rates up to 72,000 bits per second full-duplex and up to 256,000 bits per second in a monitor-only mode with external clocking for bit-oriented protocols (BOPs). The critical section for front-end speed is the data link controller (DLC, see article, page 18). The DLC is responsible for converting the bit stream on the data communications line into the characters for character-oriented protocols (COPs) or into the frames associated with BOPs. Obviously, the front end must keep up with the bit rate of the data communications line under test, or all is lost. For the DLC to simulate or emulate, it must be able to drive the data communications line at speed. It is also desirable to be able to maintain the data rate for an indefinite period of time. The ability of the machine to drive a data communications line without pausing between chunks of data is known as utilization. In the HP 4953A there is a relatively constant time between the end of one chunk of data and the beginning of the next. This causes the utilization for small strings of data (10 characters) to vary from 95% at 1200 bps to 99.8% at 72 kbps. For large strings (100 characters), utilization varies from 99% at 1200 bps to 99.9% at 72 kbps.

Another measure of performance for a protocol analyzer is the number of simultaneous triggers that may be active at once and the complexity allowed in specifying those triggers. The HP 4953A allows up to 63 trigger characters. The 63 characters may be treated as 63 separate triggers, each of which specifies a different action when found, or may be combined to form larger complex trigger strings up to 63 characters in length. For each character in a trigger, the HP 4953 has the ability to test against the complement (NOT) of that character or mask out bits of the character. The HP 4953A can also trigger on special characters such as start and end flags, frame check sequences (FCS), and framing or parity errors.

Two decisions made early in the HP 4953A development were that it would run as close to real time as possible and that it would not allow any data to be lost. This had a very definite effect on the architecture of both the hardware and the run-time software. Often protocol analyzers collect data from the data communications line and put it into a large memory buffer where they can analyze the data at their own speed. This is called postanalysis, and it has one side effect: it allows the point of analysis to fall behind what is happening on the line. To force the HP 4953A to keep up with real time, the point of analysis has been moved out of the data buffer into the data stream between the DLC and the data buffer.

It is important that the protocol analyzer quickly respond to a trigger event, perform the actions required as a result of finding the trigger, get set up for the next trigger, and start analysis again. A small FIFO (first in, first out) buffer is inserted at the output of the DLC to allow the analyzer time to set up and get running again after it has stopped as a result of having found a trigger. This brings the difference between real time and the point of analysis in the HP 4953A to at most the size of the FIFO plus a character or two of buffering in the DLC. Staying close to real time gives the HP 4953A the ability to respond to events on the data communications line very quickly.

Acknowledgments

Richard Zulch was responsible for the architecture of the application programs. Betsy Perkins made a major contribution to the software quality assurance effort.

Simple Architecture Provides High Performance for Protocol Analysis

by Stephen H. Witt and Roger W. Ruhnow

THE HP 4953A is a general-purpose high-performance protocol analyzer that is capable of monitoring serial data streams at 256 kbps and simulating serial data streams at 72 kbps. The HP 4953A is also capable of recognizing data patterns in the data stream and responding to these patterns with a variety of menu functions. A primary design consideration for the HP 4953A was to perform as a real-time monitor. This means the analyzer must process, display, and store measurement data without falling behind the activity occurring on the network under test. A second concern was cost. The instrument is intended for general-purpose use and must be affordable. A third consideration was size and weight. The analyzer must be portable. Therefore, a simple system architecture was chosen, one that would be inexpensive and portable, and would perform well.

The protocol analyzer interfaces to the line under test via an interface pod. The pod provides the physical and electrical connections between the Data Terminal Equipment (DTE) or the Data Circuit-Terminating Equipment (DCE) and the HP 4953A Protocol Analyzer (see Fig. 1).

There are six major blocks in the system, as shown in the block diagram, Fig. 2:

- System processor
- System memory
- System input/output
- Trap machine
- Front-end processor
- Power supply.

The system processor, system memory, and I/O assemblies make up a special-purpose computer that interfaces to the front-end processor and the trap machine. The front-end processor provides the necessary control of the interface to the network under test. The trap machine provides data analysis (triggering capability) of the data being acquired.

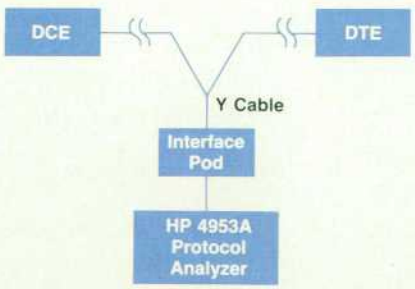


Fig. 1. *The protocol analyzer interfaces to the line under test by means of an interface pod.*

System Processor

The system processor provides processing and control for the HP 4953A. All other blocks in the system communicate with the system processor via memory mapped I/O (see system processor address map, Fig. 3). The system processor consists of an 8-MHz 68000 processor (CPU), program memory, address decoding, timers, DMA control, and interrupt control. Most of the system bus interface signals originate on the system processor. Board enable signals (to communicate with the other blocks via memory mapped I/O), the system reset signal, the system clock (16 MHz), the bus error signal, and the data transfer handshaking signals are generated by the system processor.

System Memory

The system memory consists of the ROM array (384K bytes) and the RAM array (standard 128K bytes or optional 512K bytes). Both the ROM and the RAM are 16 bits wide. This is system memory that is directly addressable by the 68000 and is not accessible to any other processor. The

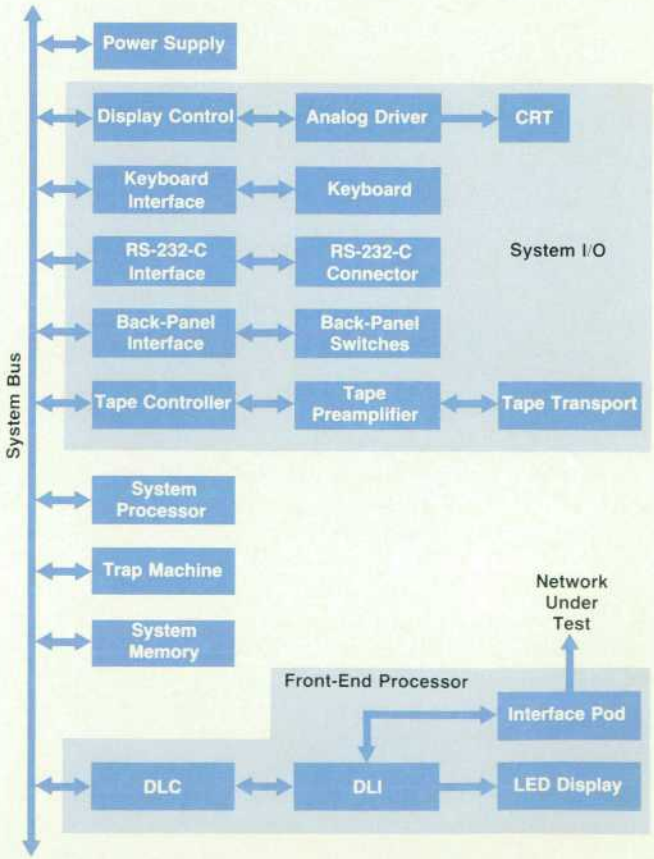


Fig. 2. *HP 4953A Protocol Analyzer block diagram.*

ROM array contains all the system software.

The RAM array is segmented as follows:

- System RAM: 64K bytes
- Data capture buffer: 16K to 256K bytes
- Application program space: 0 to 432K bytes

The system RAM is fixed in size. The data capture buffer and the application program space sizes are designated by the user through software control. More details of the operation of the data capture buffer and the application program space can be found in the articles on pages 18 and 4, respectively.

System I/O

All user interaction with the HP 4953A is accomplished via the system I/O, which consists of the keyboard and keyboard interface, the CRT display, the RS-232-C interface, and the system mass storage tape subsystem. It is only through these I/O interfaces that the user can alter the configuration of the system.

The keyboard is used to control the operation of the instrument through menu entry and interaction with data. The keyboard I/O section includes a full ASCII keyboard with eight softkeys mounted to the front of the instrument with hinges that allow for pivotal movement. It interfaces to the system processor through the keyboard interface circuitry, which interrupts the processor when a key is pressed.The system processor then reads the keyboard register to determine which key was pressed.

The RS-232-C interface exists for two purposes: to access a printer and to provide remote control. The system processor communicates with the RS-232-C interface through memory mapped I/O registers. Back-panel switches are used to set the address of slave units in a controller/slave remote control environment. They are also used to select a baud rate to which the instrument defaults at power-on. After power-on, the baud rate may be changed via softkey selection.

Fig. 3. *HP 4953A system address map. The 68000 system processor communicates with the other blocks in the system via memory mapped I/O.*

The tape section of the I/O subsystem is made up of the tape controller, the tape preamplifier, and the DC100 tape transport. The tape system provides storage of a user's menus, buffer data, run data, or application programs. Data is transferred from the tape to the system processor or vice versa using direct memory access (DMA). The 68000 deals in logical blocks of data referred to as records. It sends the tape subsystem commands to control these blocks of data.

The tape controller is made up of a system bus interface to the system processor, an 8039 processor, a state machine, a servo control loop, and the tape preamplifier. The tape preamplifier buffers the low-voltage signals between the tape transport and the tape controller. The DC100 tape transport assembly consists of the motor assembly, the read and write heads, the hole detection mechanism, the hardware for holding the tape cartridge, and the hardware for ejecting the tape cartridge.

The display is made up of the digital controller assemblies, the primary display driver (high-voltage board), the secondary display driver (sweep board), and the 9-inch CRT display tube.

The CRT controller assemblies provide all of the digital mapping from the display memory in the system processor address space to the analog circuitry that controls the electron beam for energizing the CRT tube. The display memory is accessed by the system processor in a normal 68000 write mode via memory mapped I/O.

The display uses a raster-scan system and is character based. The screen can hold 25 lines of 80 characters each for a total of 2000 characters per display. The CRT controller can implement both normal and large ($2\times$ in each dimension) characters. The normal size characters use 7×9 dots in a 9×15-dot character cell. Large characters use 14×18 dots in an 18×30-dot cell.

Four character sets are available for the display: ASCII, EBCDIC, hexadecimal, and specials. The special character set provides datacom characters (e.g., start flags, stop flags, frame check sequences) and line drawing characters (for timing diagrams and display borders). The user may define a character code set that uses the ASCII character set for its printable characters. The system processor performs the character-set mapping function from a table loaded from tape or keyboard. This mapping function is transparent to the CRT assembly. If no printable character exists for a code, the hexadecimal character is displayed.

Character attributes include half bright, underline, inverse video, cursor (alternates between inverse video and normal video at 4 Hz), overline, and blinking (the cell is blanked at 2 Hz).

Front-End Processor

The front-end processor is made up of the data link interface, the data link control, the LED display, and the interface pod. These assemblies interface to the system processor as one entity through the data link control. The data link control (DLC) provides the intelligence required to

(continued on page 15)

Protocol Analyzer Power Supply Design

The internal power requirements of the HP 4953A are +5V at 10A, +12V at 4A, −5V at 0.1A, and −12V at 0.1A. The +12V output runs the display, the tape drive, and the fan. These subsystems place certain constraints on the supply. For instance, the display is sensitive to small voltage variations, while the tape motor and the fan have large current surges. The power supply also had to pass VDE level B for EMI and the design time needed to be short.

Given the above constraints, a switching type supply was designed with linear regulators on the +12V, −12V, and −5V outputs. Three-pin regulators are used on the −5V and −12V outputs. A remote-sensed regulator is used on the +12V output instead of isolating the display supply with a separate regulator. This minimizes the number of outputs that need protection. A low-dropout (1V) regulator is used to increase efficiency.

The switching circuit (Fig. 1) is a forward converter operating at 50 kHz. A forward converter transfers power to the output when the switching transistor is on. A transistor and FET combination is used for the switch: a low-voltage FET switches a high-voltage bipolar transistor in the common-base configuration. This yields lower cost and is more rugged that a single bipolar transistor or FET.

The main switching transistor is protected by a comparator that senses the instantaneous current in the transformer primary. If the current exceeds a preset value, the switching pulse is turned off early. This limits the current during an overload condition. If the overload is sustained for more than 50 ms, the supply turns off and remains off. The ac power must be turned off for at least five seconds to resume normal operations when power is reapplied. The supply will also shut down if any output is 10% over its specified voltage for more than 10μs. The +12V, −12V, and −5V regulators have individual current limiters. However, power dissipation in the regulators is large when they are current limiting. To prevent damage, the supply shuts down if any of these outputs is 10% under specified voltage for 50 ms. These time delays ensure that turn-on and load surges will not trip the protection circuitry. The protection circuitry is fabricated on a thick-film hybrid to reduce cost and save board area.

Normally in a power supply of this type, filter networks called snubbers are used on all of the transformer windings to reduce voltage transients that could damage the semiconductors. These networks are eliminated by using bifilar windings on the transformer's primary and flyback windings, and by using the most recent technology available for the rectifiers.

Stephen M. Ernst
Development Engineer
Colorado Telecommunications Division

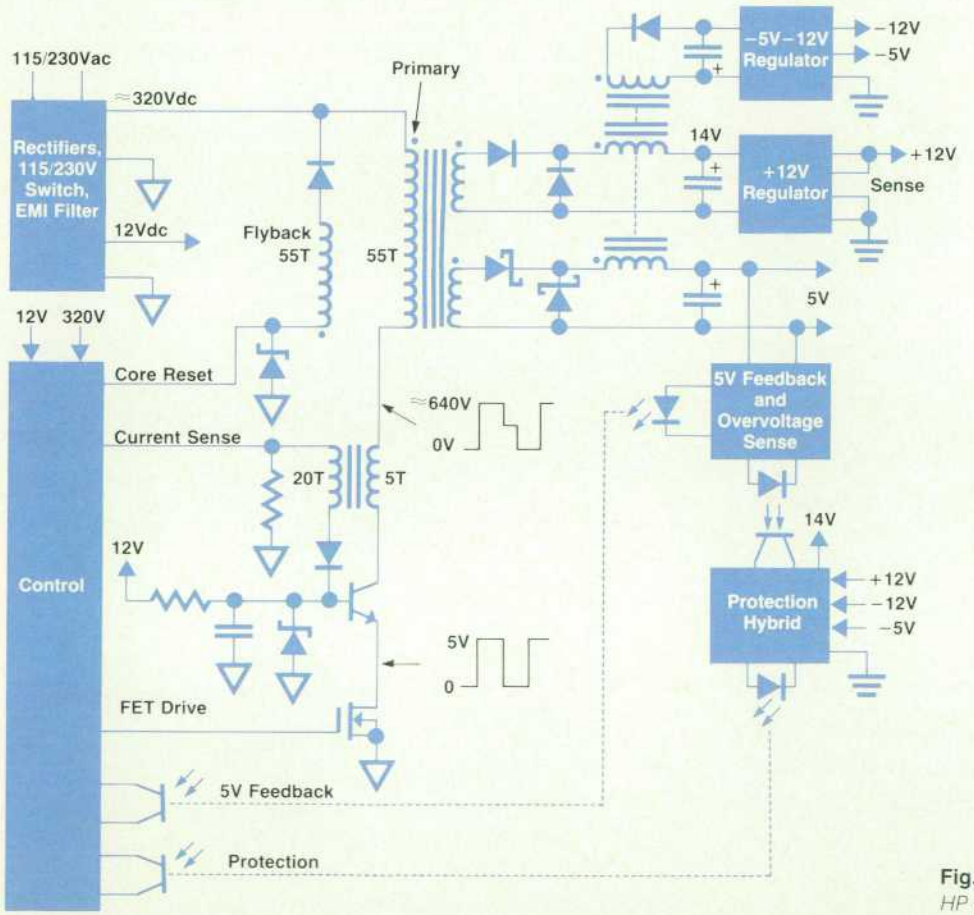


Fig. 1. *Schematic diagram of the HP 4953A power supply.*

(continued from page 13)

acquire data, translate data, provide timing, provide data, and perform all other tasks necessary to interface with a user's network. The data link control is made up of two microprocessors operating in parallel: a Z80 and a Z8. The Z80 has 32K bytes of EPROM program memory and 16K bytes of RAM storage. It executes the protocol code. The Z8 has 2K bytes of ROM and 256 bytes of RAM, both internal to the part. The Z8 handles the interface lead status and the timing functions, and puts the data in the capture buffer. The Z8 is called the timing processor. Details of the front-end design are in the article on page 18.

Trap Machine

The trap machine scans the DLC data from the HP 4953A front end for user-programmed character sequences, lead status changes, and errors or specials (conditions not specified by other data types) in data link protocol. The trap machine interfaces to the system bus and controls DMA transfers from the data link control to the system memory. Data patterns are scanned as they "pass by" on the system bus. On finding a trigger sequence, the trap machine reports trigger information to the system processor.

The trap machine has a mask field and an instruction field associated with each trigger character. The mask field allows "don't care" conditions to be specified on a bit-by-bit basis. The instruction field is used to specify trigger type and NOT conditions in addition to containing some reserved fields for system use. There are 64 available triggers, one of which is reserved for system use.

The trap machine keeps track of the time information and the interface lead status information put into the data capture buffer so that the condition of the interface can be reported to the system processor when a trigger is reported.

The trap machine consists of a C2000 gate array, a few TTL integrated circuits for system interfacing, and three NMOS RAMs.

In this description, several references are made to DLC data types. These formats are the same as for buffer memory and are described in the article on page 18.

Trigger Definition

A trigger is a sequence of characters or elements that the user wants to match in data received from the DLC. A user defines triggers in the Monitor and Simulate menus to direct run-time execution. Each element of a trigger consists of three bytes: a character byte, a mask byte, and an instruc-

Protocol Analyzer Mechanical Design

The HP 4953A Protocol Analyzer uses standard HP cabinetry, but is the first 7-inch-high, full-width cabinet in a 10.6-inch depth used in the corporation. This nonstandard depth was chosen because the HP 4953A is frequently used as a field service instrument. As a result, low weight and a size capable of fitting under an airplane seat were important constraints.

The nine 6×9-inch digital boards load from the top of the HP 4953A into a bottom-mounted backplane and run parallel to the front of the instrument. For serviceability, the boards can sit atop an extender card and face the front for easy access to either side of the board. The two-piece connectors are staggered across the backplane so that it is physically impossible to mate a digital board into the wrong slot. The two CRT driver boards, which, with top and bottom covers removed, hinge out of the instrument for easy servicing, and the display tube consume nearly all of the remaining volume inside the box. This leaves insufficient room for the power supply. Consequently, the power supply mounts vertically inside the rear panel, which protrudes out the back of the instrument. The standard rear feet were therefore too short to allow operation of the instrument while standing on its rear feet with the necessary interface cabling connected to the rear panel. Consequently, a rear foot was tooled that is one inch longer than standard to provide the necessary additional ground clearance.

With no room on the rear panel for the necessary cooling fan, the fan was mounted on the side. It draws ambient air through the right side cover across the digital boards and power supply, and exhausts it out the left side.

On the front of the instrument is a full, nondetachable, fold-down keyboard lockable at any desired angle. The front panel has a 9-inch monochrome CRT, a line switch, a DC100 tape transport, and a bank of 44 LEDs. The LEDs, 22 red and 22 green, indicate the status (on and off or mark and space) of the interface lines of the system under test. Because there exist several interface standards throughout the world, each with differing numbers of lines and differing nomenclature associated with each line, a simple means of changing the number and names of these lines' status indicators was needed. An overlay card that snaps into the front panel over the LEDs was designed for each interface. Only the correct number of LEDs can be seen through each overlay, each with its proper nomenclature.

The hinge mechanism on the fold-down keyboard extended below the plane of a standard HP bottom foot, which snap-mounts onto the bottom cover. Other feet used throughout the company were inappropriately tall for use on the bottom of the instrument. As a result, a new bottom foot, which uses a fold-down tilt stand similar to the one used with the standard bottom foot, was tooled. This new foot mounts to the front frame instead of the bottom cover.

The HP 4953A's handle is a formed, rigid metal strap with a molded hand grip. It mounts to the sides of the front frame. A side strap recessed into the side panel (typical on many HP instruments) was rejected for two reasons: first, it cut down the available path for airflow through the cabinet by nearly 50%, and second, the form factor for carrying the instrument in this orientation made it awkward and uncomfortable.

The field service environment in which the HP 4953A is frequently used demands that it be especially rugged. The instrument was tested to HP's new class B-1 standards for shock and vibration. No failures were found during either resonant vibration dwelling, or during random vibration testing. In shock tests, the instrument withstood at least 56g before experiencing unacceptable damage. Even at these shock levels, the damage was to the outer structure and did not affect the proper operation of the instrument.

Ken Krebs
Development Engineer
Colorado Telecommunications Division

tion byte. The mask byte contains the bit pattern the user wishes to AND with the received DLC character before comparing it to the character byte. An example of masking would be the elimination of the ASCII parity bit. The character byte contains the byte the user wants to use as a trigger element. The instruction byte contains the trigger type, comparison criteria, and flags indicating if this element is the start of a trigger, the end of a trigger, and/or the last trigger element to be considered before obtaining a new DLC data character. The comparison criteria possible are equal and not equal.

The three NMOS RAMs hold the trigger triplets. Sixty-four triplets are available, 63 for the user and one for the system. These can be partitioned into a number of triggers, each having a variable number of elements. If all of the user's elements are used, there can be from one trigger of 63 elements to 63 triggers of one element each. Triggers are entered into RAM from the lower addresses through the higher addresses. The trigger byte is entered into the character RAM, the mask byte is entered into the mask RAM, and the trigger type, comparison criteria, and trigger flags are entered into the instruction RAM. If the triplet is the first in a trigger, the start flag is set in the instruction RAM. If the triplet is the last in a trigger, the end flag is set in the instruction RAM. If the triplet is the last defined in RAM, the last flag is set in the instruction RAM. A trigger can be defined but rendered inactive by not setting its start flag. These RAMs are read from and written to by the system processor and are accessed by the gate array while the trap machine is running.

Search Algorithm

When searching for a programmed trigger, the trap machine uses the concept of a link bit to indicate the progression of a trigger sequence. A link bit is one bit of information associated with each trigger element. By definition, a link bit associated with a trigger element is set if and only if a sequence has been found in the DLC data that matches the trigger sequence through that trigger element. For example, if a trigger consists of trigger elements DTE A, DTE L, and DTE M, and the data received from the DLC is DTE A and DTE L, the link bit associated with the trigger element DTE L would be set. As the trap machine continues to receive DLC data that matches the trigger sequence, the link bit will move (or propagate) from trigger element to trigger element until it propagates to an element with the end flag set. In this case the trigger has been satisified.

There are several restrictions on the propagation of a link bit. If the type of the received DLC character is not the same as the trigger type, the link bits for that trigger are not modified. This property allows a trigger sequence to be unaffected by DLC data types other than its own. If the trigger type is the same as the received DLC character then the propagation of the link bit is controlled by additional considerations.

The mask byte is ANDed with the DLC character and the result is compared with the character byte. This result must match the comparison criteria specified in the trigger element's instruction byte. If these criteria are not met, the link bit is set to zero, terminating the propagation of any link bit by that element. This occurs when the DLC data sequence fails to match the trigger sequence through that element. If the comparison criteria are met, the link bit will be set if and only if the link bit of the previous trigger element was set before its last evaluation, or this element is the start of a trigger (its start flag is set).

As an example of the above rules, consider Table I. The top row is a programmed trigger sequence consisting of the DCE ASCII characters A, B, A, B, C, and A in that order. The leftmost column shows the data sequence received from the DLC, consisting of the DCE ASCII characters A, B, A, and B, DTE ASCII character C, and DCE ASCII characters C and A, in that order. Since the DTE character is not the same type as the trigger elements, it is ignored and is essentially removed from the DLC data sequence. If this is done the DLC data sequence appears identical to the trigger sequence, so a trigger is present. The table is the set of link bits associated with the trigger elements. The column of ones and zeros under a trigger element shows the link bit for that element. The link bits are initialized to zero before the searching algorithm begins, as seen in the first row. While a DLC character is being compared to the trigger sequence(s), the link bits are changed to the values shown in the corresponding row of the DLC data word. When the link bit becomes set for the last element in the trigger sequence, a trigger has been found.

As mentioned, the link bits must first be set to zero to prevent false triggers, since they represent propagating trigger sequences. The first DLC data word is DCE A. This word is compared to all of the trigger elements in order from left to right. The first element is DCE A. Using the conditions for link bit propagation, it is easy to see that only the link bit associated with DCE A will become set.

The second DLC character is DCE B. As this word is sequentially compared to the elements in the trigger, it can be seen that only the first trigger element DCE B satisfies all conditions, so its link bit is set. The third received DLC data word is DCE A. As this word is sequentially compared to the elements in the trigger, two link bits are set. The second link bit becomes set because the trap machine rec-

Table I
Propagation of link bits through a trigger

			Start of programmed trigger sequence					End of programmed trigger sequence	
			DCE A	DCE B	DCE A	DCE B	DCE C	DCE A	
DLC Data Sequence ↓			0	0	0	0	0	0	Initial link bits
First	DCE	A	1	0	0	0	0	0	No trigger found
	DCE	B	0	1	0	0	0	0	No trigger found
	DCE	A	1	0	1	0	0	0	No trigger found
	DCE	B	0	1	0	1	0	0	No trigger found
	DTE	C	0	1	0	1	0	0	No trigger found
	DCE	C	0	0	0	0	1	0	No trigger found
Last	DCE	A	1	0	0	0	0	1	Trigger found

Making a Protocol Analyzer Producible and Serviceable

Producibility and serviceability are important issues in any product design. To achieve both goals, it is important to have effective performance verification software, which indicates whether the instrument is functional, and enough additional hardware and software to allow any failures in the instrument to be located quickly. It is also important that the additions to the instrument needed strictly for performance verification and troubleshooting be kept to a minimum to keep production cost as low as possible. For example, to service the HP 4953A hardware, no electronic tools or extender cables are needed. All of the boards are readily accessible and each has been designed to be easily repaired.

Test points to be used for signature analysis (SA) troubleshooting are located at the top of each board to provide for easy hookup of the signature analyzer probe when the boards are in the cardcage. The test points consist of two rows of X M-post-type connectors (where X is the number of test points), rather than the conventional single test points. Using the connectors, SA probes can be attached quickly without using the clips supplied with the SA probe. The double row of test points allows more than one connection to be made to each test node. In addition, using connectors for test points allows all of the test points to be loaded at one time, thus reducing production cost.

M-post-type connectors are also used to implement the many jumpers required to break feedback loops for SA troubleshooting. Using the connectors instead of the more common DIP jumpers conserves board space and makes it much clearer which jumper position is normal and which position is for troubleshooting.

Over seventy performance verification and troubleshooting tests were written for the HP 4953A. The same code is used for performance verification and signature analysis as much as possible. This reduces the amount of code needed and avoids the problem that sometimes arises when performance verification detects a problem in a circuit, but separate SA software stimulates the circuit in a different manner, so the failure cannot be located.

The majority of the tests are executed each time the instrument is powered-on and take less than eight seconds to complete. Tests that require user interaction, such as the tape test, are accessible through a top-level softkey.

The tests are numbered in the order they are executed. Since testing begins at the heart of the instrument and works outwards, troubleshooting begins with the failing test having the lowest test number. Any test can be individually cycled for SA troubleshooting or for finding intermittent failures by setting a switch located on the CPU board to the desired test number and pressing the reset button. Switch settings are also available for cycling particular groups of tests or for cycling all of the tests and outputting the results to a printer.

Most test failures are displayed on the CRT. However, before the CRT can be used to display failure information, there must be some confidence that the CRT circuitry itself is functional. Therefore, the power-on tests are divided into two sets, the first dedicated to testing the circuitry needed to access the CRT display. If any of the tests in the first set fails, the failure is indicated by repetitively displaying the test number of the failing test on a seven-segment display located on the CPU board. Most of the remainder of the HP 4953A is tested by the second set of power-on tests. Any failures in this set of tests are displayed on the CRT.

John R. Rader
Development Engineer
Colorado Telecommunications Division

ognizes that another trigger sequence is being received with this character as the first element. This property is desirable in the detection of overlapping triggers. Many link bits in a trigger can be set at any given time.

An interesting item is the reception of the DTE C character from the DLC. The link bits should not be changed with the reception of this character, so the trap machine simply copies the link bits, as indicated in the table. The trigger is finally found when the link bit in the last DCE A trigger element becomes set.

It is possible to be searching for more than one trigger at a time. This is accomplished by sequentially comparing the DLC character to all triggers before requesting another DLC character. When considering multiple triggers, link bits from one trigger do not affect another trigger. Table II shows an example of two triggers.

Data Flow Description

Data is provided to and acquired from the network under test by the interface pod. The network data consists of serial data, clocks, and interface control leads. The interface pod recognizes voltage levels and changes on the interface and converts these voltages into TTL levels that can be interpreted by the digital hardware in the HP 4953A. The interface pod interfaces to the data link interface (DLI). The DLI interprets the data, clocks, and leads and displays this information by writing to the LED assembly on the front panel.

The DLI detects interface lead status changes and passes this information along with the data and clocks from the interface pod to the data link control (DLC). The DLC converts the serial data into an 8-bit parallel format, keeps

Table II
Propagation of link bits through two triggers

			Start of trigger 1		End of trigger 1		Start of trigger 2		End of trigger 2
			DCE A	DCE B	DCE A		DCE B	DCE A	
DLC Data Sequence ↓			0	0	0		0	0	Initial link bits
First	DCE	C	0	0	0		0	0	No trigger found
	DCE	B	0	0	0		1	0	No trigger found
	DCE	A	1	0	0		0	1	Trigger 2 found
	DCE	B	0	1	0		1	0	No trigger found
	DTE	C	0	1	0		1	0	No trigger found
Last	DCE	A	1	0	1		0	1	Triggers 1 and 2 found

track of 16 bits of relative time, and maintains the status of 15 interface control leads. Time stamps and information identifying the data type (see "Data Capture Buffer," page 22) are added to the 8-bit data byte, forming a 16-bit word. The DLC recognizes specific protocol events on the network under test and identifies these by generating "specials" for insertion in the buffer. The 16-bit words are passed from the DLC to the system memory via a 16-bit DMA channel that is maintained by the system processor.

The trap machine controls the flow of data from the DLC to the system memory by acknowledging the DLC's DMA requests to the system processor. The trap machine reads the data during each DMA cycle as it is transferred to the system memory. The trap machine is programmed to detect user-entered trigger sequences. When a trigger sequence is detected, the trap machine interrupts the system processor and reports the trigger. The system processor halts the trap machine, executes the appropriate menu functions, and then restarts the trap machine. During this interval the trap machine is halted and no data is transferred to the system memory. The data passes through a 128-byte-deep FIFO (first in, first out) buffer on the DLC. This FIFO allows for pauses in the data flow to the system memory.

The data is now stored in the data capture buffer in the system memory. The system processor maintains a second 16-bit DMA channel between the system memory and the tape controller. While menu programs are executing, they can control the flow of data to the tape. Run-time data may thus be stored to tape.

The system processor has access to the system memory and can read data from the data capture buffer for display on the CRT. The display RAM on the CRT is accessed through the system processor's address space. The system processor manipulates the data to provide for various display formats and sequences.

For simulation, the system processor creates data strings in parallel format and interface control lead settings from information entered by the user. This information is passed to the DLC where the data is converted from parallel to serial format for transmission on the network. Clocks are supplied and leads are set accordingly. The DLC calculates the necessary error checking sequences and appends these to the data. The DLI turns the LEDs on and off according to the information sent. The interface pod detects the data, clocks, and leads set by the instrument and reports this information via the data path discussed above for monitoring data.

Acknowledgments

Many people deserve recognition for their effort in the development of the hardware for the HP 4953A Protocol Analyzer. The architecture described was defined by Grant Grovenburg, Dave Karlin, and Bob Weisickle. Grant Grovenburg designed the system processor and system memory. Dave Karlin designed the system memory, data link interface, and data link control. Mark Keisling worked on the data link control, data link interface, and system integration. Sharlene Lin and Ed Moore developed the tape controller. Steve Ernst designed the power supply. Ken Krebs provided the mechanical design. John Rader was the service engineer. Jamie Hunt and Nate Sakamoto were the production engineers.

Serial Data Acquisition and Simulation for a High-Speed Protocol Analyzer

by Mark D. Keisling, Dorothy J. Yackle, David B. Karlin, and Elizabeth Gates Moore

THE FRONT END of the HP 4953A Protocol Analyzer is a dedicated subsystem that provides the serial test interface for the 68000 system processor. The front end consists of four assemblies (see Fig. 1): the pod (there are several to choose from), the data link interface (DLI), the data link control (DLC), and the lead status display (LEDs). The pod, located outside the instrument, is connected to the rear panel of the HP 4953A using a two-meter 50-conductor cable. The other three assemblies (DLI, DLC, LEDs) are inside the mainframe.

Fig. 2 is a detailed block diagram of the front end. Two microprocessors in the DLC control all the functions of the front end. It is the DLC that communicates with the 68000 system processor. The DLI assembly functions as an interface between the external pod, the DLC, and the lead status display. The pod connects to the network or device being tested. The lead status display indicates the logic levels of all the interface leads on the pod.

The front end monitors the data and control leads of the device under test. This information is passed from the device through the pod to the DLI. The DLI routes the data to the proper channel of a serial input/output device (SIO) located on the DLC. Working with the front-end firmware, the SIO does some initial analysis of the incoming data and then passes the data and analysis information to the timing processor (a Z8 MCU). The timing processor formats the data into the buffer data format (see "Data Capture Buffer," page 22) and appends timing information to the data. The resulting 16-bit word goes to a 128-word first in, first out (FIFO) memory device, and then to the system

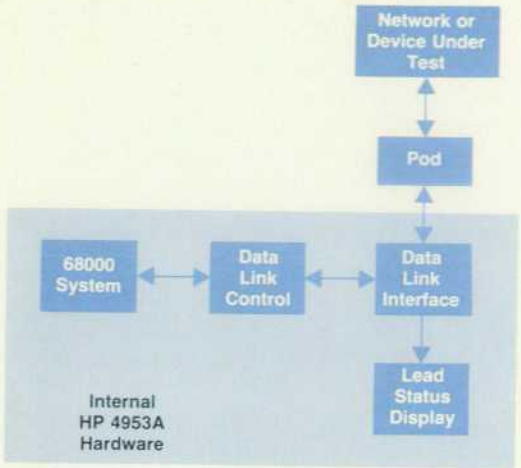


Fig. 1. *HP 4953A Protocol Analyzer simplified front-end block diagram.*

buffer memory via a 16-bit direct memory access (DMA) channel. The lead information is retained by the DLI. Only changes in lead status are reported to the DLC.

The timing processor also maintains a 16-bit status word of the lead states. This status word is sent to the FIFO following the lead change word. The status word is also written to the FIFO following a tickmark word. Tickmarks are generated periodically by the timing processor to maintain timing integrity during the execution of a run. The information sent to the FIFO is eventually sent to the system buffer memory. Lead status information is also latched into peripheral driver devices that drive the LEDs of the lead status display assembly.

The front end can also send data out of the pod to the device under test and can drive the control leads. These simulation capabilities are activated in response to Send Data String and Set Lead commands received from the 68000 system. A data byte accompanying the command specifies which string is to be sent or which lead is to be set to what state. The strings are downloaded from the system to the DLC before the run begins. Upon receiving a Send Data String command, the DLC fetches the string from its memory and passes the string data byte-by-byte to the SIO. The serial data travels out of the SIO through the DLI to the selected data lead on the pod and on to the device under test. Interface clocks are generated on the DLI and sent to the pod and to the SIO. Upon receiving a Set Lead command, the DLC passes the byte containing the control lead information through the DLI directly to the pod, where the chosen lead is set on or off.

The Pod

The pod translates the physical interface signals (e.g., RS-232-C, RS-449) from the interface voltage levels to TTL levels and from TTL levels to the interface voltage levels (see Fig. 3). Up to fifteen interface control leads, four data leads, and three clock leads can be monitored and driven. Each control, data, or clock lead can take on a voltage level within one of three ranges: logic 1 ($V < -V_r$), logic 0 ($V > +V_r$), or inactive ($-V_r < V < +V_r$). V_r is a reference voltage (3 volts for the RS-232-C pod). These TTL signal levels are transferred to the DLI using a multiplexing scheme. Each lead (up to 22) is assigned an address. The control leads occupy addresses 0 through 14. The data and clock leads occupy addresses 16 through 22. Address 15 is not used. As each lead is addressed, the pod responds by indicating which of the three voltage ranges the lead is in. The front-end hardware within the instrument can only determine the state of the control leads using this addressing scheme.

The pod can also drive any of the leads (see Fig. 4). Each of the drivers can be disabled so that the lead looks like a high-impedance load. When enabled, each driver can drive its lead to the voltage level representing a logic 1 or 0. The pod is given a seven-bit command. The lower five bits indicate which lead is to be modified (using the same addressing scheme as above), and the upper two bits indicate

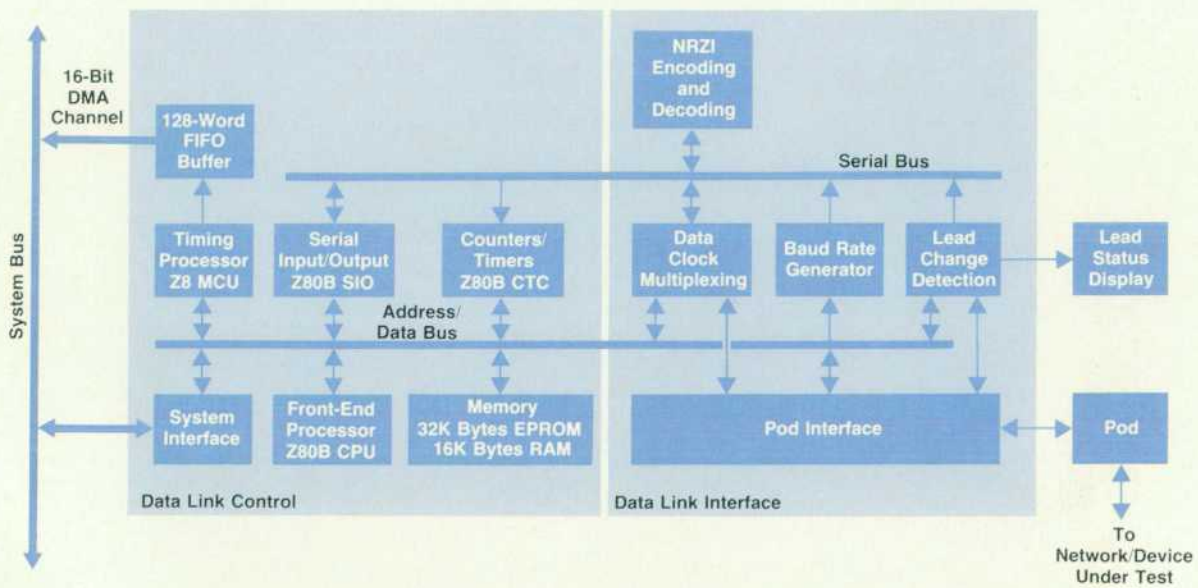


Fig. 2. *Detailed front-end block diagram.*

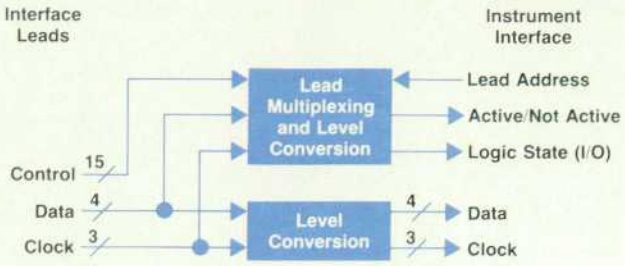


Fig. 3. *Pod receiver functions.*

whether or not the driver for that lead is to be active and, if active, what the logic level of the lead is to be (the logic level indication is not used for the data and clock leads). The pod is also supplied with a serial data signal and a serial clock signal.

All drivers are disabled when the instrument is powered on and whenever the **RESET** key on the keyboard is pressed. The proper drivers are enabled before the run begins. The Stop State menu determines which drivers are left enabled at the termination of a run.

Data Link Interface

The signals from the 50-conductor pod cable are connected to the data link interface using the system motherboard. The DLI assembly consists of five functional blocks: pod interface, baud rate generator, signal multiplexing, lead change detection, and NRZI encoding/decoding.

The pod interface block serves to buffer the signals sent to and received from the pod. All serial data and clock signals are driven differentially for increased noise immunity.

The baud rate generator block consists of a primary baud rate generator and a secondary baud rate generator. The primary generator can be programmed to provide a symmetrical clock of any frequency from 10.0 Hz to 999 kHz. Resolution is three significant digits, and accuracy is 0.005%, which is based on the 12-MHz front-end clock crystal. The secondary generator provides a $\times 16$ clock for asynchronous data communications. The secondary baud rate generator can be programmed to provide any of the following baud rates in bits per second: 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200.

The signal multiplexing block of the DLI provides a means of routing the serial data and clock signals from the pod to the appropriate channel of the serial input/output device on the DLC assembly. The multiplexing block also can route these signals through the NRZI encoding/decoding block. (NRZI stands for nonreturn to zero inverted; it is a serial data encoding scheme that embeds the clock signal in the serial data stream.)

The control lead change detection block keeps track of the current states of the fifteen control leads. Should the state of any of the control leads change, the front-end processor (located on the DLC) is interrupted. The interrupt is provided through channel 0 of the counter-timer circuit located on the DLC. Changes on any of the fifteen control leads can be ignored by clearing bits in two eight-bit registers.

Data Link Control

The DLC assembly contains the front-end processor (a Z80B CPU), 32K bytes of firmware, 16K bytes of static RAM storage, and the timing processor (a Z8 MCU). The DLC communicates with the 68000 system processor via an eight-bit bidirectional port. Interrupts are supported in either direction. The DLC accepts eight-bit commands and data and returns eight-bit status bytes to the 68000 system.

The firmware controlling the Z80 CPU can be changed by downloading new front-end software into the 16K bytes of RAM. The front-end jump table can be modified to make use of this code. This enables applications written for the HP 4953A Protocol Analyzer to customize the front-end firmware.

The DLC controls all of the front-end hardware. Registers on the DLI and pod are memory mapped within the Z80 CPU address space. These registers configure and control the various functions of the front end. The Z80 CPU can also pass commands and data to the Z8 timing processor. The timing processor is responsible for formatting data that is destined for the system buffer memory. The timing processor's firmware is contained in a ROM that is part of the Z8 MCU. This firmware cannot be changed by applications software.

The serial input/output device (SIO) is located on the DLC. The SIO converts data from parallel to serial and from serial to parallel. The SIO supports synchronous bit-oriented protocols (e.g., HDLC, X.25), character-oriented protocols (e.g., Bisync), and asynchronous data transfers. Data is clocked in and out of the SIO at a rate defined either by the baud rate generator located on the DLI or by one or more of the interface clocks from the pod. The SIO has two full-duplex channels. Channel A supports the DTE side of the interface under test and channel B supports the DCE side.

Another device located on the DLC is the counter-timer circuit or CTC. The CTC provides four independent timer/counter channels. Each channel is connected to an external signal. Channel 0 is connected to the control lead change interrupt from the DLI. Whenever a lead change is recognized, this signal makes a positive transition. The channel 0 counter then decrements from 1 to 0, posting an interrupt to the Z80 CPU. Channel 1 is connected to a clock signal from the pod, allowing timing measurements to be made on that clock. Channels 2 and 3 are connected to a 1-kHz

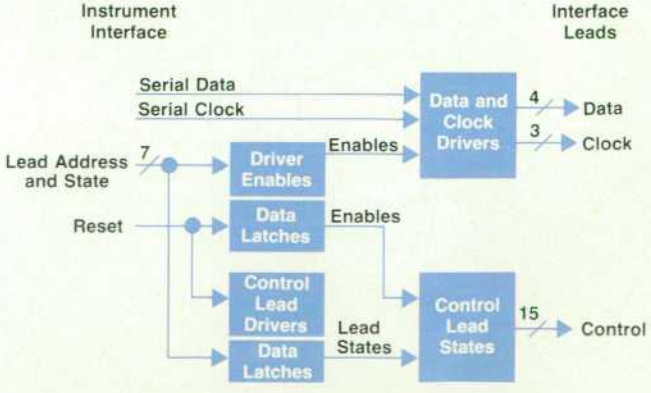


Fig. 4. *Pod transmitter functions.*

clock signal supplied by the DLI. This provides two 1-ms-resolution timers for the front end.

Link-Level Protocol Acquisition

One of the more interesting tasks performed by the front end of the HP 4953A is the dissection of the link-level protocol of the data link under test. This is the lowest level where information transfer can occur, and therefore, byte, frame, message, content, and dialogue synchronization must be achieved here. Usually, higher-level protocols are only concerned with content and dialogue. This implies that the link-level protocol can be viewed as a bridge joining the physical conditions on the link with the higher or logical levels of the data communications path.

Two fundamental types of link-level protocol are supported by the HP 4953A: bit-oriented and character-oriented. BOPs (bit-oriented protocols) are characterized by a common frame structure with byte synchronization determined by the unique eight-bit pattern called a flag (01111110):

| Flag | Address | Control | Information | FCS | Flag |

The address, control, and FCS (frame check sequence) fields are defined relative to the beginning and ending flags. This, along with the flag's uniqueness, yields inherent data code independence and data transparency (any possible bit pattern can be transmitted at any time).

COPs (character-oriented protocols) do not share this inherent transparency. Instead, they are characterized by many control characters and a byte synchronization pattern, which are all dependent upon the data code set used by the data link. Numerous frame or block structures exist, depending on the type of control or information to be sent on the data link:

| SYN SYN SOH {Header} STX {Data} ETB or ETX {Checksum}|
| SYN SYN SOH {Header} ETB BCC BCC |
| SYN SYN SOH {Text} ETX BCC BCC |
| SYN SYN DLE STX {Text} DLE ETX BCC BCC |
| SYN SYN EOT |
| SYN SYN {Address} ENQ |

Functionality of Service Routines

Frame	What is placed in buffer	Other activities performed
Address	(1) Start flag special (2) Address byte(s)	
Control	Control byte	
I Frame	Start of I field special	Accuracy of the N(R) and N(S) counts
S Frame		(1) Accuracy of N(R) (2) Restart count on certain S frames
Data	Data	
FCS	First FCS special Second FCS special End flag special	

Fig. 5. *Functions of the BOP service routines for various types of received frames.*

SYN = Synchronizing character
SOH = Start of header
STX = Start of text
ETB = End of transmission block
ETX = End of text
EOT = End of transmission
ENQ = Enquiry
DLE = Data link escape
BCC = Error detection bits

In fact, not all blocks have error detection bits (BCC) sent with them, and the data code and transparency conditions determine whether parity checking is in use. Basically, the characters received on the data link specify the block structure for that particular block.

BOPs: Routine Swapping State Machine

To acquire bit-oriented frames, many functions must be performed by the front end. These are divided between the serial input/output device (SIO) and the BOP software state machine. Byte and frame synchronization, data transparency, and FCS calculation are all handled by the SIO. Flags can always be detected because of their unique pattern, 01111110. To guarantee that this pattern never occurs in the data stream, a zero is inserted after any five consecutive ones in the data stream. The zero is deleted in decoding the data. The FCS is calculated by the SIO using the CCITT cyclic redundancy check (CRC) polynomial $x^{16}+x^{12}+x^5+1$. The SIO determines the accuracy of the FCS.

The BOP software mainly interprets the bytes of the frame, and then sends the data and special characters to the system capture buffer (see Fig. 5). On the receipt of the first data character following a flag, the ADDRESS service routine is entered. This routine places a Start Flag special character into the buffer, followed by the address byte. It then schedules the CONTROL service routine to be executed upon receipt of the next data character. This character, the control byte, provides the major decision point in the state machine. It contains information to determine the frame type: (I) information, (S) supervisory, or (U) unnumbered. Two sequence numbers in the I-frames, N(S) and N(R), are maintained by the software to provide an autoincrement capability during simulation. To help the display software, a special character is placed in the capture buffer to indicate the start of the information field. Only I-frames contain information fields.

Following the control byte, the DATA service routine executes for all the remaining data characters received before the ending flag is encountered. This routine simply places the data character in the capture buffer and reassigns itself. The end flag, signified by a different interrupt vector, signals the BOP software to check the accuracy of the received FCS. This yields either a good FCS special character or a bad FCS special character placed in the capture buffer. One other type of end-of-frame special character can occur, the Abort Frame character. To abort a frame, a sequence of eight to thirteen consecutive ones with no inserted zeros must be received. In all cases, the end-of-frame character is immediately followed by an End Flag character written to the buffer. Finally, the ADDRESS service routine is scheduled for the start of the next frame.

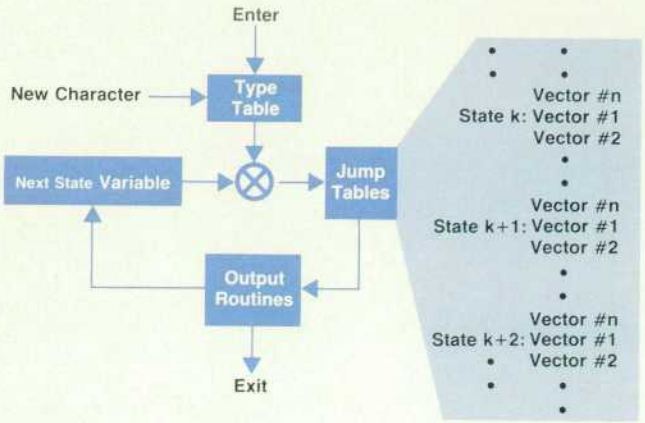


Fig. 6. *COP state machine.*

COPS: Table-Driven State Machine

The requirements for the COP front-end firmware differ somewhat from that for the BOP. Although less information is passed along to the system capture buffer, each block received by the COP software requires quite a bit of processing. The basic tasks performed by this software are:

- Maintain byte synchronization
- Send all valid data and control information to the system capture buffer
- Detect errors based on parity or CRC and report them to the system capture buffer.

To accomplish these tasks, the front end must be able to recognize all block types, independent of data code or transparency. This must be done on a character-by-character basis to determine which of these characters to include in the CRC, when to check the CRC result, and when to drop the present byte sync and look for new byte synchronization.

Fig. 6 is a diagram of the COP state machine. Each time a new character is received by the front end, the state machine is entered through an interrupt service routine and a data code conversion takes place through the type table. This conversion determines what type the incoming character represents, like SYN, SOH, STX, or DLE. The type table need only be as large as the data code that contains the most characters, and the table's output, the type, is data code independent (see Fig. 7).

The type is then used as a displacement to index into the next state's jump table. This table is pointed to by the Next State variable and contains a direct pointer to the appropriate output routine. It is not until these routines are executed that any action based on the present character takes place. Parity check, CRC, and byte synchronization are controlled here. This includes a 2.6-second timeout, which is started when byte sync first occurs and is restarted anytime another sync pattern is received. If 2.6 seconds elapses before the next sync is received or another resync condition occurs, then the present byte sync is dropped to start the hunt for new synchronization.

Other tasks done by these output routines include updating of the Next State variable, and the control (on/off) of a hardware parity checking mechanism. In the event of a parity error, a separate interrupt service routine is entered to handle the received character. This routine may or may not enter the COP state machine, depending on the type of character received. After completion of the selected output routine, the front end returns to a quiescent state, waiting for the next character interrupt to occur. The only special characters placed in the capture buffer are parity error and block check error. No start block, end block, or abort block characters are used.

Data Capture Buffer

Information brought in from the line is formatted by the front end and then placed in the data capture buffer. The buffer can be accessed by entering the Examine Data menu from the top-level menu of the instrument. Display formats are offered that provide the information from the line in an organized manner. Once information is put into the buffer it is never physically modified.

The internal buffer format is the same as that used for the HP 4955A and the HP 4951A Protocol Analyzers. Fig. 8 shows the buffer data types. All of the information for displaying the data in each of the formats is contained in this buffer. Any tape containing buffer data or menus can be created on one unit and then transported and used on any other.

Character	Hex Value			Type Number
	ASCII 7(Odd)	EBCDIC	SBT	
SYN	16	32	3A	0
ENQ	85	2D	2D	2
DLE	10	10	1F	4
SOH	01	01	00	6
STX	02	02	0A	8
ITB	1F	1F	1D	10
ETB	97	26	0F	12
ETX	83	03	2E	14
All Others	--	--	--	16
NAK	15	3D	3D	18
EOT	04	37	1E	20
ACK0	D0	70	20	22
ACK1	31	61	23	22
WACK	38	6B	26	22
RVI	DC	7C	32	22
Framing or Parity Error				24
IDLE	7F(FF)	FF	3F	26

Second Control Character
" " "
" " "
" " "

Fig. 7. *Type table for the Bisync protocol.*

The Mixed DTE & DCE display format offers the transmit and receive data mixed together. A display format providing separate lines for the transmit and receive data is available by pressing the DTE over DCE softkey. If the user wants to see only one or the other, there are DTE only and DCE only formats also. In the Data & State display (Figs. 9 and 10), four control leads can be shown with the DTE and DCE data.

When one of the bit-oriented protocols such as SDLC, HDLC, or X.25 has been selected in the Setup menu, the information can be displayed in frame and/or packet formats. The Two Col Frame display format provides a breakdown of each frame into address, type of frame, N(S), N(R), poll/final bit, data, and FCS. The packet information, such as the Q and D bits, modulo, lcn, type of packet, Ps, Pr, M bit, and data, is supplied with the Two Col Packet display format. The Frame & Packet display shows the decoded frame and packet at one time with an extra display line for data.

A message decode format is provided for DDCMP protocol. This format displays the message type, byte count, Q and S bits, response, sequence, address, header CRC, data, and data CRC. Each message is provided with an extra display line so that as much data as possible can be shown. Applications provide additional display formats for protocols added to the HP 4953A.

When a page of information is displayed, the address for the first character is determined, and then a pointer is moved through the buffer to evaluate the words in the buffer. Only some of the information in the buffer is displayable.

The physical buffer area is broken into 1024-word blocks. The block number containing the first character on the display is shown in the lower right corner of the data display. Along with the transmit and receive data (DTE/DCE data), the buffer contains timing information, lead change information, and special condition characters for protocol specific information and errors on the line.

The DTE and DCE data words have bit 15 set to 1. Each DTE/DCE word represents a displayable data character and includes six bits of timing information.

A time mark (tickmark) is put into the buffer by the front end initially at run time and at intervals during a run. The tickmark contains the ten most-significant bits of a 16-bit timer in the front end. All other words in the buffer except for an absolute status word contain the lower six bits. Whenever the lower six bits overflow, a tickmark is put into the buffer. This information is baud rate dependent.

The timing is set to a frequency to guarantee a minimum of one-character resolution. Each tickmark is followed by an absolute status word which defines the current condition of each of the 15 leads. Cursor timing uses this time stamp and the time information included with the other types of words in the buffer to determine the time between any two characters on the display.

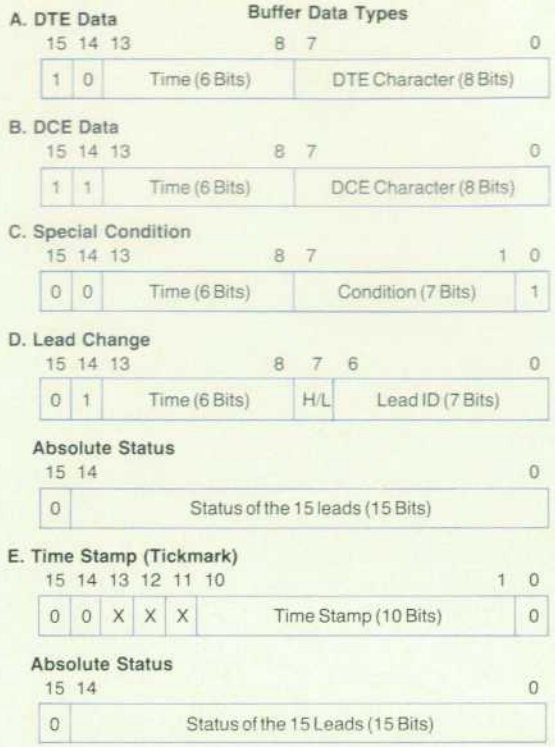


Fig. 8. *HP 4953A internal data buffer formats.*

A lead change word is always followed by an absolute status word. The lead change word supplies six bits of timing information, the address of the lead currently changing, and whether it has gone high or low.

Some of the special characters are displayable, some are not. The special characters representing the FCS and start/end flags for bit-oriented protocols are displayable characters representing bit sequences brought in from the line. Other special words in the buffer can represent parity or framing errors, in which case the previous data character is set to blink on the display, but the special condition itself is not displayed. The special words that represent the start of a message for the DDCMP protocol are simply information for the system to use, making it easy to display

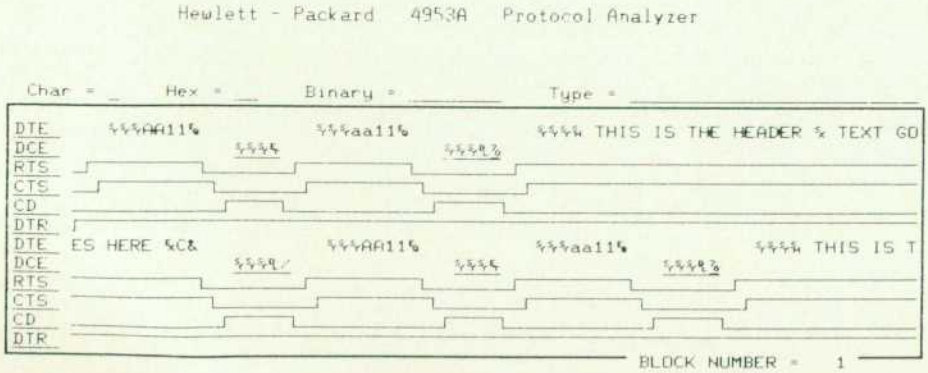


Fig. 9. *The* Data & State *display can show four control leads with the DTE and DCE data. Fig. 10 shows how the information for this display is stored in the data buffer.*

9EF1 ; DTE data character – 1

	Timing Information						EBCDIC DTE Data Character							
1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1

A22D ; DTE data character – Eq
000E ; tickmark
7F20 ; absolute status
6D80 ; lead change – RTS low

	Timing Information						H/L	Lead ID						
0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0

7F21 ; absolute status
6D81 ; lead change – CTS low
7F23 ; absolute status
0010 ; tickmark
7F23 ; absolute status
0012 ; tickmark
7F23 ; absolute status
0014 ; tickmark
7F23 ; absolute status
0016 ; tickmark
7F23 ; absolute status
6F05 ; lead change – CD high
7F03 ; absolute status
0018 ; tickmark
7F03 ; absolute status
001A ; tickmark
7F03 ; absolute status
001C ; tickmark
7F03 ; absolute status
001E ; tickmark
7F03 ; absolute status
0020 ; tickmark
7F03 ; absolute status
E432 ; DCE data character – Sy

Fig. 10. *Data buffer format for the* Data & State *display (Fig. 9).*

the messages, but neither affect other words in the buffer nor are displayable themselves.

The first character put into the buffer is always a tickmark, which is always followed by an absolute status word. This can be overlaid by new data after the buffer has been filled. When the HP 4953A has stopped executing (receiving data), the pointer to the next available location for data can be anywhere in the buffer, and the buffer may have filled any number of times, in which case the first valid tickmark is determined.

Acknowledgments

Mark Ennamorato and Sharlene Lin contributed to the pod designs that were originally done for the HP 4955A Protocol Analyzer. Toni Graham did the original COP state machine design, and Ed Moore and Tom Kowalczyk did the HP 4955A front-end hardware design, which is the basis for the HP 4953A front-end design. Peter Heitman was responsible for the original buffer format in the HP 4955A.

A Low-Cost, Portable Field Service Protocol Analyzer

It's menu and file compatible with HP's higher-performance analyzers and has many of the same capabilities.

by Vonn L. Black, Alan Delwiche, Chris L. Odell, and Stephen B. Tursich

DIGITAL DATA COMMUNICATIONS pervades the business and private sectors of our world so much that a truly portable protocol analyzer can be an invaluable tool in maintaining such equipment as computers, terminals, printers, and modems. The HP 4951A Protocol Analyzer (Fig. 1) is designed to meet the requirements for field service in a single portable test set. With the HP 4951A, a user can monitor data transmission, simulate network equipment, perform bit error rate tests, and remotely transfer data and programs. Customer support and field service personnel of service providers and computer or communications equipment manufacturers will find this compact analyzer useful for datacom network installation and maintenance. Communications centers will find its monitor and simulate capabilities helpful in diagnosing system operating problems.

The main contributions of the HP 4951A are high performance, ease of use, and low cost in a highly portable package weighing less than 6.5 kg. A menu-driven softkey user interface makes it easy to use. One softkey invokes an Autoconfigure routine that automatically determines a line's protocol, data code, speed, error checking, and other parameters. The HP 4951A is menu and data file compatible with the higher-performance HP 4953A (see article, page 4) and HP 4955A Protocol Analyzers, making possible remote transfers of data, programs, menus, timers, and counters between members of this analyzer family.

The built-in CRT display gives the user a choice of dis-

play formats. All data monitoring, triggering, and simulation can be performed at rates up to 19.2 kbps. When monitoring bit-oriented protocols, data can be captured at speeds up to 64 kbps. The bit error rate test simultaneously measures bit errors, block errors, errored seconds, and percent error-free seconds.

Architecture

Fig. 2 shows a simplified HP 4951A hardware block diagram.The NSC800 8-bit microprocessor is the heart of the system, which contains all the makings of a portable personal computer. The data link hardware gives the instrument its ability to analyze digital communications equipment. A full keyboard, a 5-inch CRT, and an integral DC100 tape drive are other hardware features. The HP 4951A has about the volume and weight of five liters of milk. This smallness, necessary for portability, was a driving factor in every step of the design. The low power consumption of the instrument, 13 watts, is commensurate with its size and contributes to its reliability.

The NSC800 is used because of its low-power CMOS characteristics, its enriched Z80 instruction set, and its flexible interrupt scheme. Byte-wide CMOS RAM provides the instrument with 64K bytes of memory, 32K of which is for the capture of data and timing information. This battery-backed RAM retains information for about six months (at room temperature) before the battery needs recharging. The system firmware resides in 104K bytes of internal ROM and provides the same menu-driven softkey user interface as the more expensive HP 4953A and HP 4955A Protocol Analyzers.

The specialized hardware for the data link control (DLC) centers around a serial communications controller IC. This part is programmed to support all standard bit rates up to 19.2 kbps, and both bit-oriented and character-oriented, synchronous or asynchronous protocols. Baud rate generation and NRZI protocol are also implemented by this chip. This dual-channel device allows the DLC to monitor both the DTE (data terminal equipment) channel and the DCE (data circuit-terminating equipment) channel simultaneously or to monitor one channel while simulating data on the other. Physical interfaces such as RS-232-C or RS-449 are supported by an interface module. MIL-STD 188-C is supported by the RS-232-C module; the user simply selects (by softkey) the data to be inverted.

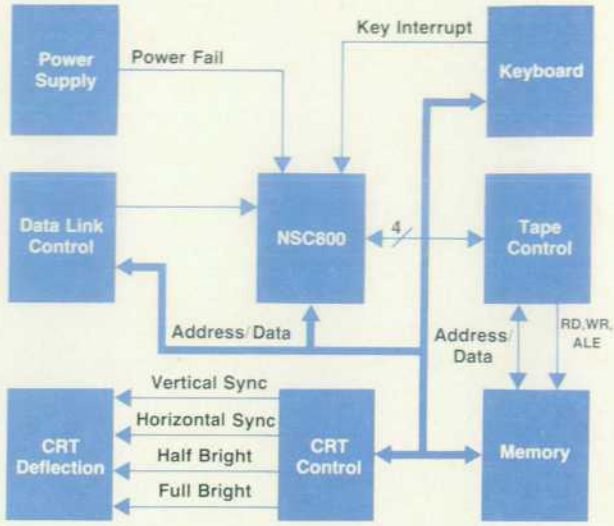


Fig. 2. *Simplified HP 4951A hardware block diagram.*

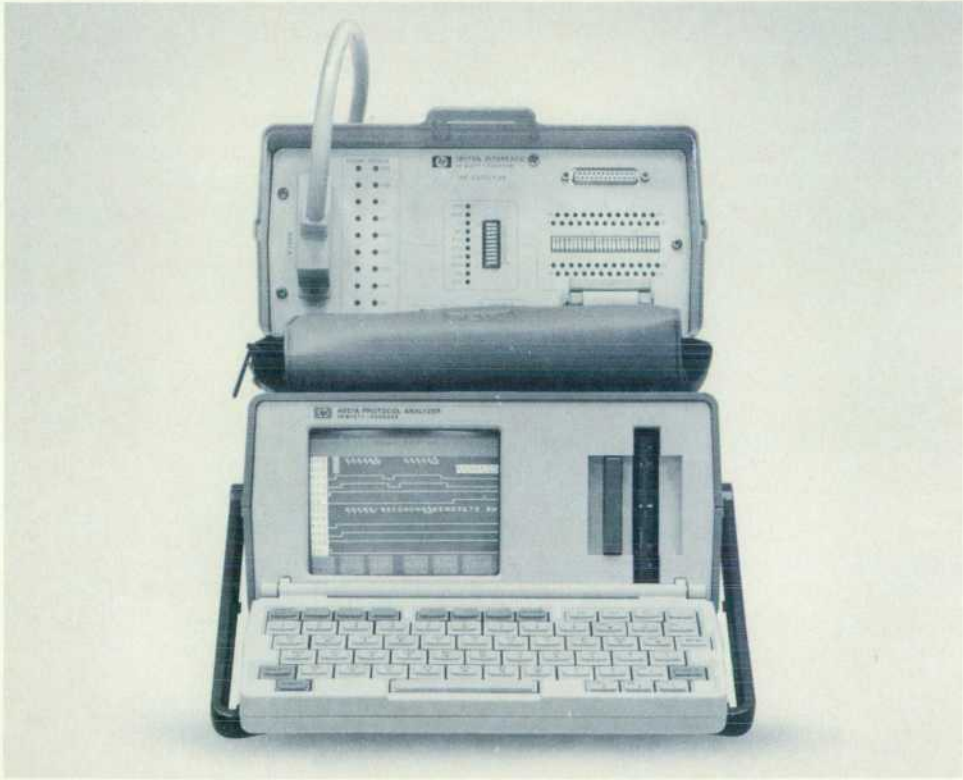


Fig. 1. *The HP 4951A Protocol Analyzer is designed for field service. It has a menu-driven softkey user interface, and one softkey invokes an Autoconfigure routine that automatically determines a line's protocol, data code, speed, error checking, and other parameters.*

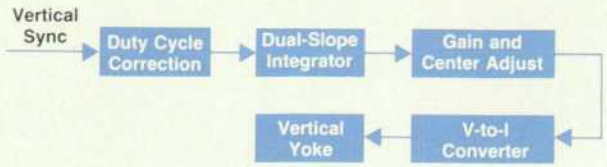


Fig. 3. *HP 4951A vertical deflection amplifier block diagram.*

The DLC also provides a high-resolution clock to time stamp data or control lead activity for use in timing measurements. Bit error rate tests (BERT) are also performed via the DLC. The HP 4951A is the only member of the HP protocol analyzer family that has this feature, which provides a means of digitally testing the integrity of the physical data connection.

CRT Display

The green-phosphor CRT displays 16 lines of 32 characters each using an 8×14-dot character cell and a 7×9-dot character font. The extra large cell provides underline, overbar, and underbar attributes to all fonts. Additional attributes implemented in the deflection circuitry include half bright, inverse video, blinking, and flashing video.

The vertical deflection amplifier block diagram is shown in Fig. 3. Vertical sync is provided by the CRT controller and is synchronized with the horizontal and video timing. After duty cycle adjustment, the vertical sync signal drives a dual-slope integrator to create a voltage ramp (sawtooth). The gain and centering stage affects the magnitude and position of the vertical deflection of the CRT beam. The voltage ramp is converted to a current ramp in the voltage-to-current converter stage, which drives the vertical deflection yoke. For small angles of deflection, the amount of deflection is proportional to the current in the yoke. Therefore, a linear current ramp is used to create the desired vertical beam deflection. The amplifier does not correct for the nonlinearity in the tangential beam velocity which results from maintaining a constant angular velocity through a varying radius. For the CRT used the resulting distortion is less than 10% and is deemed acceptable, allowing the design of the vertical amplifier to be simplified to save space, cost, and power.

The video amplifier (see block diagram, Fig. 4) is designed for low-power operation. The amplifier shifts the voltage level of the dot information provided by the CRT controller to a level suitable for driving the cathode of the CRT. The amplifier generates levels corresponding to full bright, half bright, and blank states. The drive circuits use Schottky diode clamps and speed-up capacitors in the transistor base circuits for fast switching. Low-power operation is achieved by providing a relatively high-impedance dc path to the cathode, but a much lower ac impedance for the level drivers; this makes it possible to drive the cathode capacitance with the speed necessary for crisp transitions in intensity (full bright, half bright, or off).

While most of the horizontal amplifier circuitry is conventional, the drive circuit has been greatly simplified. The generally practiced method of driving the flyback transformer with a bipolar switch was abandoned in favor of a power MOSFET scheme. The FET eliminates the temperature dependent base storage time of bipolar devices and the transformer base drive circuitry required to turn off the saturated transistor quickly. The major benefit, however, is the elimination of phase control circuitry by driving the horizontal amplifier open-loop, directly from the horizontal sync signal generated by the CRT controller, while holding the overscan to 5% of the total sweep.

The CRT display consumes only seven watts and requires less than 40 square inches of printed circuit board area.

Mass Storage

The battery-backed RAM is complemented by the additional data and menu storage capabilities of the DC100 tape drive, which uses the delta-distance recording scheme and HP's standard Logical Interchange Format. Low power, small size, low cost, and performance were the overriding factors in the design of the tape controller.

Power is conserved by pulse width modulation of the voltage to the motor and by switching off the lamps when the motor is idle. Active power consumption of the drive is less than 10 watts during fast forward operation and only 6 watts during read/write. Standby power is about 100 mW.

The tape electronics occupies a printed circuit board area of less than 30 square inches and provides double the throughput of the HP 4955A's tape drive. Read/write speed is 40 ips at 1600 flux reversals per inch. An 8048 microcontroller handles data formatting and control of direct memory access to the system memory area. This translates into less than 3% overhead on the main processor during the most demanding run-time data transfers. Each tape cartridge provides about 16 times the capacity of the 32K-byte data capture buffer.

The tape controller was originally designed for the HP 4951A and has been incorporated in the higher-performance HP 4953A as well.

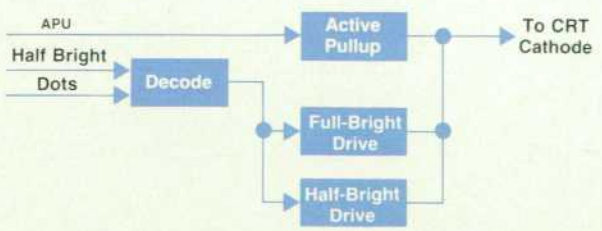


Fig. 4. *HP 4951A video amplifier block diagram.*

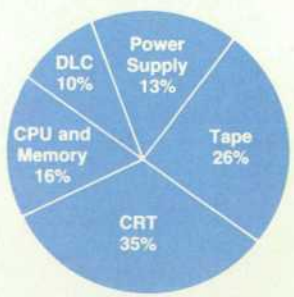


Fig. 5. *HP 4951A printed circuit board space allocation.*

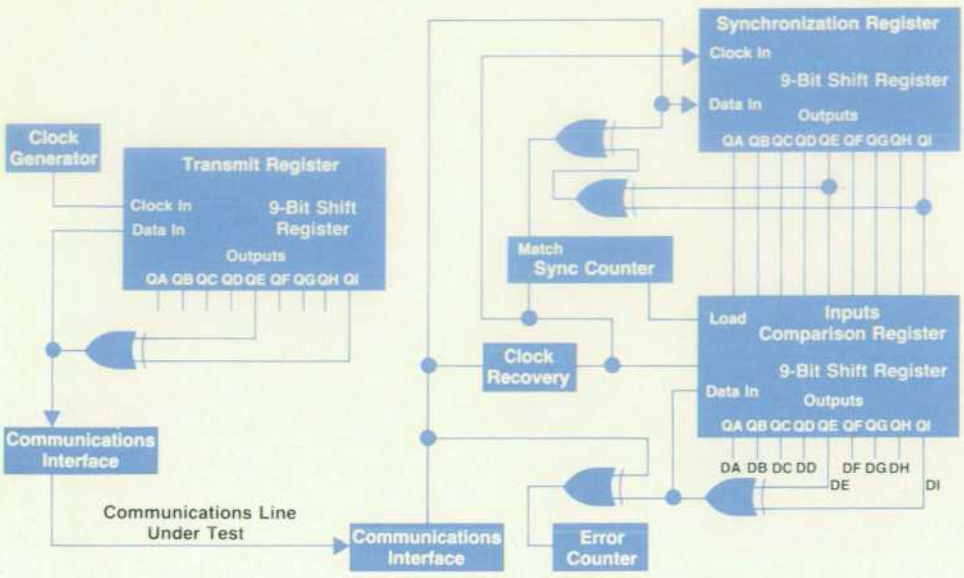


Fig. 6. *Block diagram of a hardware implementation of the bit error rate test (BERT).*

Power Supply

A custom switching power supply was designed to provide the necessary efficiency in a small area. A commercially available supply was not practical at the time because of the irregularity of the load, most of which is on the +12V supply for the CRT and the tape motor. The HP 4951A supply switches at 100 kHz and provides up to 35 watts of power (while the tape motor is accelerating) with less than 10 mV of ripple on the 12V supply. Even with a dc-to-dc efficiency of approximately 85%, it was critical that all the electronics consume as little power as possible. The CRT and the tape electronics are prime power conservers in this instrument.

System Firmware

As can be seen in Fig. 5, only about 10% of the available space in the HP 4951A is dedicated to special hardware for protocol analysis. The instrument is tailored to data communications by the 104K bytes of resident firmware (software stored in ROM). The HP 4951A not only provides the user with a friendly, menu-driven interface similar to that of the HP 4955A, but also provides special measurement features and capabilities. These include BERT, Autoconfigure, and many applications packages.

The user interface provides 15 menus from which approximately 50 parameters may be selected, in addition to the Monitor and Simulate menus and the applications. Each applications program may contain hundreds of lines that cause data to be sent, tested, timed, counted, and so on. Five timers and five counters are accessible through these programs, just as they are in the larger HP 4953A and HP 4955A. Program flow can be based on the occurrence or lack of any combination of up to 63 trigger events or interface conditions. The BERT and Autoconfigure menus of the HP 4951A are not provided in the other two analyzers.

Bit Error Rate Test

One of the functions of the HP 4951A that didn't fit in the limited amount of hardware space is bit error rate testing. Instead, this function is implemented in software, using the DLC hardware. Performance requirements prevented a simple implementation of the hardware algorithm, necessitating a new approach to the bit error rate test.

Bit error rate testing involves sending a predefined bit stream over a data communications channel and analyzing the data received at the other end of the channel to determine the error rate. Fig. 6 shows a block diagram of the main elements in a hardware implementation.

The transmitter is a shift register with certain bits fed back to create a pseudorandom bit sequence which repeats every 511 bits. The receiver contains two shift registers with the same bits fed back. A sync register does not have feedback but is used to compare each bit received with the bit expected, based on the previous nine bits. The sync counter counts the number of consecutive matches between expected and received bits. Synchronization is achieved when nine consecutive received bits match the expected values. The comparison register is then loaded with the current contents of the sync register. These contents are the same as those of the transmit register for a given point in the bit stream and cause the comparison register to output the same bit pattern as the transmitter. This becomes the reference for comparison against the incoming data. If the data received differs from the output of the comparison register, an error is counted.

The straightforward way to implement such a tester in software is to simulate each of the three registers, operating on one bit at a time. At the top speed of the HP 4951A (19,200 bps), each bit must be analyzed in just 52 microseconds. The C code shown in Fig. 7 for this implementation of the BERT will not run fast enough on a 4-MHz NSC800. In fact, a handcompiled version of this algorithm is much

```
/*  Transmit shift register: next_bit = the next bit to transmit, */
next_bit = ((current_transmit_state & 0×100) >> 8) ^ ((current_transmit_state & 0×10) >> 4);
current_transmit_state = (current_transmit_state << 1) | next_bit;

/*  Receiver synchronization shift register: when 16 valid bits have
    been received, we are synchronized; when 16 invalid bits have
    been received, we are out of sync.  */
next_sync_bit = ((current_sync_state & 0×100) >> 8) ^ ((current_sync_state & 0×10) >>4);
if (next_sync_bit ! = received_bit)
{
  sync_count = 0;
  if (++out_sync_count > 16)
  {
    out_sync_count = 16;
    out_of_sync = 1;
  }
}
else
{
  out_sync_count = 0;
  if (++sync_count > 16)
  {
    sync_count = 16;
    out_of_sync = 0;
    current_receive_state = current_sync_state;
  }
}
current_sync_state = (current_sync_state << 1) | received_bit;

/*  Receiver error checking shift register; received_bit = the bit
    just received from the data line, next_receive_bit = expected value. */
if (out_of_sync = = 0)
{
  next_receive_bit = ((current_receive_state & 0×100) >> 8) ^ ((current_receive_state & 0×10) >> 8);
  current_receive_state = (current_receive_state << 1) | next_receive_bit;
  if (next_receive_bit ! = received_bit)
  {
    ++error_count;
  }
}
```

Fig. 7. *C code for a software implementation of the BERT hardware shown in Fig. 6. This approach is too slow for the HP 4951A.*

too slow (about 100 μs per bit), so a new approach was taken for the HP 4951A.

The hardware implementation uses serial operations, while a software approach has the advantage of operating on eight bits at a time. Let's look at what happens to the contents of the shift register over the course of eight consecutive shifts. We see in Fig. 8 that the new value generated at each state is shifted into bit QA, and the rest of the register is shifted right one bit. Thus, after eight shifts, bits QA through QH contain the next eight values that would have been generated. The last line of the table shows the formulas for each bit. Examining these results we find that we can generate the final state by doing an eight-bit-wide exclusive-OR of bits QB through QH with bits QG, QH, QI, QA, QB, QC, QD, and QE, and an exclusive-OR of three bits of the result with bits QC, QD, and QE, and rotating the result to the correct position.

A section of Z80 code to implement this algorithm is shown in Fig. 9. This code executes in 23.5 μs on a 4-MHz processor, and the code to implement all three registers takes less than 100 μs for eight bits. This is roughly equivalent to the time for the bit-serial simulation and represents an eightfold improvement in throughput.

Such careful attention to optimizing algorithms is the key to the performance of the HP 4951A and also shows up in the data capture and analysis software.

Autoconfigure

What do you do when you want to monitor a data line, but you don't know anything about the data? Is the protocol SDLC? Is the data synchronous? Is the data code ASCII? When the user presses the Auto Configure softkey on the top-level HP 4951A menu, an algorithm called Autoconfigure uses the data it receives on the line to determine—often statistically—the protocol and other parameters. These parameters are displayed in a setup menu, and the HP 4951A automatically goes into the Run mode and begins monitoring. The main advantage of Autoconfigure is that it allows even novice users to monitor a data line.

How does Autoconfigure decide what's on the line? It looks for the simplest parameters first and then uses this information to look for the more complex parameters. First, it finds the baud rate by measuring the time between transitions on the data line. This bit time is measured over a large number of bits and the lowest bit time is used. Using this method, Autoconfigure can measure baud rates up to 19.2 kilobits per second.

After the baud rate is found, the next step is to determine whether the data is synchronous or not. A transmit clock, supplied by the DCE or the DTE, is required to transfer synchronous data. If there is a clock, a timer is used to determine if a clock pulse is virtually equivalent to a bit time, as previously found. If so, Autoconfigure looks for synchronous protocols, such as SDLC or Bisync. Otherwise,

Shifts	Shift Register Bit Position								
	QA	QB	QC	QD	QE	QF	QG	QH	QI
0	a	b	c	d	e	f	g	h	i
1	e+i	a	b	c	d	e	f	g	h
2	d+h	e+i	a	b	c	d	e	f	g
3	c+g	d+h	e+i	a	b	c	d	e	f
4	b+f	c+g	d+h	e+i	a	b	c	d	e
5	a+e	b+f	c+g	d+h	e+i	a	b	c	d
6	d+e+i	a+e	b+f	c+g	d+h	e+i	a	b	c
7	c+d+h	d+e+i	a+e	b+f	c+g	d+h	e+i	a	b
8	b+c+g	c+d+h	d+e+i	a+e	b+f	c+g	d+h	e+i	a

Fig. 8. *BERT shift register contents during eight consecutive shifts. The HP 4951A BERT algorithm generates the final state by performing two exclusive-OR operations and a rotation.*

it looks for asynchronous protocols, such as character async or SDLC-NRZI.

Autoconfigure tries one synchronous protocol at a time. If it finds the flags or sync characters it is looking for, the protocol is considered found and Autoconfigure begins looking for the data code and other parameters. It looks for synchronous protocols in the following order:

1. Bit-oriented protocols or BOPs (e.g., SDLC). Autoconfigure looks for 7E flags and a good CRC-CCITT error check.
2. Bisync. Autoconfigure looks for standard sync characters (e.g., 32H 32H).
3. Other character-oriented protocols. Autoconfigure looks for nonstandard sync characters, (e.g., Transcode's 3AH 3AH with parity).
4. IPARS. Autoconfigure looks for the 3FH 3EH pattern.

When Autoconfigure looks for asynchronous protocols, it tries SDLC-NRZI first. It searches for 7EH flags and a good CRC-CCITT error check. If these are not found, Autoconfigure tests for asynchronous data by first trying to frame on 5-bit characters. If too many framing errors occur, the next frame size is tried until an acceptable frame size is found or until the maximum frame size has been tried.

The data code is determined in different ways. For bit-oriented protocols, ASCII is chosen if the most-significant bits of the characters are zeros; otherwise, EBCDIC is chosen. For Bisync and other character-oriented protocols, the data code is often determined by the corresponding sync characters (e.g., EBCDIC uses 32H 32H). For asynchronous protocols, the frame size is instrumental in determining the data code (e.g., Baudot is the data code chosen for 5-bit frames).

Autoconfigure users should be aware of certain features that may affect them. First, Autoconfigure always selects SDLC for bit-oriented protocols. Monitoring will always be correct except in some cases of X.25 or HDLC with extended address and extended control. Also, whenever Autoconfigure sees a character-oriented protocol, including Bisync, it displays the parameters on a Char Async/Sync setup menu. Other parameters Autoconfigure may find, if applicable, include a transparent text character, the bit sense, and the error-detecting code.

Acknowledgments

One of the most important ingredients in the HP 4951A design recipe is expressed well by Thomas Edison's statement, "I never did a day's work in my life—it was all fun." The entire project team enjoyed working together and demonstrated it by open and frequent communication between team members. Colin Appleyard headed the project team and was mainly responsible for the definition of the instrument. Will Taylor and Eck Zimmerman did the mechanical design. Tom Wisdom wrote the Autoconfigure and remote code. We also recognize the many people who supported the project through its development outside the R&D laboratory and those responsible for assembling the HP 4951A on the production line.


```
Starting State: Register H = a b c d e f g h,
                Register L = i 0 0 0 0 0 0 0.

        ADD     HL,HL   ;  H = b c d e f g h i, CARRY = a.
                        ;  L = 0 0 0 0 0 0 0 0.
        LD      B,H     ;  B = b c d e f g h i,
        RL      L       ;  L = 0 0 0 0 0 0 0 a, CARRY = 0.
        RRC     L       ;  L = a 0 0 0 0 0 0 0, CARRY = a.
        RL      H       ;  H = c d e f g h i a, CARRY = b.
        LD      A,H     ;  A = c d e f g h i a.
        RL      H       ;  H = d e f g h i a b, CARRY = c.
        RL      H       ;  H = e f g h i a b c, CARRY = d.
        RL      H       ;  H = f g h i a b c d, CARRY = e.
        RL      H       ;  H = g h i a b c d e, CARRY = f.
        AND     A,0E0H  ;  A = c d e 0 0 0 0 0.
        XOR     A,H     ;  A = c+g d+h e+i a b c d e.
        XOR     A,B     ;  A = b+c+g c+d+h d+e+i e+a f+b g+c h+d i+e.
        LD      H,A     ;  H and L = new state.
```

Fig. 9. *Z80 code to implement the HP 4951A BERT algorithm.*

Remote Monitoring and Control of Semiconductor Processing

This addition to HP's Semiconductor Productivity Network acts as a host computer to IC processing equipment, providing remote control and data gathering for fabrication personnel.

by Wesley H. Higaki

THE INCREASING FABRICATION COMPLEXITY, shrinking device geometries, and larger chip sizes of VLSI circuits are requiring more and more automation to provide acceptable yields and adequate control of process parameters. Automated process monitoring and control can provide exact duplication of processing steps, accurate measurements of the results, and precise adjustments to improve the process.

Automation is common to some degree in modern semiconductor processing and measurement equipment. Microprocessors and minicomputers control the process steps in most wafer processing systems (diffusion furnaces, plasma etchers, ion implanters, etc.) by using closed-loop feedback with stored "recipes" or process programs and loop coefficients. This reduces mechanical complexity (fewer switches, relays, etc.) and minimizes the need for human intervention during the process sequence. If a suitable interface to an external computer system can be provided, such process equipment can be controlled and monitored remotely.

Defining, creating, and storing recipes and then applying them to computer control allows greater repeatability of a process step. Like a computer program, process programs yield greater control over the quality of the output once the programs have been qualified. A computer-controlled piece of equipment fed the same program over and over will yield almost exactly the same results each time; with manual control, the results can be unpredictable. Computer control also allows for easy monitoring and recording of the process. Since closed-loop feedback control relies on monitoring capabilities, storing and analyzing the monitored data becomes the next logical step.

Equipment controllers are designed to monitor and control a process step. They have limited data storage and computing capabilities. They also have a very narrow view of the overall integrated circuit fabrication process. Individually, each piece of process equipment can optimize and control the process step for which it is responsible, but cannot correlate the data it collects with the data other pieces of equipment have collected. To do this, a host computer system is required.

PC-10: Equipment Monitoring and Supervision

PC-10 is a recent addition to HP's Semiconductor Productivity Network (SPN, see Fig. 1). This product (Fig. 2) acts as a host computer system for processing equipment. PC-10 provides the capability to monitor and control semiconductor processing equipment remotely. Its greatest asset is that it provides the capability to store and download recipes to the appropriate system at the appropriate time. This feature alone increases equipment versatility and can reduce processing errors dramatically. From the PC-10 system, a user can request equipment status, store measurement data, remotely control equipment, store and restore calibration data, and store and restore recipes. Equipment-generated alarm conditions are recorded and reported.

PC-10 acts as the central node in the equipment network to gather all of the data from the equipment and feed it

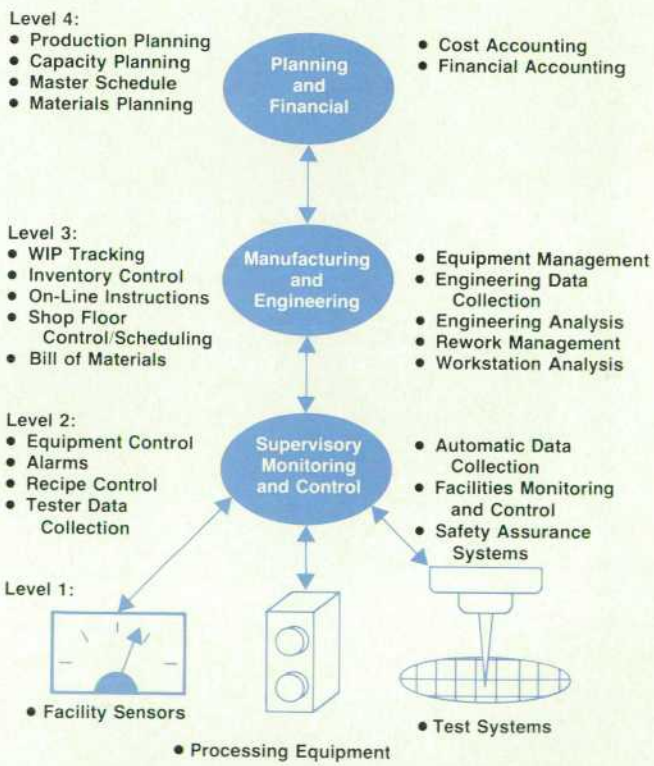


Fig. 1. *HP's Semiconductor Productivity Network is a collection of hardware and software products designed to aid microelectronic product fabrication personnel in controlling, monitoring, improving, and managing many of the levels of the complex manufacturing process.*

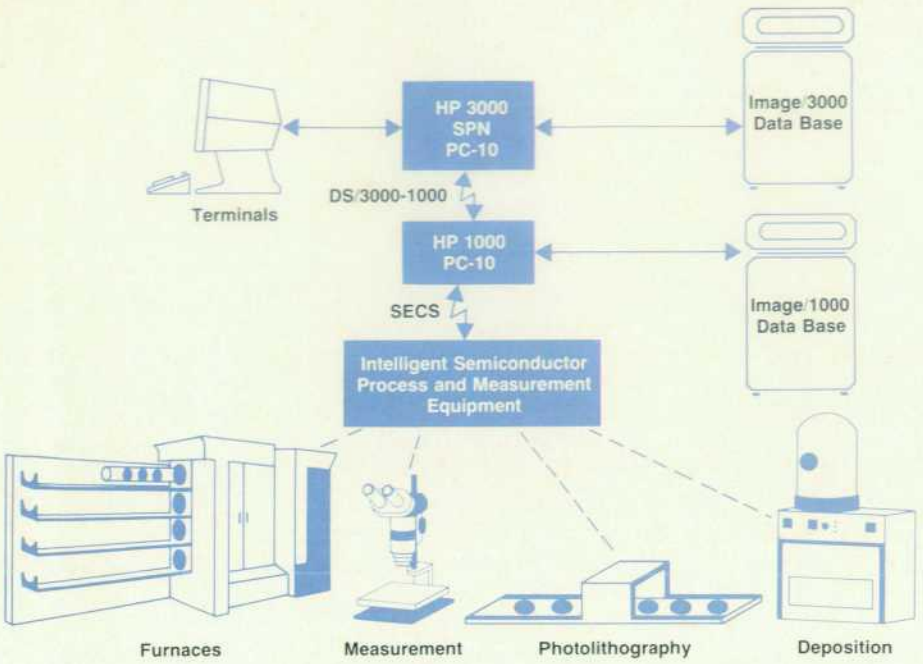


Fig. 2. *PC-10, an SPN product addressing levels 1 and 2 in Fig. 1 and coupled to IC-10 in level 3, allows the user to access fabrication equipment for checking its status, giving it a new recipe, monitoring process values, and controlling it without having to enter the fab area. Connections to the equipment are accomplished through the use of special message translators interfacing with SECS ports.*

into the SPN system. Process engineers can monitor their processes much more precisely by having the equipment report events to PC-10 for recording in material and equipment history logs. Once the data has been gathered, engineers can use other SPN tools to correlate the data from one step in the process with data from other (previous) steps. This gives the process engineer overall visibility of the process that could not be obtained easily without a host computer system.

The ability to store and download processing recipes ensures not only that the specified operation is performed, but that any engineering changes to these recipes are applied uniformly to any affected operation. Since PC-10 is the central node in the network in charge of maintaining recipes, engineers are assured that the latest version of the recipe is downloaded to the equipment.

Equipment monitoring, through PC-10, gives engineers a window into the fab area from their offices (Fig. 3). They

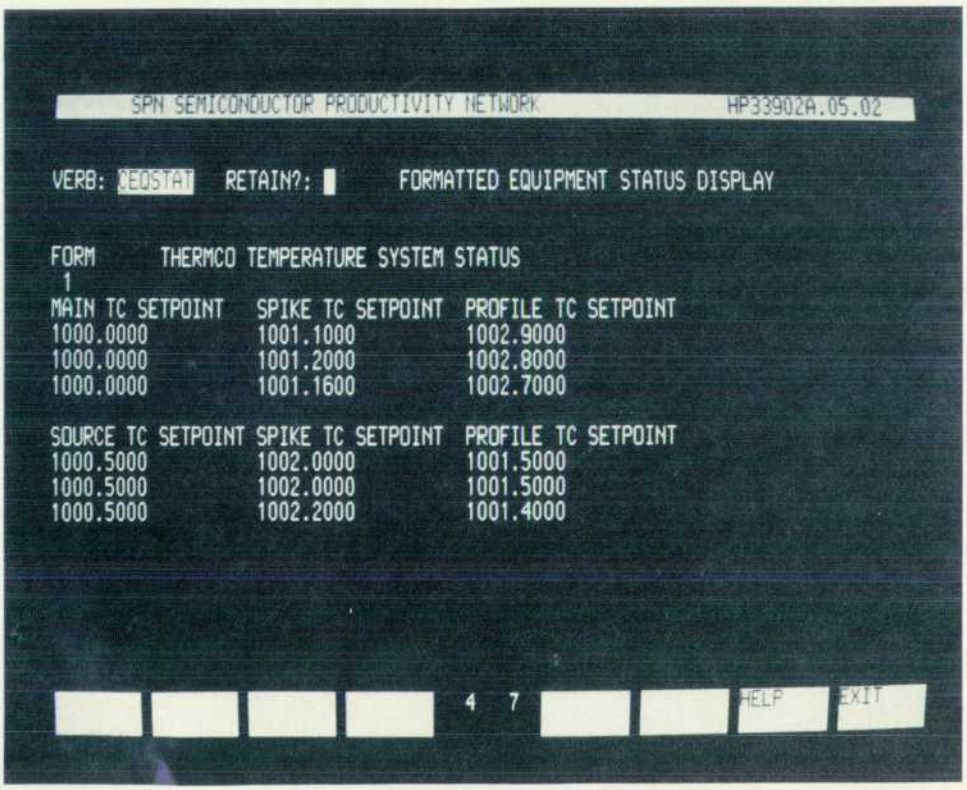


Fig. 3. *Typical SPN display screen using PC-10 to access the status of a diffusion furnace.*

can monitor the progress of their processes and their equipment at remote SPN terminals. Through PC-10 they can also review all of the history information that has been collected from the equipment.

Data Communications

Data communications is a major part of PC-10 in two areas. PC-10 uses DS/3000-1000 to link the SPN HP 3000 Computer System with HP 1000 Computers, which act as real-time network nodes. PC-10 also relies heavily on the SECS communications standard (see box on page 33) to link the HP 1000 Computers to the process equipment.

The DS/3000-1000 link allows PC-10 to use the real-time capabilities of an HP 1000 to interface processing equipment. User transactions initiated on the HP 3000 through the SPN transaction handler are transmitted down the DS line to the HP 1000 for processing. This link is useful for functions such as automatically downloading a recipe when a tracked material lot arrives at different pieces of equipment.

PC-10 has applications software on top of the process-to-process communications capabilities in DS/3000-1000. These applications have the intelligence to recover from any soft DS failures transparently to all of the users on the system. PC-10 also has the capability to link multiple HP 1000 Computers to a single HP 3000 central node.

Communication with the process equipment is the key to monitoring and supervision. To communicate with each of the different types of process equipment, PC-10 relies heavily on the use of the equipment manufacturing industry's communications standard. The implementation of this standard by process equipment manufacturers and PC-10 has made connecting the equipment to a remote monitoring and control system a much simpler task.

History of PC-10

PC-10 has essentially two roots: the tightly integrated SPN family of software packages and the process control system (PCS) developed and used by HP Laboratories.[1] PC-10 is designed to integrate with the rest of the SPN products such as IC-10, EN-10, and EA-10.

IC-10 is the major tracking product of SPN. IC-10 is based on the HP 3000 Computer using Image data bases, one for parameter data and one for production data. Its data bases define the location-level tracking of material (wafers) through the process. A location is a general area in the fabrication area such as diffusion or photomasking. Since PC-10 addresses individual pieces of equipment and not general locations, operation-level tracking has been developed to subdivide locations into discrete equipment operations. EN-10 is used to collect measurement data manually for analysis by EA-10.

Between IC-10 and operation-level tracking, process engineers can completely define the flow of material through the fabrication area. When material is tracked through this process to a particular piece of equipment, which is linked via SECS to PC-10, the proper machine instructions can be loaded and the process can be monitored from any SPN terminal. Measurement data collected by PC-10 from the equipment is stored in the SPN production data base for analysis by EA-10.

PCS was developed by HP Laboratories' Integrated Circuit Processing Lab (ICPL) to track material through operations with the capability of transferring the machine instructions to the processing equipment, specifically a Thermco diffusion furnace with minicomputer controllers. This system was based on an HP 1000 using an internally developed data base and communicating with the furnaces using a forerunner of SECS. Many HP IC facilities now use some version of PCS to operate a portion of their fabrication areas. The problems with PCS are that it does not provide the complete solution SPN does, it is unsupportable as an external product, and it was designed for communicating with only the Thermco furnaces.

To be successful in the marketplace, PC-10 had to address two major issues: integrate well with SPN and interface with many different kinds of semiconductor processing equipment.

Interfacing Equipment

To automate a complete IC fabrication area, PC-10 must be able to communicate with many different types of equipment built by many different manufacturers. Our goal has been to develop interfaces easily and quickly, which means

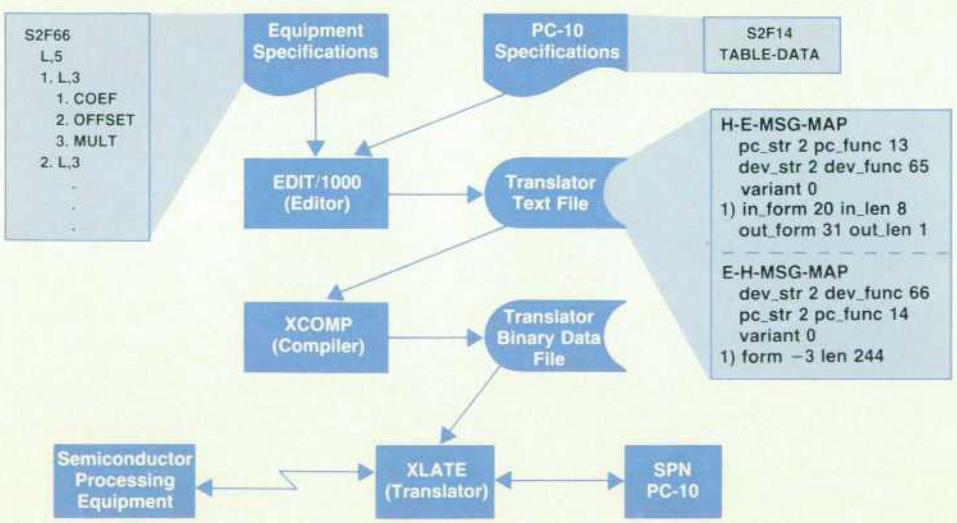


Fig. 4. *Translator flow diagram. New equipment is added easily to PC-10 by simply developing the appropriate host-to-equipment and equipment-to-host tables for the associated SECS II data streams. Examples are shown for stream 2 function 14 and stream 2 function 66 messages (i.e., S2F14 and S2F66).*

avoiding writing custom software to fit all of the idiosyncrasies of individual processing systems. Instead, PC-10 handles IC process equipment by separating it into general classes. SECS II (see box below) is a mandatory prerequisite of the equipment before an interface to PC-10 can be developed.

HP's approach to interfacing is to survey a representative number of processing systems within a class to develop a generic model. A class is a group of equipment systems that operate similarly and perform the same general functions so that the communications requirements look the same to PC-10. The assumption is that each piece of equipment in an equipment class supports a subset of the SECS II data streams and functions that PC-10 supports for that class. We also assume that the order of the messages, which is not defined by SECS II, is generally the same for all equipment in that class. To date we have encountered batch, metrology, serial, and material handling equipment. We expect that this list will grow as we begin to explore the use of PC-10 for remotely controlling and monitoring other types of equipment.

Batch processing systems, such as diffusion furnaces, process wafers in large quantities (batches). The primary characteristic is that once a batch has started processing, no more wafers can be added until the process sequence has run to completion. PC-10 will download only one recipe to the batch station when the batch is tracked in and no other batches will be allowed in until the first batch is done.

Metrology systems are classified separately because they provide certain measurement data to PC-10. PC-10 supports SECS II stream 6 messages, which handle the transfer of measurement data to the host system for this class of equipment. Examples are line width, film thickness, and defect measuring devices. Wafers passing through these stations are not processed, but are merely measured to determine the effectiveness or accuracy of previous process steps.

Serial processing handles wafers one at a time. Wafers from one lot may be entered into the equipment for processing before the preceding lot has been completed. Photolithography wafer track systems are a prime example of serial equipment. PC-10, to ensure that the proper recipe is executed for each lot, must check to see if the recipe already executing is the proper recipe for the next lot. If not, it must download the new correct recipe at the proper time for beginning processing of the new lot.

Material handling requires an entirely different set of messages, since material handling systems are responsible only for transporting the wafers from one station to another. PC-10 instructs the material handling system to take a lot or a group of lots to a particular piece of equipment. Examples of material handling systems include robots or tracks used to move the wafers through the fabrication area.

The challenge we face when we address a new class of equipment to develop our models is to perform an adequate survey of such equipment on the market and develop an accurate, yet general model of how these pieces of equipment operate. This requires that we understand how the equipment works, what it is used for, and how PC-10 should interact with it. In the past we have reviewed communications specifications from the vendors and received training on how to operate the equipment.

Once our models have been developed, it becomes a matter of evaluating equipment to see if they fit the model. If a piece of equipment fits the model, then we must resolve any differences in the equipment's SECS message format and content. PC-10 does this at the lowest level of its software—the message translator.

Translators

The translators convert SECS messages as interpreted by the equipment into PC-10 interpretable messages. This includes translating binary values into ASCII representations and vice versa, resolving data item length differences, masking off unneeded data items, and formatting data for displays or data base records.

The translators are table-driven. Thus, to interface a new piece of equipment, all that must be done is to generate a new table and pass it to the translator program (Fig. 4). No new code is written. This technique greatly reduces the time needed to develop a new interface compared to having to custom-code the interface.

The price we pay for this short development time is an interface that may not take advantage of all of the features

SECS

The semiconductor process equipment manufacturers have identified the need for their equipment to communicate with a larger host computer system and developed the Semiconductor Equipment and Materials Institute (SEMI) Equipment Communications Standard (SECS). SECS defines parts of all seven ISO open system interconnect (OSI) communications layers. SECS I incorporates the use of RS-232-C cabling and pin definitions and a relatively simple line protocol. SECS II defines messages to request and send status information, transfer recipe data, report alarm conditions, send remote equipment control commands, and handle material transfer.

SECS I uses a simple ENQ-ACK handshake across an RS-232-C line with checksums at the end of each message. SECS I also defines time-out intervals between handshake responses, individual message characters, and message responses. Message headers are defined in SECS I to include equipment identifiers, message identifiers, message block numbers, and other system information.

SECS II defines message types, format, content, and directions. SECS streams are groups of messages assigned to a general set of equipment functionality. Within each stream, the individual messages are assigned function numbers. For example, SECS stream 1 function 5 (abbreviated S1F5) is a formatted equipment status request, and stream 1 function 6 is the reply with the status information. Similarly, stream 7 function 5 is used to request the transfer of a process recipe and stream 7 function 6 is used to transfer the recipe. SECS II also defines whether a reply is required or not, the message content and format (including data item definition headers), and whether a message may be used from equipment-to-host and/or host-to-equipment.

A major limitation of the SECS standard is that it defines messages and their content only; it does not define how the messages are used together to perform a function. Equipment manufacturers are left to decide what messages to use to perform functions that were performed manually before. This, of course, makes it difficult to develop translators for external systems to communicate with such equipment.

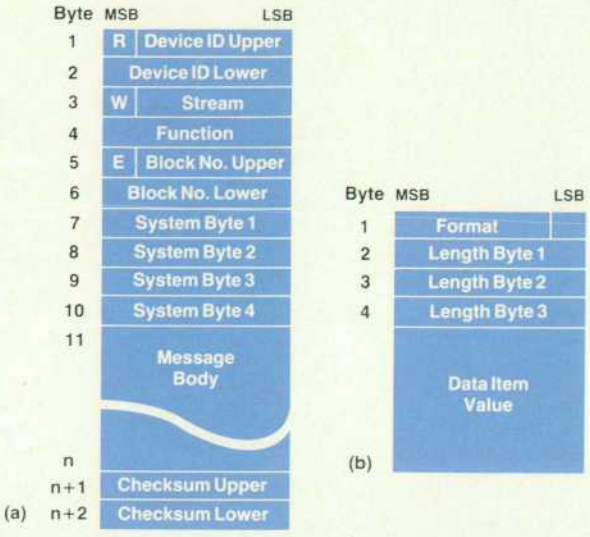


Fig. 5. *SECS II formats for (a) messages and (b) data items.*

a piece of equipment has to offer or may make operating the equipment more awkward than if custom software were written. However, with only the translators changing for each different piece of equipment, the other software layers remain unchanged and so do not have to be updated whenever new equipment is added to a fabrication area.

The first step in building a translator is to review the equipment vendor's SECS implementation specifications to see if the general interpretations match those of PC-10; that is, there is a message comparable to that of PC-10's for every critical message. Critical message sets differ from equipment to equipment. A measurement data report is critical to film thickness measuring devices. Recipe transfer messages are critical for furnaces, because that capability is what makes the interface useful.

SECS II defines a 10-byte header and message bodies (see Fig. 5). Within the 10-byte header are the device ID (bytes 1 and 2), message type (stream) and function (bytes 3 and 4), message block number (bytes 5 and 6), and system bytes (bytes 7 through 10). The translator is used to resolve the differences in the interpretation and usage of stream and function numbers between PC-10 and the equipment.

Within the SECS II message body, there can be several data items. Each item is preceded by a format header that describes the item. This header is a 2-to-4-byte item descriptor. The first byte contains the data format (i.e., ASCII or binary) and a 2-bit length byte count. The length byte count indicates the number (1 to 3) of following bytes that contain the length of the data item (Fig. 5b). The translator resolves the implementation and the equipment interpretation.

The translator resolves differences in the PC-10 and equipment vendor's interpretations of SECS II streams, functions, and data item formats by using a translator table for host-to-equipment messages and another table for equipment-to-host messages (Fig. 4). Each table maps the appropriate equipment message stream and function to an equivalent PC-10 stream and function. This mapping is not necessarily one-to-one.

A variant table is included to provide the capability for a one-to-many mapping of messages. The first item in the message is used to determine which stream, function, and message body format are used. The variant table entries contain data (either ASCII or binary) to be compared with the message data. If a match is made, then the mapping entry with the corresponding variant number is used to format the translated messages.

Each table also maps, item by item, the data format and length to be sent. Each line in the host-to-equipment map defines the format and length of the individual data items from PC-10 to the translator and the format and length to be used when the message is sent to the equipment. The equipment-to-host map contains the description of the data items as they will be passed to PC-10.

The table entries have the ability to pass the data through the translator without reformatting. This is a feature PC-10 uses to handle calibration and recipe data, since PC-10 will only store and deliver recipe data as the equipment provides and expects it.

The table entries can also mask off data items as necessary. This is useful when PC-10 provides data items that the equipment does not use or expect in its messages.

Reference

1. C.R. Clare, "A Process Control Network," *Hewlett-Packard Journal*, Vol. 32, no. 6, June 1981.

Authors

July 1985

4 Protocol Analyzer

William Grant Grovenburg

Born in Nowata, Oklahoma, Grant Grovenburg received a BSEE degree from the University of Oklahoma in 1972 and an MSEE degree from San Jose State University in 1978. At HP since 1973, he has been a member of the design team for the HP 3000 Series 64 Computer and for the HP 4953A Protocol Analyzer. Grant is married, has a daughter, and lives in Colorado Springs, Colorado. He enjoys softball, singing, playing his guitar, and using his home computer.

Aileen C. Appleyard

Born in Leeds, England, Aileen Appleyard received a BS degree from Nottingham University in 1969 and an MS degree from Heriot-Watt University in 1973. After joining HP's South Queensferry Division in 1970 she worked on pulse code modulation test equipment. Later, at the Colorado Telecommunications Division, she was the software project manager for the HP 4953A Protocol Analyzer. Aileen is now a resident of Manitou Springs, Colorado, and continues to manage HP projects.

Wayne M. Angevine

Wayne Angevine is a graduate of the University of Colorado (BSEE 1980) and of the University of Southern California (MSEE 1981). Before coming to HP in 1983, he worked in the areas of computer-aided design, IC reliability, and image processing. At HP he has contributed to the development of the HP 4953A and to logic design for engineering workstations. Wayne was born in Boulder, Colorado, is married, and now lives in Colorado Springs, Colorado. He likes model sailplanes, cross-country skiing, hiking, and choral music.

Roger W. Ruhnow

At HP since 1976, Roger Ruhnow contributed to the early hardware design of the HP 3000 Series 32 Computer and to the development of the HP 300 Business Computer. He was one of the hardware architects for the HP 4953A Protocol Analyzer and also worked on gate array development and system software. He was born in Port Byron, Illinois and attended Bradley University (BSEE 1975) and the University of Illinois (MSEE 1976). Roger lives in Colorado Springs, Colorado and is active in his church. His hobbies include woodworking, photography, music, hiking, and gardening.

12 Protocol Analysis Architecture

Roger W. Ruhnow

Author's biography appears elsewhere in this section.

Stephen H. Witt

Steve Witt graduated from Michigan Technological University with a BSEE degree in 1979 and came to HP the same year. He worked on the I/O subsystem for the HP 300 Business Computer and later designed hardware and software for the HP 4955A Protocol Analyzer. Recently, he was R&D project manager for hardware development for the HP 4953A. Steve was born in Ottawa, Canada and now lives in Colorado Springs, Colorado with his wife, who is expecting their first child. He is active in the United Way organization and in his church. His leisure activities include photography, swimming, bicycling, reading, and softball.

18 Data Acquisition and Simulation

Elizabeth Gates Moore

Beth Moore was born in Wurzberg, Germany, and lived in many places in the U.S.A. and Europe as part of a military family. She attended La Salle College, from which she earned a BA degree in mathematics and computer science in 1979. After coming to HP the same year, she worked in the quality assurance departments of two different divisions, then contributed to the design of the HP 4955A and HP 4953A Protocol Analyzers. Beth lives in Colorado Springs, Colorado, is married to another HP engineer, and is expecting their first child. She is a member of the Society of Women Engineers, and likes golf, needlework, and soccer.

David B. Karlin

Born in Chicago, Illinois, Dave Karlin studied electrical engineering at Cornell University (BSEE 1980) and at Stanford University (MSEE 1984). Since coming to HP in 1980, he has worked on data link control software design for the HP 4955A Protocol Analyzer and on software and hardware design for the HP 4953A Protocol Analyzer. He is now a project manager and a member of the IEEE. Dave lives in Colorado Springs, Colorado and likes skiing, softball, and soccer.

Mark D. Keisling

A native of Midland, Michigan, Mark Keisling earned his BSEE degree from Michigan Technological University in 1981 and came to HP the same year. Much of his work at HP has been in the area of software quality assurance for the HP 4955A Protocol Analyzer. Mark lives in Colorado Springs, Colorado, is interested in television production, and enjoys volleyball.

Dorothy J. Yackle

A project manager at HP's Colorado Telecommunications Division, Dotty Yackle has been at HP since 1979. She was an application engineer for the HP 64000 Logic Development System and software engineer for the HP 4953A and HP 4955A Protocol Analyzers.

Dotty was born in Bethpage, New York and attended both the University of California at Santa Barbara and the University of Colorado at Colorado Springs. She earned a BSEE degree in 1979 from the latter. She now lives in Colorado Springs, Colorado with her husband and son and enjoys bicycling and playing soccer and softball.

Vonn L. Black

Vonn Black was born in Pomona, California and graduated from Brigham Young University with a BSEE degree in 1981. After joining HP's Colorado Telecommunications Division the same year, he designed the power supply, DLC, and tape controller for the HP 4951A. Vonn, his wife, and three children live in Colorado Springs, Colorado. He is an active church member and an avid BYU fan. He enjoys competitive volleyball and vacationing in Mexico City with his family.

Stephen B. Tursich

Steve Tursich was born in Tonawanda, New York and earned bachelor's degrees in computer science and electrical engineering from the University of Minnesota in 1982. Since coming to HP that same year, he has worked as a design engineer on the HP 4951A Protocol Analyzer. Steve is a member of the IEEE and of the ACM and is interested in hardware/software interfaces and user interfaces. He lives in Black Forest, Colorado with his wife and enjoys electronics.

24 Low-Cost Analyzer

Alan Delwiche

At HP since 1982, Alan Delwiche worked on software development for the HP 4951A Protocol Analyzer and is now a project manager for applications software for future products. He attended the University of California at Berkeley (BA 1969) and the University of Oregon (MS 1977). Before coming to HP, he developed software for applications in laboratory automation and in data communications. Alan was born in San Jose, California and now lives in Manitou Springs, Colorado. He is active in his church, enjoys spending time with his family, and likes skiing, running, backpacking, and reading.

Chris L. Odell

At HP since 1981,Chris Odell designed the digital and analog circuitry for the CRT and the RS-232-C and RS-449 interface modules for the HP 4951A Protocol Analyzer. He was born in Wenatchee, Washington, completed a BSEE degree at California Polytechnic State University at San Luis Obispo, and did work toward an MSEE degree at the same institution. Chris is married, has a son, and lives in Colorado Springs, Colorado. He is active in his church and enjoys spending time with his son. His hobbies include skiing, softball, volleyball, and bridge.

30 Semiconductor Process Control

Wesley H. Higaki

At HP since 1979, Wes Higaki wrote controller software for a diffusion furnace at HP Laboratories and later worked on the HP PC-10 Process Control System. He is now the PC-10 project manager, specializes in manufacturing automation, and is a member of the Semiconductor Equipment and Materials Institute. Wes is a native Californian who was born in Redwood City and attended the University of California at Davis (BS 1977) and the University of Santa Clara (MS 1980). He lives in Sunnyvale with his wife and twin sons and is interested in carpentry.

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

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

HEWLETT-PACKARD JOURNAL

July 1985 Volume 36 • Number 7

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 3000 Hanover Street
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
1181 KK Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

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

5953-8537