FEBRUARY 1978

HEWLETT-PACKARD JOURNAL

© Copr. 1949-1998 Hewlett-Packard Co.

A Logic State Analyzer for Evaluating Complex State Flow

Sequential triggering and selective trace are two of the capabilities that enable this 32-bit logic state analyzer to capture only the states of interest in complex program flow. It also counts states, and times their execution to help evaluate program performance.

by George A. Haag

THE FIRST LOGIC STATE ANALYZER was introduced in 1973,[1] marking the beginning of a new age for the logic designer. For the first time since the computer era began, he now had a tool to monitor state flow in digital systems.

Since then, advancing technology has increased the variety and complexity of logic devices, resulting in more complex forms of state flow and bus protocol. Consequently, a new logic state analyzer, Model 1610A (Fig. 1), has been designed to provide measurement facilities commensurate with these more complex requirements. Because it has application to a wide variety of microprocessors, to the many microprocessor peripheral activities with their more involved state sequences, and to minicomputers with their use of both macro- and microprocessor languages, Model 1610A may truly be called a general-purpose logic state analyzer.

The Need for a General-Purpose Instrument

Most contemporary logic designs employ the concepts of algorithmic state machines, most of which now incorporate microprocessors. The behavior of these machines is characterized by their state flow and the major application of logic state analyzers has been monitoring the address and data buses of these devices during investigations of system performance. However, programming for these machines now goes far beyond simple "in-line" code to include various forms of branches, loops, nested loops, subroutines, re-entrant routines, and recursive routines. Data structures in advanced designs include arrays, stacks, queues, and linked lists, and the machine architectures involve context switching and mapped and virtual memory schemes. Model 1610A has the program tracing capabilities needed for analyzing the performance of these more complex systems.

A microprocessor-based system may have only one to five chips directly involved with the CPU but the peripheral functions may involve up to several hundred integrated circuits. The controllers for keyboards, CRT displays, cassette and floppy disc drives, direct memory access, and the other functional units needed for a given task may use a variety of IC technologies, usually MOS, bipolar, and ECL.

Cover: *The instrument shown here is the HP Model 1615A Logic Analyzer, an instrument with a new capability for simultaneous logic state and timing analyses, described in the article beginning on page 14. The other articles in this issue describe further new developments in data-domain instrumentation for the fast-burgeoning world of digital electronics.*

In this Issue:

A Logic State Analyzer for Evaluating Complex State Flow, by George A. Haag ..... **page 2**

Viewpoints–Chuck House on the Ongoing Revolution in Digital Testing **page 11**

Interactive Logic State and Timing Analyses for Tracking Down Problems in Digital Systems, by John A. Scharrer, Robert G. Wickliff, Jr., and William D. Martin ..... **page 14**

Entry Level Logic State Analyzer Has High-Level Capability, by Charles T. Small and Alan J. DeVilbiss ..... **page 21**

Adapting the 1611A Logic State Analyzer to Work with the F8 Microprocessor Family, by Deborah J. Ogden **page 28**

© Hewlett-Packard Company, 1978

© Copr. 1949-1998 Hewlett-Packard Co.

Fig. 1. *Model 1610A Logic State Analyzer traces the flow of states up to 32 bits wide in minicomputers, microprocessor-based systems, and other sophisticated logic systems. The interactive display and keyboard simplify the establishment of highly selective protocol for capturing only the data of interest.*

Model 1610A has four 8-bit input pods, with independent threshold levels settable from −10V to +10V. These, plus flexible data formatting facilities, make Model 1610A adaptable to systems that use a variety of custom designs.

Sophisticated new ICs are arriving on the market to simplify the design of HP-IB interfaces, CRT controllers, mass-memory controllers, and many other systems. In addition, bit-slice microprocessors are being designed with faster, more powerful micro-coded processor architectures that use more involved state sequences. Model 1610A with its advanced triggering and flexible data formatting capabilities is readily applied to these areas.

Most minicomputers execute several micro-coded instructions to implement one instruction of a higher-level language. Tracing micro-code with a logic state analyzer usually involves the collection of simple instruction sets executed at a fast rate. Model 1610A's ability to count states or measure time is useful in optimizing micro-code performance. Tracing higher level code requires less speed but much greater selectivity, which Model 1610A also offers.

New Capabilities

Like other logic state analyzers, Model 1610A monitors the states (1's or 0's) of bus lines or other points in a digital system and stores a sequential series of states for display and examination. Triggering circuits initiate data capture at a point selected by the user in the program, enabling him to find out exactly what occurs in any part of the program as it is

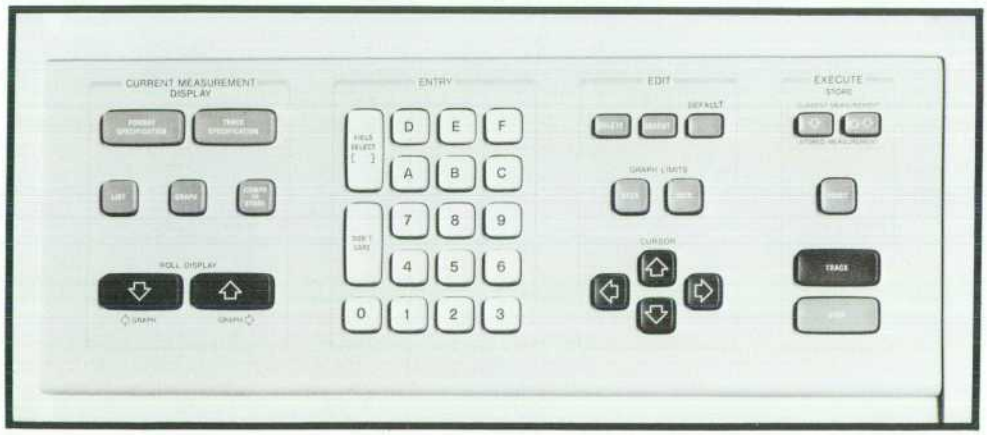


Fig. 2. *Keyboard control of Model 1610A was greatly simplified by using the display for much of the information that normally would appear on the front panel. The five keys in the* CURRENT MEASUREMENT DISPLAY *block at the left enable the selection of interactive "menus" that direct the operator to enter appropriate variables.*

© Copr. 1949-1998 Hewlett-Packard Co.

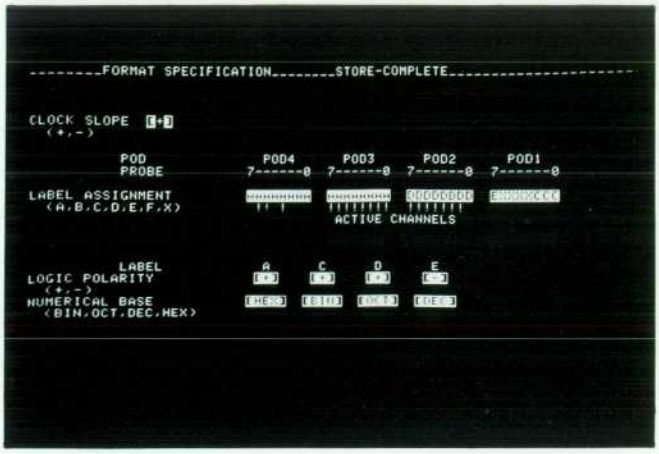


Fig. 3. *FORMAT SPECIFICATION menu enables the operator to specify the logic polarity of the data input and the numerical base assigned to sets of input lines, as grouped by assignment of labels. An exclamation point under a probe marker indicates that that probe is actively sensing transitions. Inverse video (black on white) indicates where the operator is to make entries.*

actually being executed.

What distinguishes Model 1610A from its predecessors is its capability for sequential triggering, selective trace, sequence restart, and time and state count. With much microprocessor programming now including various forms of branches, loops, nested loops, and subroutines, Model 1610A's sequential triggering capabilities enable it to hold off data capture until an executing program passes through the particular branch or loop of concern. This is done by specifying a series of steps in the program that must be encountered in the specified order before the analyzer starts to gather data. Furthermore, the user can specify a certain number of times that each step must occur, such as in a loop, before going on to look for the next word in the sequence.

Model 1610A also has a way of eliminating the capture of much unnecessary data: selective trace. To use this mode, the user can specify up to seven states (addresses, data, or whatever) and the analyzer will then store only those states as they occur during program execution, and ignore the others. A long sequence can thus be condensed into a shorter one. Furthermore, the user can select a range of states for storage by inserting X's (don't cares) into any of the digit positions. For example, if a state were entered as $A5XX_{16}$ in selective trace, then any of the states within the range $A500_{16}$ to $A5FF_{16}$ would be stored for display as they occur. Thus, data capture can be restricted for example, to only those instructions addressed to a particular peripheral.

Another useful capability designed into the 1610A is the ability to measure the time intervals that occur between program steps acquired for display. By providing information on how much time a program spends in loops or in servicing interrupts, the time-measurement capability is a useful aid in performance monitoring and program optimization. The time intervals are measured and stored simultaneously with the state sequences and are available for display as either state-to-state time intervals (relative time) or total accumulated time with respect to the trace point (absolute time).

The counting capability can also be applied to the number of states occurring between displayed states. This helps determine whether the program is spending time on non-essential activities while carrying out the desired task.

A new feature that can be particularly helpful in evaluating program performance is the 1610A's ability to present a graph of state magnitude versus time, which gives an easily-interpreted overview of program execution (see Fig. 6). Each dot in the display represents one program step with its vertical displacement proportional to the numerical magnitude represented by the state and its horizontal position determined by its order of occurrence. The occurrence of branches, loops, or any other departures from "in-line" code are immediately apparent.

An Overview

Model 1610A has 32 high-impedance inputs, arranged in groups of eight on four pods (a fifth pod is provided for the clocking input). Data can be input at clock rates up to 10 MHz. Worst-case set-up/hold time is 20/0 ns on all 32 input channels, enabling Model 1610A to monitor state flow in minicomputers and peripherals, as well as in microprocessors and the sophisticated new random-logic chips now arriving on the market.

Up to 64 words 32 bits wide can be stored at a time. Roll keys enable the user to display any consecutive

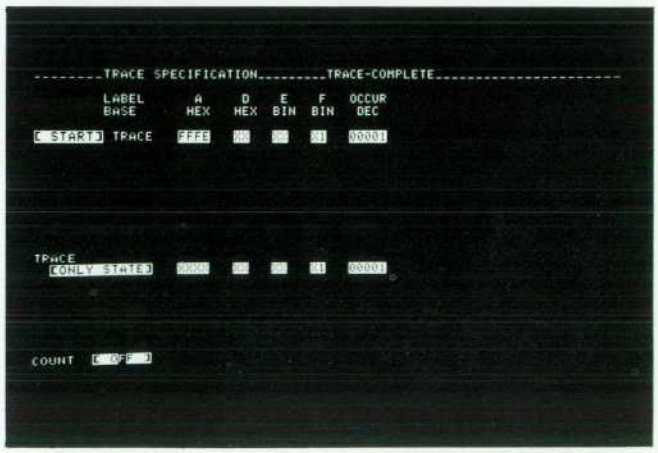


Fig. 4. *TRACE SPECIFICATION menu helps the operator define where in the program data acquisition is to start. In this example, data acquisition begins when the state FFFE occurs on the inputs labelled A and the least-significant bit of F is 1.*

© Copr. 1949-1998 Hewlett-Packard Co.

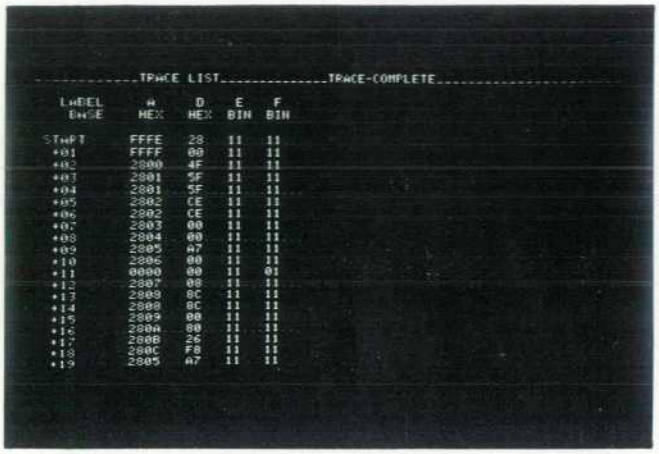


Fig. 5. *TRACE LIST menu displays the data that is captured and stored following the occurrence of the trigger state, in this case FFFE as specified in the TRACE SPECIFICATION menu of Fig. 4. The data is displayed in the numerical base selected for each of the labelled groups of inputs. The ROLL DISPLAY keys enable any consecutive 20 of the 64 captured states to be brought up on the display for viewing.*

20 of the stored words for examination. For ease in reading the digital data captured by Model 1610A, the data may be formatted in octal, decimal, or hexadecimal form for display, as well as in the binary form in which it is acquired. A printer output that interfaces to Model 9866A/B thermal line printers enables specifications and results displayed on the CRT to be documented at the push of a button (see Fig. 8).

Menu Control of an Instrument

A major challenge facing the designers of this instrument was how to provide the user with the means of controlling all this capability. The use of a key-per-function arrangement would have resulted in an overly complicated keyboard. A "menu" control approach with directive displays and a simpler keyboard is used to eliminate the complexity.

Referring to Fig. 2, the user selects a menu by pressing one of the keys within the CURRENT MEASUREMENT DISPLAY group at the left of the keyboard. As an example, suppose he pressed the FORMAT SPECIFICATION key. The resulting display would be as shown in Fig. 3.

The top line of the display identifies the menu selected and the current state of the instrument (TRACE COMPLETE). The second line contains any messages for the user concerning incorrect operation or undesirable external conditions (WARNING - SLOW CLOCK). The blocks in inverse video (black on white) indicate where entries are to be made from the ENTRY group of keys. Annotation adjacent to the inverse video fields explains the meaning of each field and lists the choices.

The user inputs data into an entry field by first selecting the field with the CURSOR keys in the EDIT section of the keyboard. These move a blinking cursor to the desired entry field.

Brackets in an entry field indicate that the input to this field is controlled by the FIELD SELECT key in the ENTRY group. Pressing this key causes the entry field identified by the cursor to cycle through its allowable choices. For example, in the CLOCK SLOPE field, the FIELD SELECT key cycles the entry field through "+" (positive edge) and "−" (negative edge). The FIELD SELECT key thus functions as a selector switch and, in conjunction with the CURSOR keys, replaces the many keys that would be required in a key-per-function implementation.

The FORMAT SPECIFICATION menu is used to format the acquired data to suit a variety of applications. All the data input probes are represented on the display, grouped according to their respective pods (see Fig. 3). The probes are connected to a bus in the system under test and labels can be assigned to identify the parameter monitored by each probe. In Fig. 3, A's were entered consecutively on all probes of pods 3 and 4 to indicate that these 16 bits are to be treated as one combined variable, the address bus of a microprocessor. Another parameter is defined by D's, the data bus, on the 8 bits of pod 2. Up to six different dummy labels, from A to F, may be used to segment the 32 data channels into parameters. Assignments may be specified without regard for pod boundaries as long as labels are defined along consecutive probe positions. Those probes that aren't used may be turned off by entering X's in their position with the

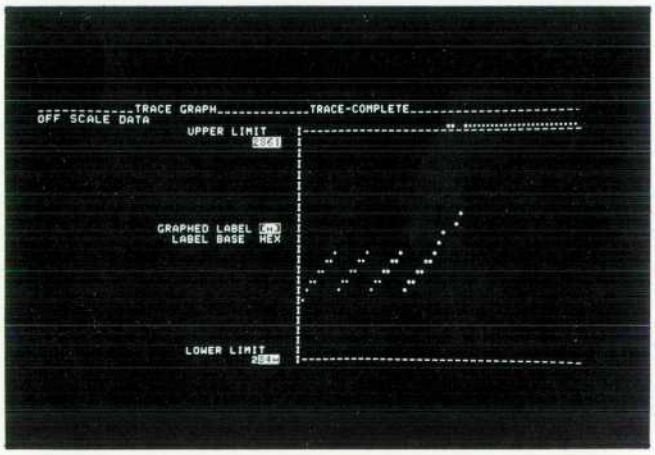


Fig. 6. *TRACE GRAPH menu gives an overview of all 64 states captured in the analyzer's memory, each state being represented by a dot whose vertical position corresponds to its numerical value and horizontal position to its order of occurrence. Departures from a straight line show where the program loops and branches. The GRAPH LIMITS keys allow the upper and lower limits to be set so the vertical scale can be expanded for comprehensive viewing. States outside this range are displayed in a horizontal row above or below the dashed lines. The brightened dots represent the states that will be displayed when the analyzer is switched to the TRACE LIST mode.*

© Copr. 1949-1998 Hewlett-Packard Co.

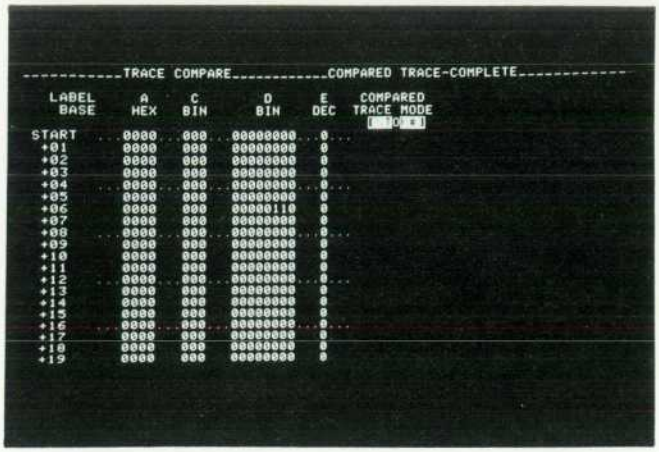


Fig. 7. *TRACE COMPARE mode compares a stored trace listing to incoming data, displaying a non-zero where bits differ as in step 06 of this program. The analyzer can also be directed to rerun a measurement continuously and stop when the current and stored traces are either equal or not equal, making it easier to capture intermittent problems.*

DON'T CARE key so they will not be displayed in the trace listing.

The menu is completed by selecting the desired logic polarity and numerical base for each assigned label. A "+" logic polarity indicates that the variable is strobed high true while a "−" indicates low true. Selecting a numerical base (binary, octal, decimal, or hexadecimal) defines the radix of the number system that will be used to specify and display the data for that label.

Other menus are shown in Figs. 4, 5, 6, and 7. The TRACE SPECIFICATION menu (Fig. 4) enables the user to establish the trigger conditions for acquiring data and provides a control overview of the trace and count functions. Following specification of the data format and trigger condition, the user can press the TRACE key in the EXECUTE group, and data is stored starting with the state in the program flow that meets the trigger conditions. The stored data may then be displayed as a program listing (Fig. 5) or as a graph (Fig. 6), or it may be stored for comparison with data to be acquired later (Fig. 7), such as comparing the playback from a disc memory to the original data.

Rather than start the trace when trigger conditions are met, the user can elect to "center" the trace, where the 31 points prior to and the 32 states following the trace point (trigger) are captured for display, or to "end" the trace, where the 63 states prior to the trace point are captured.

How It Is Used

The capabilities of Model 1610A can be illustrated by a few examples. Consider the branched-code flow diagram of Fig. 8. The state flow in either path A or path B down to address 2942 is "in-line" code and is traceable by any logic state analyzer with a single trigger state in the appropriate branch. However, the state sequence following 2942 may differ according to which branch was traversed. To trace this part of the program following a particular branch, triggering on a known state in the branch and then setting the digital delay to start data capture at 2942 might work, but there are two difficulties to be considered. First, the path length in terms of clock pulses is rarely known to the user because he is seldom aware of the number of states involved in executing program algorithms. Second, the path length itself is often variable, depending on the number of wait loops, interrupts, data-dependent loops, and so on.

By enabling path-dependent tracing, the sequential triggering capability of Model 1610A overcomes these difficulties. As shown by the trace list in Fig. 8, Model 1610A first finds address 28AF and then starts the trace at 2942, thus assuring that the common segment of code is traced only when the program has passed through path A. The path length may vary, but the start of the trace is constant. The count of states in the trace listing of Fig. 8 shows that 81 states were executed between 28AF and 2942 during that pass.

This example had only one branch. Most algorithms contain a multitude of decision branches,

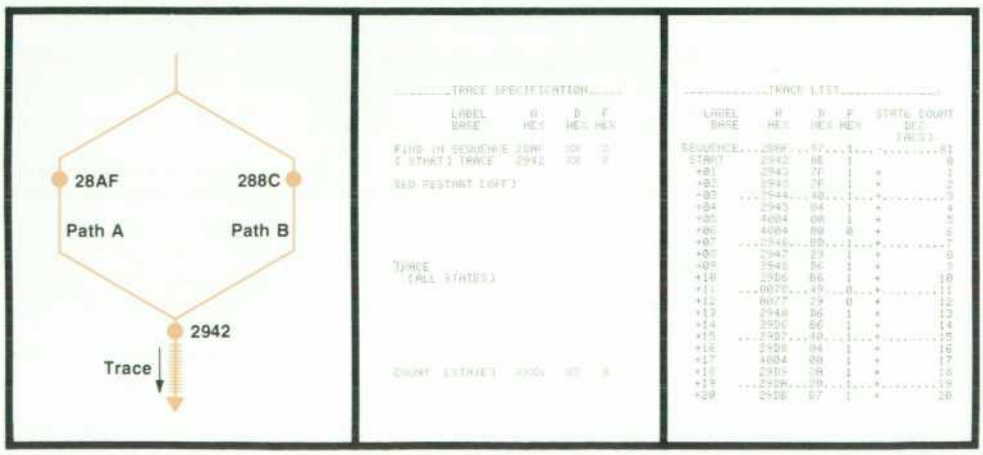


Fig. 8. *The program in the common leg beginning with state 2942 may differ according to which path the program followed. The printout of the TRACE LIST shows how sequential triggering was used by Model 1610A to capture data beginning with state 2942 only when the program followed path A.*

© Copr. 1949-1998 Hewlett-Packard Co.

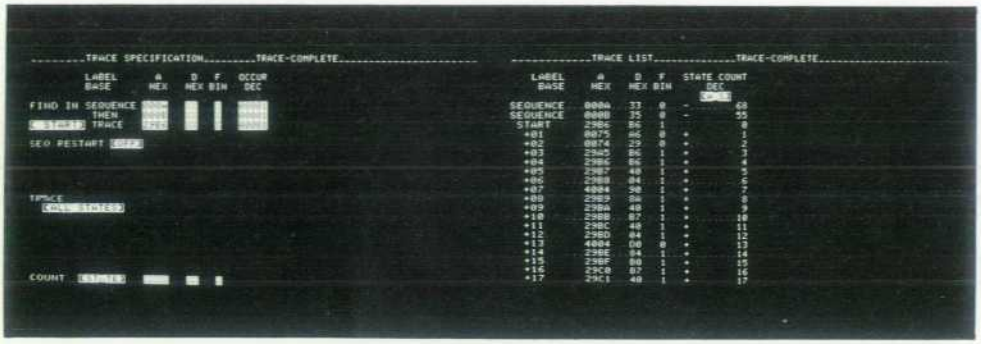


Fig. 9. *This example shows how variables stored at various addresses before a mathematical routine can be included in a record by making these addresses part of a trigger sequence. In this example, the number 33 is multiplied by 35 in the routine beginning at state 29B6.*

perhaps as many as one branch for every five instructions. To analyze these more typical cases, Model 1610A provides as many as seven terms in a tracepoint sequence. (Terms are added by use of the INSERT key in the EDIT section of the keyboard.)

Sequential triggering is useful in other ways. For example, consider data-dependent procedures such as a multiply routine whose behavior depends on the values it is given to multiply. These values are often loaded into registers or memory well before the routine is called, but without knowing what they are, the user can't tell if the multiply activity traced by a logic state analyzer is appropriate to those particular values.

With the 1610A, the trigger sequence can be specified to first search for and find the addresses where the values are stored, and then start the trace where the multiply routine starts. The values thus become part of the trace record, as shown in Fig. 9.

Occurrence

Fig. 10 depicts a common situation where a loop is nested within another loop. Model 1610A has an "occurrence" feature that enables it to pick out any pass through any loop for tracing. It does this by requiring a given state condition to be satisfied repeatedly a given number of times before it advances to the next line in the trace specification.

Referring to the example in Fig. 10, suppose the J loop completes 11 passes each time it is called by the I loop, and the I loop completes 17 passes before exiting to address 28CE, and suppose it is desired to trace the eighth pass of the J loop while it is in the fifth pass of the I loop. The trace specification (Fig. 10) causes the instrument to find in sequence state 28B7, then eight occurrences of 28AE followed by five occurrences of 28A5. The number 8 (decimal) is entered into the occurrence field for 28AE and 5 was entered into the occurrence field for 28A5. The state count in the resulting trace list (Fig. 10) verifies that $8 \times 11 + 5 = 93$ passes through the inner loop occurred before the trigger sequence was satisfied.

The range of occurrence counts is from 1 to 65,536. Occurrence is actually a more general form of the digital delay offered in earlier logic state analyzers. Digital delay may be specified for the 1610A by adding another state, consisting of all X's (i.e., any state), to the sequential trigger specification and entering the desired delay into the occurrence field for that state. In most cases, however, state occurrence is found to be more useful than digital delay.

Sequence Restart

Not every branched or looped problem can be analyzed simply by the use of sequential triggering and occurrence. Consider the flow diagram of Fig. 11. The dashed line indicates a "zero length" path where the branch from 2450 to 2452 is direct with no intervening states. A typical routine might be: if the branch carry set instruction (2450) is met, jump di-

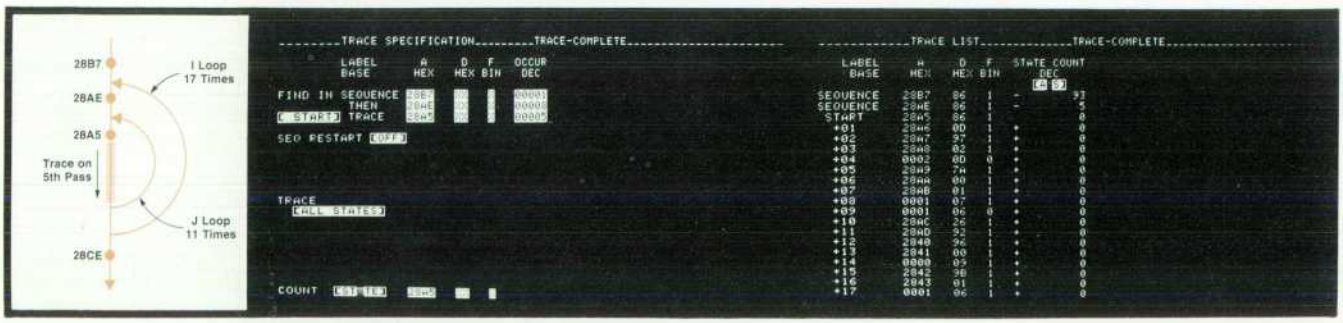


Fig. 10. *Any pass in a loop nested within other loops can be captured for display by using the OCCURRENCE feature in conjunction with sequential triggering. In this example, tracing begins with the eighth pass of the J loop during the fifth pass of the I loop.*

© Copr. 1949-1998 Hewlett-Packard Co.

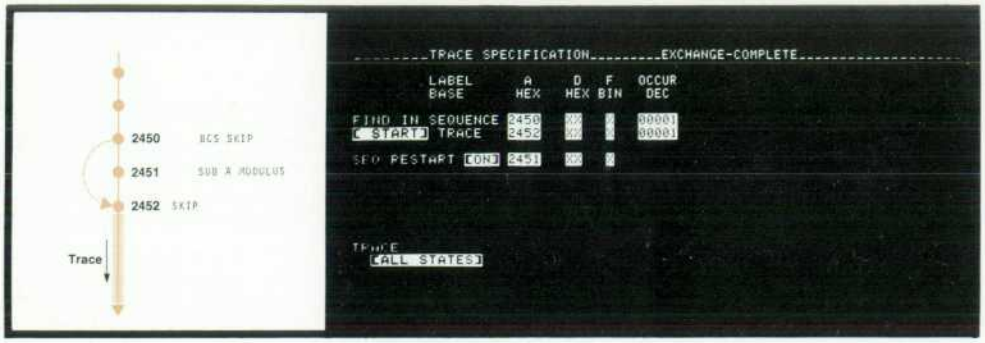


Fig. 11. *A trace beginning at state 2452 only when the program jumps directly from 2450 to 2452 can be assured by using* SEQUENCE RESTART *to abort the trigger sequence if 2451 occurs before 2452.*

rectly to SKIP (2452); otherwise execute the subtract instruction (2451) then go to SKIP. How, then, can sequential triggering be used to start a trace at 2452 only when the zero-length path is taken?

The sequence restart capability of Model 1610A resolves this problem. Sequence restart allows the entry of a state that will cause the trigger sequence to be restarted if that state occurs before the trigger sequence has been completed. In the example of Fig. 11, the trigger sequence could be: find 2450 then start the trace on 2452. With 2451 entered into the sequence restart field, if 2450 had been found and 2451 occurred before 2452, the sequence would be aborted and the analyzer would begin anew to look for 2450, starting a new trigger sequence.

The SEQ RESTART field appears in the TRACE SPECIFICATION menu whenever a second state is entered into the trigger specification. Normally it is in the OFF state (see Fig. 8) but can be turned ON with the FIELD SELECT key. The restart state may then be entered.

Fig. 12 shows how sequence restart applies to analyses of loops. In this example, it was desired to find out how much time elapses between the start and end points when traversing path 2. The path is described in the trace specification by finding 2875, then 28B3, and start the trace at 29E4. However, without sequence restart, each state in the sequence could be satisfied on a different pass through the loop. For example, state 2875 could be acquired at the beginning of a cycle through path 1 while 28B3 and 29E4 are acquired during a subsequent cycle through path 2, giving an erroneous count between 2875 and 29E4. With 29E4 entered as the restart condition, the trigger sequence would be restarted at the end of any cycle that branched through path 1. The resultant trace list (Fig. 12) shows that the correct path required 148 μs from start to end. In general, restarting on a state that occurs at the beginning or end of an algorithm forces a single-path solution.

This use of sequential trigger and sequence restart enables the length of a path through an algorithm to be measured directly. Both the time count and the state count have a 32-bit range (4.3×10^9) counts. Time values are measured with a resolution of 100 ns and are displayed with four-digit resolution in units of μs, ms, or s.

The restart feature has broad application. For example, it can trace a transmission on the HP interface bus that addresses device 10 to talk and device 5 to listen before the unlisten command is given (restart on unlisten). In a virtual memory environment (swapping programs with disc), a trace specification can find the trace point of interest within the program when swapped into memory, and restart if it is ever swapped out. In short, the instrument's analysis capabilities are available to start, center, or end a trace in virtually any form of state flow.

Selective Trace

Fig. 13, a memory map of a microprocessor routine, helps illustrate the use of selective trace. Up to seven state conditions can be specified and if a monitored

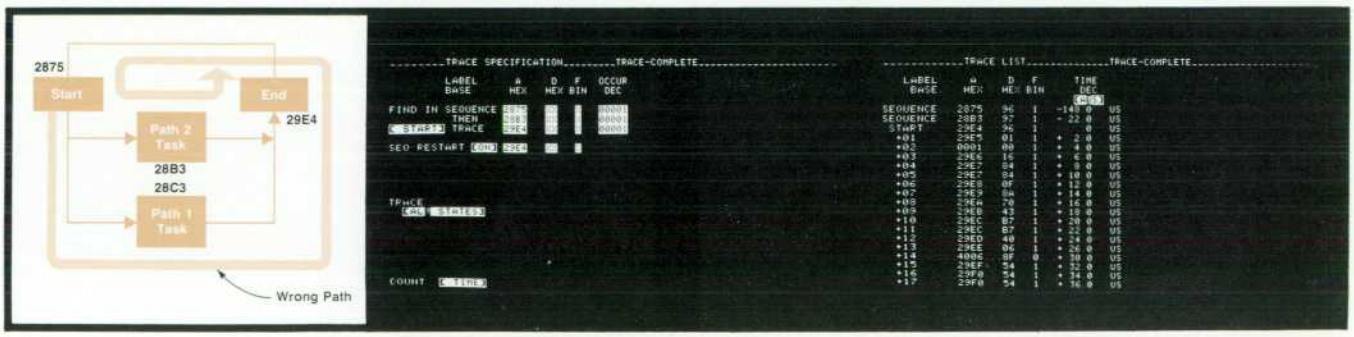


Fig. 12. SEQUENCE RESTART *can prevent an erroneous measurement by aborting the trigger sequence any time the program branches into path 1.*

© Copr. 1949-1998 Hewlett-Packard Co.

Fig. 13. *Selective tracing captures only significant information and eliminates insignificant detail. The time measurements at lines 11 and 13 indicate subroutine calls were executed. Details of these subroutines were excluded from the trace list by the selective trace.*

state meets any of these conditions, it will be included in the measurement. In general, selective tracing conserves the trace memory, allowing pertinent data to be gathered over an arbitrarily large time interval. Only important states are collected while all others are discarded. With the incidentals removed, trace listings are easier to read and comprehend.

The routine of Fig. 13 represents one of many routines in a total program and is traced starting at its entry point (2942) when called by 29FF. With selective trace, specifying 294X allows routine instructions from 2940 to 294F to be collected. Specifying 40XX collects the routine's output data written to a file. The remaining conditions specified allow the collection of a few key variables that represent inputs to the routine. This is all that is necessary to understand the routine's behavior.

What is significant is what this selective trace did *not* collect. The trace memory was spared from collecting several unnecessary variable references, memory-consuming wait loops and handshakes, state flow necessary to execute called subroutines, stack references, and random interrupt transactions. The count function, by recording the time between traced states, indicates where states were omitted (this often leads to discovering excessive time spent in a malfunctioning procedure!).

The selective trace technique lends itself to characterizing the behavior of procedures as "black box" transfer functions in the same way that the output of an electronic circuit as a function of its input characterizes the behavior of the circuit. Tracing only the inputs and outputs of a procedure can totally describe its behavior while collecting only a small fraction of the total number of states. Conversely, using selective trace to trace only key states of major routines can describe an overview of state flow between all routines as a whole. To illustrate, Fig. 14 shows another memory map. To evaluate arithmetic expressions, the main program calls the various subroutines while executing. Suppose it is desired to monitor the order in which the routines are called while various expressions are evaluated, and find the time spent in each routine. The 1610A is asked to start the trace upon reading the first expression, beginning at address 29B3. Then the selective trace feature is employed to trace only the entry point to each routine (2951, 2969, or 297F). Since these states occur only once per call, the trace memory does not become filled with unwanted details of each routine's execution. Finally, a simultaneous count of time is specified to measure the path length through each routine.

The resulting trace list (Fig. 14) shows the order to

Fig. 14. *Selective trace can be used to trace program flow between major program segments and the order in which subroutines are called following initiation of program execution at address 29B3. The* COUNT *feature discloses how much time is spent in each subroutine.*

© Copr. 1949-1998 Hewlett-Packard Co.

be as expected. It also shows that the divide and multiply routines take nearly as long as the exponentiate routine, contrary to expectations. Detailed analyses of the divide and multiply routines might therefore be performed to identify which internal procedures or loops might be refined to shorten their execution times.

Capturing Every nth State

Another way to encompass a long state sequence within the analyzer's finite memory is to use the OCCURRENCE field in the TRACE ONLY STATE specification (see Fig. 4). For example, if the number 3 were entered in this field, and the TRACE ONLY STATE specification were filled with Xs (don't cares), the instrument would store every third state, effectively compressing a 192-state sequence into the 64-state memory.

This capability is especially useful with the GRAPH display (Fig. 6) as it then provides an overview of state sequences longer than 64. This is comparable to the way that slowing the sweep rate of an oscilloscope compresses a displayed waveform to place more of the waveform on screen although slowing the sweep rate may obscure fine detail. When using the OCCURRENCE feature with the graph display, the major details of the graph are retained if the number of states between acquired states is not too great.

In Summary

Space does not permit further discussions of the TRACE GRAPH and TRACE COMPARE functions (Figs. 6 and 7) but suffice it to say that Model 1610A was developed to suit the ever-changing and increasingly complicated needs of the digital hardware and software designer. Analysis features such as sequential trigger, occurrence, sequence restart, selective trace, and simultaneous count of state or time offer the power necessary to treat complex state flow in microprocessors and minicomputers. Yet operator interface remains simple because of the menu control concept. These features offer the designer a reduction in development time with measurement applications limited mainly by his imagination.

Acknowledgments

Gordon Greenley shared the software development task. Jim Donnelly and Steve Shepard designed the high-speed data acquisition system, Paul Sherwood engineered the processor, I/O, and performance verification, and Guy Howard contributed the probes, CRT drive circuits and the power supply design. Product design was by Don Skarke.

Reference

1. W.A. Farnbach, "The Logic State Analyzer—Displaying Complex Digital Processes in Understandable Form," Hewlett-Packard Journal, January 1974.

SPECIFICATIONS
HP Model 1610A Logic State Analyzer

CLOCK AND DATA INPUTS

REPETITION RATE: to 10 MHz.
INPUT RC: 50 kΩ shunted by ≤14 pF at probe tip.
INPUT BIAS CURRENT: ≤20 μA.
INPUT THRESHOLD: TTL, fixed at approximately +1.5 V; variable, ±10 Vdc.
MAXIMUM INPUT: −15 V to +15 V.
MINIMUM INPUT
SWING: 0.5 V.
CLOCK PULSE WIDTH: 20 ns at threshold level.
DATA SETUP TIME: 20 ns.
HOLD TIME: 0 ns.

TRIGGER AND MEASUREMENT ENABLE OUTPUTS

TRIGGER OUTPUT (rear panel): 50 ns ±10 ns positive TTL level trigger pulse is generated each time trace position is recognized. If trace position includes a word sequence; pulse occurs when last word is found. Trigger outputs continue until a new specification is traced or STOP key is pressed. Pulse rep-rate is 0 to 10 MHz depending on input data rates. In continuous or compared trace modes, internal display process blanks out pulses for 100 μs at rep-rates of <20 Hz.

MEASUREMENT ENABLE OUTPUT (rear panel): Positive TTL level measurement enable output goes high and remains high when 1610A is looking for trace position and goes low when trace position is recognized or if STOP key is pressed. In continuous or compared trace modes transitions repeat each time the 1610A makes a new measurement.

MEMORY DEPTH: 64 data transactions; 20 transactions are displayed on screen. Roll keys permit viewing all 64 data transactions.

TIME INTERVAL: Resolution, 100 ns; accuracy, 0.01%. Maximum time, 429.4 seconds.

EVENTS COUNT: 0 to $2^{32}-1$ events.

DIMENSIONS: 230 mm H × 425 mm W × 752 mm D (9.063 × 16.75 × 29.625 in.).

WEIGHT: 26.5 kg (58.5 lb).

ACCESSORIES SUPPLIED: four 10248A data probes, one 10247A clock probe.

PRICE IN U.S.A.: $9500.

MANUFACTURING DIVISION: COLORADO SPRINGS DIVISION
P.O. Box 2197
Colorado Springs, Colorado 80901 U.S.A.

George A. Haag

Raised in Colorado, George Haag obtained a BSEE degree from Colorado State University in 1968, then joined HP's San Diego Division. He worked on the logic and servo design of the 7200A Time-Share Plotter, was project leader on the 9125B Calculator Plotter, and did the software and logic design for the 9862A Calculator Plotter. He transferred to the Colorado Springs Division in 1972 and started software development for the 1610A, becoming project leader in 1975. He is now a section leader. George designed and built his own home, doing just about everything but the foundation and framing, but still finds time to participate in motorcycle scrambles and take his family skiing (he has a wife and two children, 3 and 6).

© Copr. 1949-1998 Hewlett-Packard Co.

Viewpoints

Chuck House on the Ongoing Revolution in Digital Testing

The logic analyzers described in this issue represent the latest step in the evolution of a new breed of diagnostic test equipment for digital system troubleshooting. Logic analyzer concepts have been discussed before in these pages[1,2,3] and by now it is generally acknowledged that logic analyzers are a powerful contribution for design analysis and troubleshooting of digital systems.

The way in which these instruments have evolved is very interesting, and the implications are widespread for many engineers, technicians, and even company managements. For this much is clear: the world of electronics is on the threshold of a profound revolution in the nature of design tasks and diagnostic requirements, not to mention the opportunities and challenges in new product development.

The nature of these changes may be illustrated by considering the interplay between component suppliers, end-product manufacturers, users, and test equipment suppliers. In the 1960s, virtually all component vendors supplied classical RLC parts along with discrete semiconductors. The end-product manufacturer employed many circuit designers, equipped almost entirely with analog circuit skills, for the design and development of the end product. Users varied—many were technically skilled, since engineers often built products for other engineering groups (e.g., communications and military electronics). There was, of course, a consumer electronics segment where the users were largely nontechnical and the service groups were semi-skilled.

For all of these groups, the classical analysis techniques seemed appropriate. Those techniques, as we all know, were the frequency domain and time domain approaches pioneered by the mathematics of Maxwell, Fourier, Heaviside, and LaPlace. Voltmeters, oscilloscopes, counters, and spectrum analyzers are but a few of the many test instruments that have evolved to facilitate analysis by these techniques. Significantly, these instruments were basically the same whether used in a lab, production, or field-service environment—differing to be sure in portability, cost, accuracy, and sophistication, but not in terms of the parametric test method.

The advent of integrated circuits and minicomputers led to a significant expansion of computer-based system design opportunities, and eventually to a modification of the product cycle relationships. The first difference to note is the addition of a "systems-design" block to the components-to-user-chain, an admission of the importance of applications software design, I/O considerations, and support peripherals. Note that these designers were not very often designing hardware, except for translators, buffers, and other circuits needed to overcome a basic interface mismatch between two system parts. Designers and troubleshooters frequently found themselves spending long hours debugging software, searching for glitches, or checking for noise margins. To ease their problems, they resorted to using one minicomputer to develop another—in-house groups built simulators, and even assemblers and editors to help them develop applications software more quickly. For field service, diagnostic routines became imbedded in the software.

In effect, a new class of instrumentation was being defined as a function of need. This need was almost transparent to the outside world, because in large part electronic engineers not at a minicomputer company were still designing circuits on a nodal basis, and test equipment used by designers and computer service technicians still included voltmeters, counters, and scopes. In fact, scopes were still the primary digital troubleshooting tools after the diagnostic routine indicated that a problem existed.

But such inefficient tools! The errors in computers are largely errors of signal data flow. A flag line fails to set at the right time, the memory address is read incorrectly, the wrong instruction is executed, the data being massaged is transmitted with a dropped bit—these are not electrical parameter failures, except that they are incidently accomplished with electronic circuits. They are data errors, occurring because of an incorrect data sequence, and as such their analysis is more appropriately considered data domain analysis.

Into the Data Domain

The characterization of the data domain analysis concepts at HP came slowly during the late 1960's, based on a conviction that classical test instruments were largely irrelevant to the design and test requirements of the rapidly growing computer industry. That view was shaped by our own evolving design experience with desk-top computers, hand-held calculators, and minicomputer controller systems.

Nodal testers were the earliest aids for design and troubleshooting of digital systems, and they are still the least expensive. By the early 1970's the logic probes, clips, comparators, current tracers, and totalizing counters of HP's Santa Clara Division had become widely used to locate stuck nodes, pulse activity, shorts and opens, and pulse burst counts.[4,5,6,7] The time-space information content of digital signals is so important, however, that these simple tools have been augmented by more precise techniques based upon error-correcting code theory.

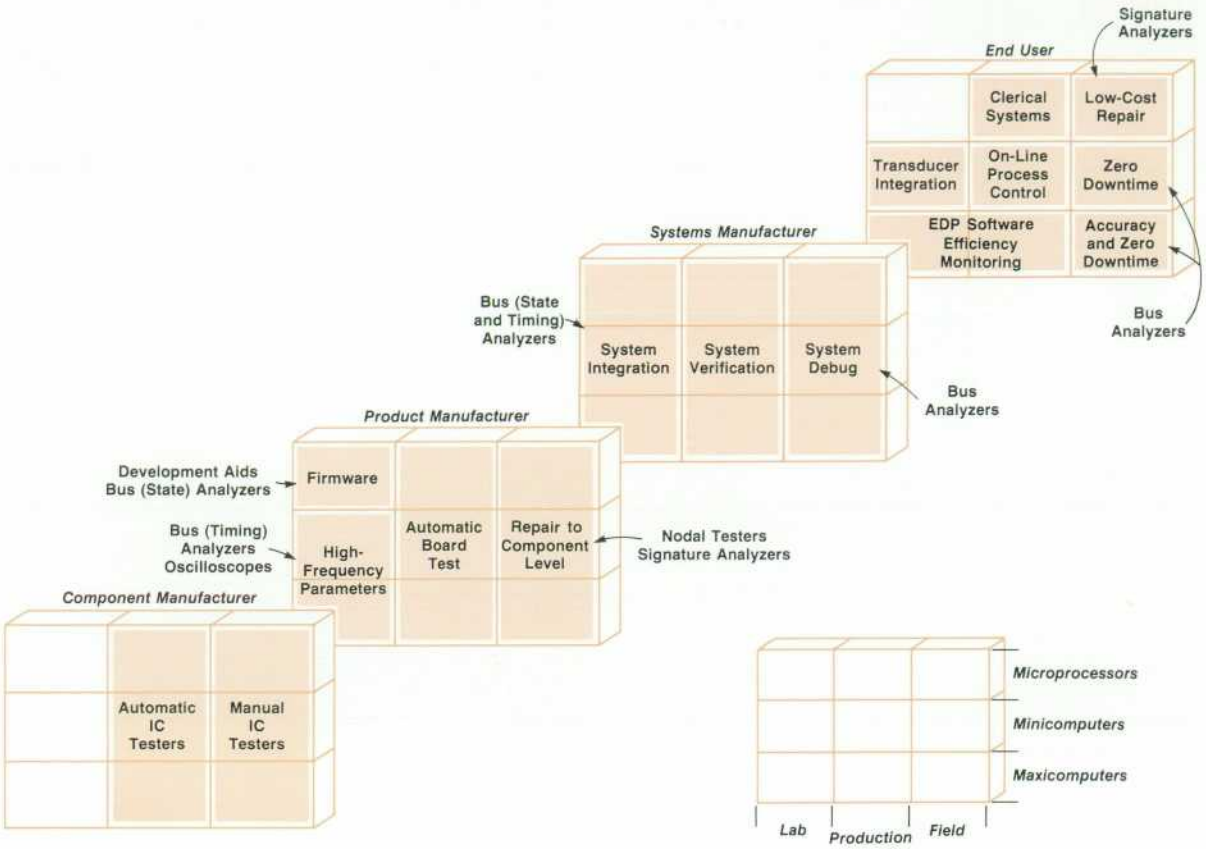
Perhaps the most powerful way of compressing large amounts of data is typified by the classical cyclic redundancy checksums (CRC) generated for error checking in large memory systems. "Signature analysis" techniques based upon CRC patterns are now appearing in instruments for data comparison analysis where large amounts of data must be monitored, collated and analyzed in a short time by an unskilled operator.[8] The technique requires some interactive work by the digital system designer to provide the proper signal checks and buffers from extraneous phenomena (such as electromechanical switch bounce), but the serviceability and analysis power of the technique usually far outweighs the initial design investment.

At HP's Colorado Springs Division, our initial thoughts were on building "digital scopes" but we went to considerable effort to explain that they were scope-like tools in terms of value for digital designers, rather than digitizing scopes. For our internal understanding, we began with a duality, trying to describe a function of word and event, f(W,E), analogous to the time domain function of volts and time, f(V,t). This allowed development of trigger word conditions that were analogous to the scope functions to which we were more accustomed.

We began also to view sampled data in a different context—sampling theory had always been used before at HP to develop a stroboscopic reconstruction of continuous high-frequency phenomena. The mathematics of state variables and algorithmic-state-machine (ASM) concepts became very popular topics for designers at HP around 1970 and 1971 and state-flow data table presentations are a clear outgrowth of this work. Thus, sampled-data theory moved from a relationship with time-domain analysis on continuous phenomena to data-domain analysis on discontinuous or truly discrete event-time phenomena.

An important consideration is the total amount of data that must be collected at every event-time in order to characterize system

© Copr. 1949-1998 Hewlett-Packard Co.



Legend for All Levels

behavior. The program counter, the instruction register, the accumulators, and so forth, contain specific coded data that collectively describe the machine status at any one event-time. In addition, most digital machines are built to operate on external data—to add, subtract, multiply, or divide data, to make branching decisions from data comparisons, and to accumulate, store and process still more data. Thus, to select data domain analysis as a descriptor for a way of analyzing digital logic machines suggests both an awareness of the machine's external function of working with digital data and its internal operation in terms of an organized flow of data sequences.

Data domain analysis, then, is a set of analysis techniques concerned with designing, monitoring, and correcting the behavior of a digital machine as a function of its internal data sequences and its external data manipulations.

Locating the Problems

Data domain problems are manifested as improper data sequences. It is important to note that the problem effect is always functional (i.e., data errors are transmitted) whether the cause is functional or electrical. This is true even for noise or voltage margin testing. Consequently, the first analysis step is locating the malfunction in the data flow sequence.

Locating a problem in data flow with an external instrument requires data registration or synchronization between the two systems, followed by data capture, possible manipulation, and presentation to the user. To meet this need, the Colorado Springs Division offered the Model 1601L, the world's first commercially available parallel-logic-state analyzer, in mid-1973. It did a relatively modest job—it was 12 channels wide, and had 16 words of memory and a simple Boolean breakpoint trigger.[1] It was developed to serve ASM designers. The 1601L consequently emphasized the development of such things as Boolean triggering for unique data occurrence registration and the portrayal of data as data—in machine code format with no time-relevence or electrical parameter display whatsoever.

The excitement at HP was not in the first logic state analyzer, however. It was in the concept and its application to the "computer-on-a-chip" revolution that was being born in 1973. Our view even then was that this was a singular opportunity for a fundamental change in test and measurement requirements—and that we could help to create that change by providing a reasonably comprehensive line of instrumentation for the new class of problems.

Today's Equipment

After several years of experience now, some obvious subtrends have emerged within this instrumentation area. We may differentiate between three major groups of equipment: development aids, bus analyzers, and nodal testers. We have already mentioned nodal testers. Logic state and timing analyzers are bus analyzers. Development aids are tools aimed at improving the time-efficiency of digital design, primarily in software and IC development.

The more sophisticated development aids have been primarily minicomputer-based to provide a higher-level language (e.g., PL-1, FORTRAN, BASIC, ALGOL) for code generation, and a more complete operating system for ease of text-editing, linking, and relocation of complex subroutines.

The difficulty with microprocessor-based systems has been the lack of completeness in the operating system software, relative lack

© Copr. 1949-1998 Hewlett-Packard Co.

of high-level language support, and lack of versatility for several microprocessors. Manufacturers of most of these systems today are promising improvements in the software support, including compiler-level language capability.

The disadvantage of mini-based systems has been lack of coordination between minicomputer vendors, users, and instrument manufacturers to provide a relatively simple system configuration. Emulation is not as easy to obtain as with a dedicated microprocessor-based system, nor is it as much of a concern when writing in a higher level language (which lacks direct analysis correlation in any event) for a complete chip set on a single-board computer. One significant advantage of this type of system is the ability to share the same software between multiple terminals so that several designers can simultaneously have access to the same major routines. There are today software companies offering cross-assemblers for several microprocessors available for most popular minicomputers. This, in conjunction with ROM simulators (word generators) and logic analyzers offering assembly-language tracing (e.g., HP 1611A) offers a very complete solution for the more extensive design laboratories using microprocessors.

System Bus Analyzers

The nodal testers are of chief value when it is known that a particular product or module is malfunctioning. Bus analyzers are of value for the more difficult task of ascertaining which portion of a system is malfunctioning. For example, a full system may have a central single-board-computer, two disc memory units, two data-entry terminals, and twenty on-line process-control transducers. We may fully expect the four peripheral units to incorporate one or more microprocessors themselves, resulting in a multi-computer network of sorts. Unravelling the network transactions is the first priority for troubleshooting.

We may characterize each major bus area as CPU, I/O and peripheral. The relevant parameters of the CPU bus structure are synchronous with clock speeds usually 3 MHz or less (for virtually all minicomputer and microprocessor systems). Bus contents may be address, data, instructions, and control signals. The I/O bus, in contrast, usually carries asynchronous, multiplexed data at speeds up to 10 MHz. Moreover, it sometimes runs much longer physical distances in a facility. Consequently, we may expect to find that race conditions, noise spikes, and glitches from various causes are much more significant problems on an I/O bus than on a CPU bus. Moreover if they exist, they will be much harder to trace to a source because of the multiplexed architecture and data flow at a particular point.

The CPU and I/O buses generally are parallel buses, which means, among other things, that the data format may be the same on both. Peripheral buses, in contrast, usually employ serial data transmission, which requires further data formatting. They are lengthy buses, but with very slow data rates, so glitches are not so significant except in batch transmissions (e.g., across the continent).

The point to be recognized here is that designing, installing, and servicing a general computer network is not a trivial task for which a simple tester can be described. At the same time, some very usable general-purpose testers that will work in a variety of these application areas can be defined.

The current second-generation set of logic analyzers described in this issue are the most effective bus analyzers available. They are configured for most bus analysis problems more or less on a bus-by-bus basis, with the very significant inclusions of cross-bus event correlation and very powerful selective data trace for linked and nested loop algorithms.

Matching Product and Need

Let us now stand back from product considerations and attempt to place perspective on the existing and future measurement needs. We view the problems of digital systems test at four levels: product design, product support, system design, and system support. The drawing at left depicts the typical tasks found in a product cycle (e.g., from lab design to end-user maintenance). Each has its own set of problems, but all are closely related.

It is interesting to consider what an end-user "lab environment" might be for a bank or airline company using a large computer system. For this grid, it seems natural to consider the EDP user-group applications software as flowing through a lab design into production and use. It is harder to conceive of a fast-food outlet doing any software or digital hardware design for their POS terminals, but easy to consider their maintenance requirement.

The grid illustrates the most effective area for each of the data-domain analysis equipments I have described. There are, of course many overlaps and shadings where different equipment has value that is not apparent in this diagram. Nonetheless, it should serve as a useful guide for assessing the major area of contribution of each type of instrumentation.

The original data-domain thesis is, of course, but a beginning. It is important that we recognize that the breadth of data domain analysis embraces not only local transactions of word-events, but the entire global picture of data state-space usage. This, consequently, includes the statistical pattern of event-flows (e.g., histograms, repetitive CRC sums, or perhaps even correlation for ergodicity), and it suggests that the mathematics of data-domain analysis already exists, but is not recognized conjunctively yet with these measurement techniques.

As we move into the 1980's, data-domain analysis equipment will become all-pervasive in our industry. Very large-scale integrated circuitry (VLSI) and its extensions not only offer the opportunity for lower-cost, higher-performance products for our society, but also the opportunity to change our design and test philosophies. What data-domain analysis tools make possible is top-down design for digital designers—which should permit us to consider not only block-structured programming languages, but block-structured IC layout and block-structured diagnostic and test routines. Without this change, we could imagine the fast-paced semiconductor revolution producing a chip capability of one million junctions or more for perhaps ten dollars by 1982—and then finding that IC designers or software designers need two or more years to use the $10 device.

Data-domain analysis techniques will help the revolution continue.

References

1. W.A. Farnbach, "The Logic State Analyzer—Displaying Complex Digital Processes in Understandable Form," Hewlett-Packard Journal, January 1974.
2. C.T. Small and J.S. Morrill, Jr., "The Logic State Analyzer, a Viewing Port for the Data Domain," Hewlett-Packard Journal, August 1975.
3. J.H. Smith, "A Logic State Analyzer for Microprocessor Systems," Hewlett-Packard Journal, January 1977.
4. G.B. Gordon, "IC Logic Checkout Simplified," Hewlett-Packard Journal, June 1969.
5. R. Adler and J.R. Hotland, "Logic Pulser and Probe: A New Digital Troubleshooting Team," Hewlett-Packard Journal, September 1972.
6. J.F. Beckwith, "Current Tracer: A New Way to Find Low-Impedance Logic Circuit Faults," Hewlett-Packard Journal, December 1976.
7. D. Priebe, "Multifamily Logic Clip Shows All Pin States Simultaneously," Hewlett-Packard Journal, December 1976.
8. R.A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," Hewlett-Packard Journal, May 1977.

Chuck House is manager of the logic analyzer department at HP's Colorado Springs Division. He has a BS degree in solid-state physics from the California Institute of Technology, an MSEE degree from Stanford University and an MA in History of Science from the University of Colorado. Upon getting his BS degree in 1962 he joined HP as a development engineer and was involved in many oscilloscope and CRT display projects before delving into logic analyzers.

© Copr. 1949-1998 Hewlett-Packard Co.

Interactive Logic State and Timing Analyses for Tracking Down Problems in Digital Systems

A new instrument combines 16-bit logic state analysis with 8-bit logic timing analysis to speed the location of problems involving asynchronous as well as synchronous events.

by John A. Scharrer, Robert G. Wickliff, Jr., and William D. Martin

BECAUSE THE SYMPTOM of a digital system failure is a deviation from the system's program sequence, logic state analyzers were designed to locate functional problems and do so even when the system is functionally complex. Although locating a functional problem is usually all that is needed to solve the problem, there are instances when more information is needed than a logic state analyzer can provide. For example, an occasional glitch on a clock, reset, or interrupt line may give rise to a functional problem, and timing errors, such as a foreshortened or missing read pulse, can alter a normal state sequence. Logic *timing* analyzers can help locate the sources of these kinds of problems.

Logic timing analyzers detect events that occur asynchronously with respect to the clock of the system being investigated. These events may occur as handshake sequences across an I/O port, where the order in which lines toggle is of interest, and they often arise as a result of signal delays, cross-coupling between conductors, reflections from impedance mismatches and the like. Logic state analyzers, on the other hand, are synchronous in nature, capturing the states of the monitored lines at the moment a clock edge occurs.

A logic timing analyzer might be described as a multichannel digital-storage oscilloscope with two-level resolution. It looks at up to eight input lines simultaneously and if any of them are above a threshold voltage level when sampled, that sample is stored as a 1 in the memory while samples from the lines below threshold are stored as 0's. The display circuits read out the memory repetitively and recreate the input signals as two-level waveforms synchronized with the sampling pulses (Fig. 1). The timing relationships between logic level changes are thus easily evaluated.

Usually, a sampling rate four or five times faster than the system clock is sufficient to capture asynchronous events of significance. For a glitch or event narrower than the sampling period, most analyzers latch the logic level until the next sampling pulse so the glitch can be detected. (A glitch could be described as a spurious narrow pulse or double transition that crosses the logic threshold level.)

At first glance it would seem that a real-time oscilloscope would provide more precise timing information, which is true, but there are no fast, real-time oscilloscopes that can display eight channels on a single sweep nor are there any that can display the events that occur *before* a trigger. Like logic state analyzers, logic timing analyzers can continuously replace old data in their memories with new data but stop and retain the data when the trigger event occurs. The logic timing analyzer thus provides information that the real-time oscilloscope cannot.

Combining State and Timing Analyses

All too often, digital system malfunctions occur in an apparently random and erratic manner. When

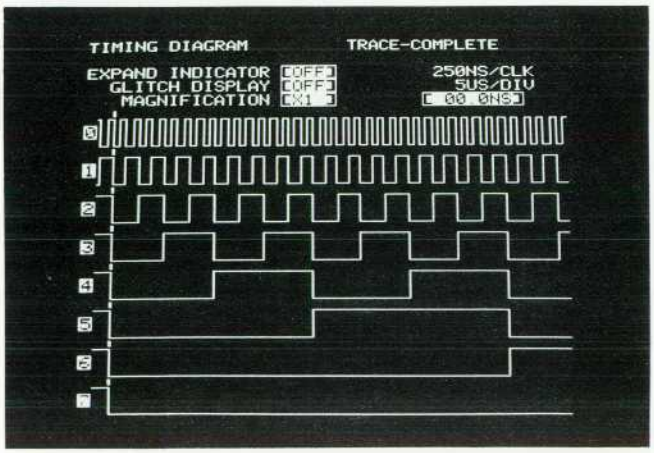


Fig. 1. *Typical logic timing analyzer display shows the order in which the eight monitored lines toggle. The trigger point is indicated by the short vertical bars at the left of the display. Raster-scan techniques are used to generate the display so alphanumerics as well as waveforms can be displayed.*

© Copr. 1949-1998 Hewlett-Packard Co.

Fig. 2. *Model 1615A Logic Analyzer functions as both a logic state analyzer and a logic timing analyzer. The state analyzer part of the instrument can trigger the timing analyzer so timing problems related to a particular part of a program can be located. Conversely, the timing analyzer can trigger the state analyzer so state information related to a timing event can be observed.*

using a logic analyzer to look at a "time snapshot" in a system going through a long sequence of functional states, it is important that the snapshot be taken when the system is executing the states or problem of concern and that the snapshot is located in the proper point in the state flow. If these conditions are not met, it is quite easy for the user to wander down the wrong path simply because he has observed and is trying to fix a glitch that is relatively unimportant to the problem at hand.

It thus becomes desirable to monitor both the synchronous and asynchronous behavior of a malfunctioning system. To make this possible, logic state analysis and logic timing analysis have been combined in a new instrument in such a way that both areas can be investigated simultaneously (or independently) in a single-shot environment. This combination of capabilities greatly shortens the time needed to track down a malfunction.

The interaction between state and time made possible by this instrument, Model 1615A (Fig. 2), can best be described by an example. The example concerns a microprocessor system under development that had a small keypad for data entry and control. It had been observed that the system detected a key-down condition and serviced the key when in fact no key had been pressed. To track down this problem, Model 1615A, set up in the timing mode, was connected to the microprocessor's interrupt line and set to trigger on a high on this line. Model 1615A then displayed interrupt pulses of normal duration on this line when none should have occurred. However, no glitches were detected.

Additional logic timing analyzer inputs were then connected to the inputs to the interrupt-request generator, a monostable multivibrator. Model 1615A showed that a glitch occurred on one of these inputs.

The next step was to find out what was happening in the system when the glitch occurred. The state probes of Model 1615A were connected to the microprocessor's address bus and the analyzer was switched to dual state and timing operation. The trigger was changed to start the timing trace on the glitch on the interrupt-request generator's input line, and the timing analyzer part of the instrument was used to end data capture by the state analyzer part. The microprocessor states leading up to the glitch were thus captured for display. Repeated tracings made in this mode revealed that the microprocessor was always executing the same instructions when the glitch occurred.

The listing of the microprocessor code showed that the instruction being executed when the glitch occurred was "read I/O port 200." The I/O read line and the chip-select input to port 200 were then connected to the timing machine. The resulting timing display revealed the source of the problem: the falling edge of the I/O read pulse coincided with the glitch.

Examination of the circuit board revealed that these two signals ran side by side for some distance. Rerouting one of them to eliminate the capacitive coupling between lines solved the problem.

© Copr. 1949-1998 Hewlett-Packard Co.

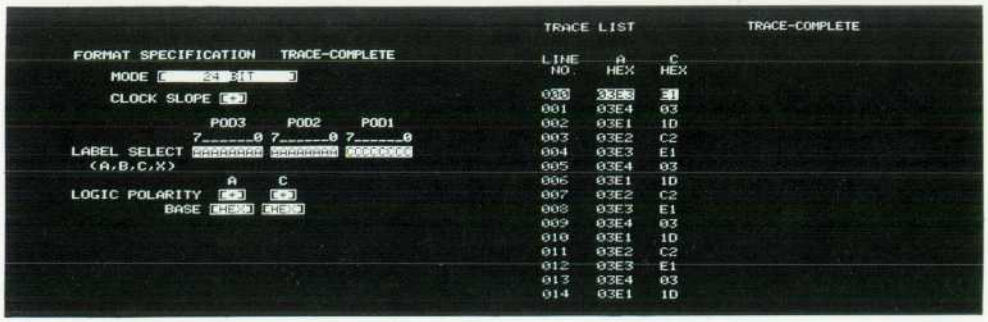


Fig. 3 *When configured as a 24-bit logic state analyzer, Model 1615A allows the user to label the input probes and select the numerical base for the captured data, using the* FORMAT SPECIFICATION *"menu" (left), for easier interpretation of the captured data when displayed on the* TRACE LIST *(right).*

The Instrument

Model 1615A Logic Analyzer has three primary modes of operation. The first is as a 24-bit, 20-MHz state analyzer (Fig. 3). Memory is 256 words deep with 15 words on display at one time. The words can be displayed in binary or formatted into octal, decimal, or hexadecimal code. The input clock can be qualified with up to six qualifiers and the qualifier expression can contain two ORed terms so that processors with independent read/write qualifiers can be monitored. Other capabilities include selective trace and tracing after the nth occurrence of a state.

The second mode is as an 8-channel, internally clocked, timing analyzer (Fig. 4). The captured data is displayed in the timing diagram format and the X axis can be expanded by a factor of 10 for greater display resolution. Other characteristics include internal clock rates up to 20 MHz, detection of glitches as short as 5 ns, asynchronous pattern triggers, trigger delay in terms of time or external clock periods, OR and NOT triggering, and direct readout of time intervals. It can also be triggered on glitches.

The third and newest mode involves time-state interaction. In this mode, 16 bits can be traced for state analysis while at the same time 8 bits are traced for timing analysis. The state analyzer trigger word can trigger or arm the timing analyzer, enabling the timing window to be located more precisely near states of interest (Fig. 5). Conversely, timing (including glitch trigger) can trigger or arm the state machine, thereby locating states related to timing phenomena, as described in the example problem above.

Glitch Detection

Detectors that monitor the eight timing input lines capture and store glitches independently of the sampled data. Several advantages accrue from this arrangement. The first is that the glitches can be intensified on the timing display, making their presence obvious (see Fig. 4). This is particularly important when narrow events occur in a relatively long time span. Even though system operation may be quite slow, narrow glitches can affect the operation of the logic families commonly used in slow systems, so the presence of a glitch nanoseconds wide should be made known even when the displayed timing diagram spans milliseconds of data.

The second advantage is that a glitch occurring in the same sample period as a data change is detected separately and displayed as an intensified edge on the data. A double transition or "hook" on a data transition at a clock edge is an example. Separate detection of glitches is important because glitches are likely to occur at clock edges, where there are cross-coupling or grounding problems, because this is when logic lines are changing. Clock-edge reflections from impedance mismatches on heavily loaded or widely distributed clock lines are another source of glitches. Most glitch detectors have missed these glitches because the detectors stretch them for the sampler.

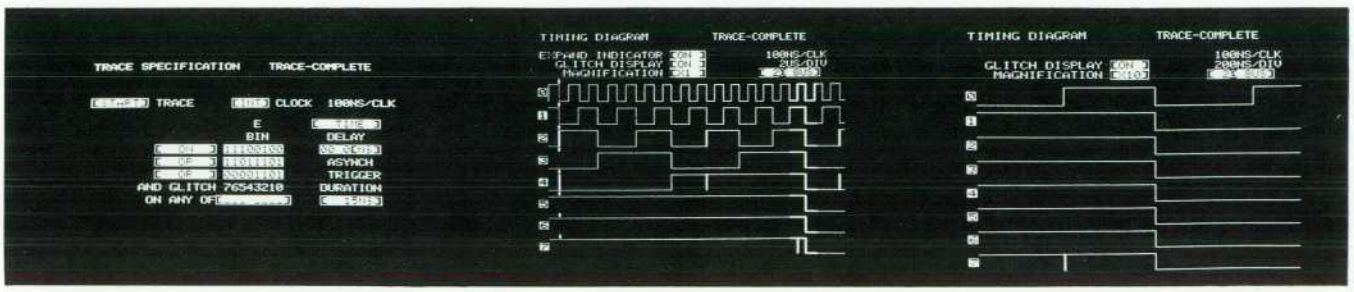


Fig. 4. *Configured as an 8-bit logic timing analyzer, Model 1615A allows the user to select up to three ORed trigger patterns with the* TRACE SPECIFICATION *menu (left). Triggering on glitches can be ANDed with the trigger patterns. Detected glitches are displayed as higher-intensity vertical bars on the* TIMING DIAGRAM *display (center). The brightened segment of the display can be expanded horizontally by a factor of 10 (right) for increased display resolution.*

© Copr. 1949-1998 Hewlett-Packard Co.

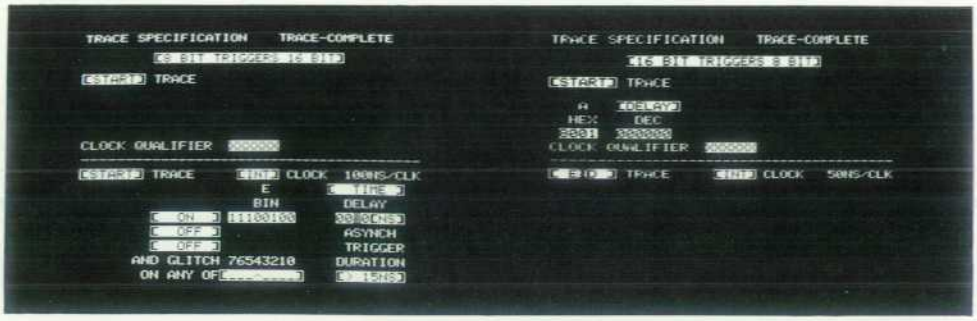


Fig. 5. *In the TRACE SPECIFICATION at left, the trigger point is specified (below the dashed line) so the timing analyzer will start data capture by the state analyzer (above the dashed line). In the TRACE SPECIFICATION at right, occurrence of the trigger specified for the state analyzer ends data capture by the timing analyzer, capturing the pre-trigger timing data.*

Thus, if there is a glitch and a data change in the same sample period, only the data is displayed. Model 1615A, on the other hand, captures both individually and displays the glitch as a brightened edge on the data.

A third advantage of this type of glitch detection is that glitches can serve as triggers in conjunction with the eight timing lines. A combination of states for the timing lines can be ANDed with the specification that a glitch be required on any ORed combination of the same lines for a trace to start. This is a powerful tool for triggering at the source of a system fault.

Versatile Triggering

Triggering is further enhanced by Model 1615A's ability to trigger on the logical NOT-AND of the timing lines. In this way, a timing trace can be initiated when any one of several control or status lines changes state. Furthermore, up to three combinations of states or their NOTs can be ORed to create a three-term trigger specification. For example, if it is desired to trigger on the exclusive-OR of two lines, Model 1615A can be set to trigger on XXXXXX01 or XXXXXX10, and triggering will occur when either of these states occurs.

The actual trigger point is indicated on the timing diagram display by short bars on a vertical line (Fig. 4). To avoid triggering on transient states that are not of importance in determining a valid trigger condition, the length of time that a trigger pattern has to exist to be considered valid is selectable in a range from 15 ns to 2 μs.

Model 1615A can also be edge-triggered on an extra input line provided on the clock probe. Either a + or − edge can be selected. Actual triggering can also be delayed with respect to the trigger state either in units of absolute time or qualified clock pulses.

As a logic timing analyzer, Model 1615A has an internal clock operating with periods selectable from 50 ns to 500 ms per sample, and it can also work with an external clock. Operation is either single shot or repetitive with pre-trigger or post-trigger data collection. The trigger can be delayed in units of an external clock even though the internal clock is used for sampling. The timing display shows 249 of the 256 samples taken, with vertical graticule lines and a time- or samples-per-division readout added to facilitate reading and relating data. The timing data can also be displayed in tabular form. The glitch display can be turned on or off as desired although glitches are always captured and made available for display.

Interfacing to Model 1615A

Data and clock inputs to Model 1615A are through four pods similar to those used in other Hewlett-Packard logic analyzers. Each pod has eight miniature probe inputs and a ground. Pod 1 is for the timing inputs, pods 2 and 3 for state inputs, and pod 4 for an external clock, six clock qualifiers, and a trigger input. Logic threshold levels are selectable between −10V and +10V.

The man-machine interface is keyboard-and-menu-display, similar to Model 1610A Logic State Analyzer described in the preceding article. The CRT display is a raster-scan type for both the timing diagram and alphanumerics. Menus are called up by the keyboard and changes are made by moving a cursor to the desired location and pressing the appropriate data entry keys. The input lines can be labelled on the display to make it easier to format and evaluate data listings (Fig. 3).

Technical Details

The use of a microprocessor to perform the operations of Model 1615A greatly simplified the amount of hardware needed to perform the many functions. However, for high-speed data acquisition, the microprocessor is simply a controller and the actual data collection is performed by dedicated hardware.

Each of the data inputs is to a fast ECL-III comparator with a 50-kΩ input impedance and a differential output. The outputs are applied through a short delay line to a latch within the instrument and the outputs of the latch go to the data memory and the trigger circuits.

The clock probe is electrically identical to the data probes except there is no delay line between the comparator and the latch for the clock. This insures that the data channel hold time is zero to make certain that the data captured is exactly that "seen" by the system

© Copr. 1949-1998 Hewlett-Packard Co.

under test. The clock latches the data and activates a timing generator that writes the data into the memories, advances the memory address counter, and latches the trigger information.

The output of the data latch is also used as an address for the trigger memory. If a trigger condition has been stored at that address, the memory output indicates a valid trigger condition. The trigger memory is sampled about 50 ns after the corresponding clock pulse is received, provided the six clock qualifiers meet required conditions. If the END TRACE mode was selected, the trigger word and the 255 previous qualified words are retained in memory in response to a valid trigger. If the START TRACE mode was selected, an 8-bit counter is started which allows an additional 255 qualified words to enter prior to inhibiting any further data memory loading.

When TRIGGER DELAY is in use, a valid trigger enables a 20-bit counter and the trace point is delayed until the selected number (up to 999,999) qualified words have entered.

Model 1615A may be set up to capture trigger events only instead of the trigger event and a series of qualified words. This is done by allowing the memory address and index counters to advance by one each time the trigger condition (trigger plus delay) has been met. This mode is useful, for example, for restricting data capture to only data that is written to a particular I/O port.

The instrument may also be set up to trigger only after the trigger word has occurred a specified number of times, a useful mode for capturing data on the exit of a loop. This is done by allowing the 20-bit delay counter to advance only when the trigger word is received.

Logic Timing and Glitches

The 8-bit timing analyzer section functions similarly to the state analyzer section except that the clock is internally generated (at rates between 2 Hz and 20 MHz). Also, a filter on the output of the trigger memory requires the trigger state to be maintained a definite length of time (selectable between 15 ns and 2 μs) for a valid sample to be obtained.

The horizontal time base can be expanded by a factor of 10 for increased display resolution (Fig. 4). The section to be expanded is shown by the expand indicator, the brightened portion of the normal display, and is moved right or left by the ROLL keys. The expand indicator can also be used to measure the time interval between any two points on the display by zeroing the relative time indicator with the FIELD SELECT key while the leading edge of the expand indicator is positioned on the first point, then moving the leading edge to the second point. The time interval is then displayed in the relative time indicator field on the display.

The 8-bit timing section also has the glitch detectors. These look at the outputs of the timing data-input pod comparators in parallel with the data latches and thus separate the glitches from the sampled data, storing them in a separate memory.

To overcome some of the problems associated with earlier glitch detectors,[1] the 1615A detectors were designed to detect glitches defined by two conditions:
(1) If a logic high is detected by the sampler clock and a positive transition occurs during the following sampler clock period;
(2) If a logic low is detected by the sampler clock and a negative transition occurs during the subsequent sampler clock period.

To see the advantages of this scheme, consider the typical glitches shown in Fig. 6. The glitch at A would be missed without any glitch detection scheme. With a pulse stretcher, the glitch is lengthened sufficiently to allow its detection by the next sample pulse.

With the new detection scheme, the sample clock latches a low level at time T_0 and a negative transition is detected at time T_{g2}, indicating the presence of a glitch. The glitch is then displayed as a bright, vertical bar rather than as a data unit.

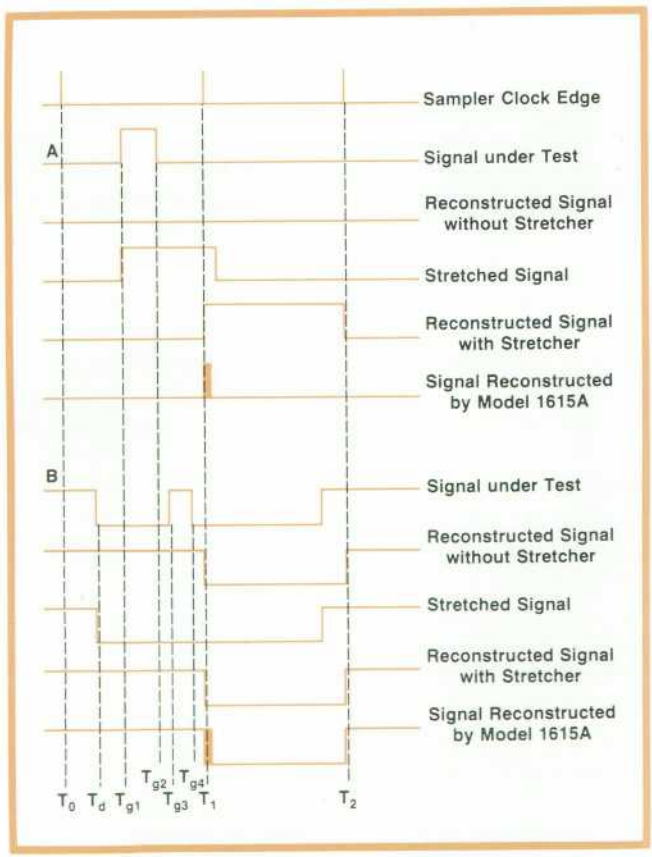


Fig. 6. *Representative glitches (A and B) and how logic timing analyzers react to them.*

© Copr. 1949-1998 Hewlett-Packard Co.

As for the glitch at B, a pulse stretcher would react to the transition at time T_d and stretch it until the next sample, ignoring the glitch at time T_{g3}. With the new scheme, the sample clock latches a high level at time T_0. A positive transition is then detected at time T_{g3}. For display, the bright bar is superimposed on the data transition at time T_1.

Circuit Details

A simplified block diagram of the glitch detector for one channel in Model 1615A is shown in Fig. 7. To prevent dead time when the glitch detector is being reset, there are two detectors per channel. This allows one to be operational while the other is being reset.

Operation is as follows. Assume that U1-Q is driven low and U1-$\overline{Q}$ high by a sample clock, enabling the A detector and inhibiting the B detector. If the sample clock had also detected a logic high on the input signal, the sampler output would be high and U4-A would also go high when the reset pulse occurs. This forces U6-A$\overline{Q}$ low. If a positive transition then occurs before the next sample clock, U5-A$\overline{Q}$ will go low.

The next sample pulse sets U1-$\overline{Q}$ low so all inputs to U7 will be low, setting U9 high. The glitch reset pulse then loads the output of U9 into the glitch memory for later display. The output of U9 can also be used as a trigger.

The same sample pulse also sets U1-Q high, disabling the A detector while U1-$\overline{Q}$ enables the B detector by going low.

Operation for negative-going glitches is similar except that U3-A becomes active in place of U4-A, and the roles of U5-A and U6-A are interchanged.

Acknowledgments

Mechanical design was by Harry Short who also

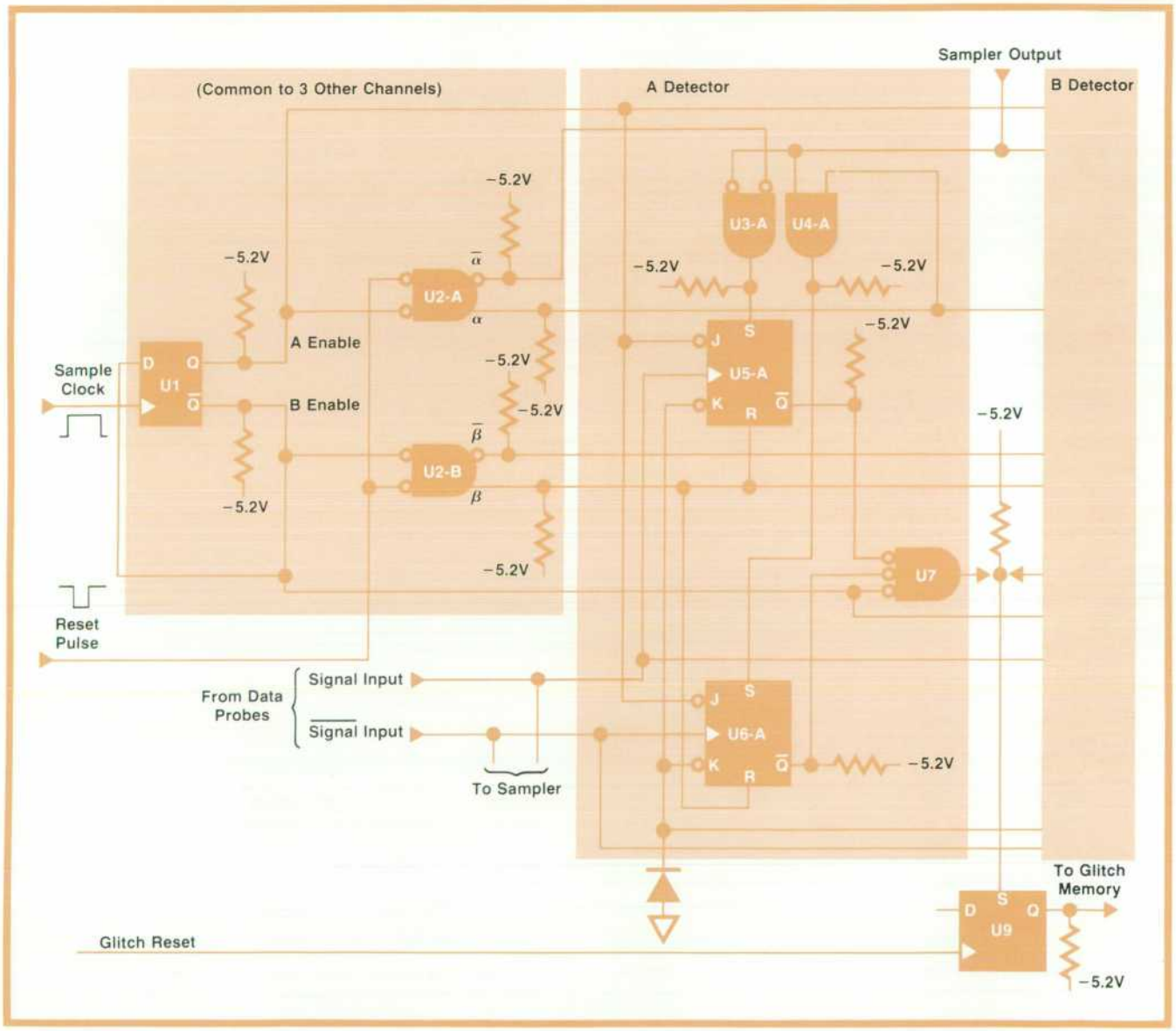


Fig. 7. *Simplified block diagram of Model 1615A's glitch detection circuits.*

© Copr. 1949-1998 Hewlett-Packard Co.

worked with Larry Koperski in smoothing the instrument's way into production. Jim Williams provided industrial design expertise. Dave Hood wrote a good part of the software and took on the responsibility for all hardware in the latter stages of the project, as well as providing a comprehensive self-test system. Product marketing managers Don Corson and Bruce Farly contributed to the definition of the instrument. Thanks are due Bill Farnbach, in whose section the product was defined, and to Chuck Gustafson, in whose section the product was carried to completion.

William D. Martin

Bill Martin joined the HP Colorado Springs Division in 1972 upon getting his MSEE degree from the University of Florida (he obtained his BSEE degree there a year earlier). Initially he did circuit design for a storage scope and later worked on the vertical-channel preamplifier of the Model 1710B Oscilloscope. He then returned to Florida, doing digital design for remote-controlled aircraft, but returned to HP a year later where he worked on the data-acquisition section of Model 1615A. In off-hours, Bill likes to make furniture and he also likes to take the family (wife and two boys, 4 and 1) camping.

Robert G. Wickliff, Jr.

Bob Wickliff was born and raised in Columbus, Ohio, and obtained BSEE (1969) and MSEE (1970) degrees from nearby Ohio State University. Following graduation, he worked on electronic countermeasures as a graduate research associate in OSU's ElectroScience Lab, coauthoring several papers in the process. Four years later he joined HP, going to work in the CRT lab, where he contributed the design of the Model 1741A Oscilloscope's variable-persistence/storage tube, then in the logic timing analyzer group where he did the firmware for Model 1615A. Bob and his wife enjoy extended travel in their VW bus but lately they've been stay-at-homes, doing plumbing, wiring, and other things in the shell home they bought.

John A. Scharrer

A native of Sheboygan, Wisconsin, John Scharrer obtained a BSEE in 1963 and an MSEE degree in 1965 from the University of Wisconsin, then joined Hewlett-Packard. At HP, he was involved for several years in vertical amplifier design for scopes and CRT displays and was project leader on some of the 1200-series scopes and displays, the 1707A oscilloscope, and several 180-series oscilloscope plug-ins. He was project leader for the Model 1615A Logic Analyzer and is now group leader for logic timing analyzers. In his spare time, John enjoys biking, tennis, and bridge-playing with his wife, and camping with the whole family (all girls, 11, 8, and 4) in their camper.

References

1. R. Adler, M. Baker, and H.D. Marshall, "The Logic Analyzer: A New Kind of Instrument for Observing Logic Signals," Hewlett-Packard Journal, October 1973.

SPECIFICATIONS

HP Model 1615A Logic Analyzer

CLOCK QUALIFIER AND DATA INPUTS
- REPETITION RATE: to 20 MHz.
- INPUT RC: 50 kΩ shunted by ≤14 pF at probe tip.
- INPUT BIAS CURRENT: ≤20 μA.
- INPUT THRESHOLD: TTL, fixed at approximately +1.4 V; variable ±10 Vdc.
- MAXIMUM INPUT: −15 V to +15 V.
- MINIMUM INPUT
  - SWING: 0.6 V.
  - CLOCK PULSE WIDTH: 20 ns at threshold level.
  - SETUP TIME: 20 ns.
  - HOLD TIME: zero.

SYNCHRONOUS OPERATION
- TRIGGER DELAY: to 999,999 clocks.
- TRIGGER OCCURRENCE: to 999,999.

ASYNCHRONOUS OPERATION
- SAMPLE RATE: 2 Hz to 20 MHz.
- DATA SKEW: 9 ns maximum.
- MINIMUM DETECTABLE GLITCH: 5 ns with 30% peak overdrive or 250 mV, whichever is greater.
- GLITCH TRIGGER: on any selected channel(s), if glitch is captured glitch is AND ed with asynchronous pattern trigger.
- EXTERNAL TRIGGER PULSE WIDTH: 5 ns minimum with 30% peak overdrive or 250 mV, whichever is greater.
- PATTERN TRIGGER: any 8-bit pattern. Trigger duration required is selectable 15, 50, 100, 200, 500, 1000, or 2000 ns ±15 ns or 15%, whichever is greater.
- DELAY TIME: to 1, 048, 575 × sample period.

TRIGGER OUTPUTS (rear panel)
- 16/24 BIT TRIGGER OUTPUT
  - LEVEL: high, ≥2 V into 50 Ω; low, ≤0.4 V into 50Ω.
  - PULSE DURATION: approximately 25 ns.
  - DELAY FROM INPUT CLOCK: approximately 85 ns.
- 16/24 BIT TRACE POINT OUTPUT
  - LEVEL: high, ≥2 V into 50Ω; low, ≤0.4 V into 50Ω.
  - PULSE DURATION: starts at beginning of trace and ends at trigger point (pattern trigger plus delay).
- DELAY FROM INPUT CLOCK: approximately 85 ns.
- 8 BIT PATTERN OUTPUT
  - LEVEL: high, ≥2 V into 50Ω; low ≤0.4 V into 50Ω.
  - PULSE DURATION: pattern duration minus asynchronous trigger duration width.
  - DELAY FROM PATTERN RECOGNITION TRIGGER AT PROBE: approximately 75 ns plus asynchronous trigger duration width.

GENERAL

MEMORY DEPTH: 256 data transactions (in timing display mode, 249 samples are displayed).

POWER: 100, 120, 220, 240 Vac; −10% to +5%; 48 to 66 Hz; 230 VA max.

DIMENSIONS: 189 mm H × 426 mm W × 664 mm D (7.438 × 16.75 × 26.125 in.).

WEIGHT: 19.1 kg (42 lb).

ACCESSORIES SUPPLIED: three 8-bit Model 10248B data probes and one Model 10248B opt 001 clock probe with probe leads and tips (three for data and one for clock, qualifiers, and external trigger).

PRICE IN U.S.A.: $6800.

MANUFACTURING DIVISION: COLORADO SPRINGS DIVISION
P.O. Box 2197
Colorado Springs, Colorado 80901 U.S.A.

© Copr. 1949-1998 Hewlett-Packard Co.

Entry Level Logic State Analyzer Has High-Level Capability

Operable by a first-time user without any prior instruction, this compact, portable logic-state analyzer is also capable of sophisticated analyses of data flow. Moreover it's programmable, making possible low-cost, automatic systems for functional testing.

by Charles T. Small and Alan J. DeVilbiss

WHAT STARTED OUT during the product definition stage to be an easy-to-understand, low-cost logic state analyzer has evolved into a highly versatile though economical instrument, Model 1602A (Fig. 1). A multi-faceted tool that is equally useful in the design lab, in the field, and on the production line, this new instrument still retains a high degree of "friendliness" for the uninitiated user.

Like other logic state analyzers, the new Model 1602A has a multi-input probe that can be connected to an address bus, a data bus, or other group of digital signal lines, and it can be set to capture and store a series of states or words appearing on these lines in response to the occurrence of a selected trigger word. The user can thus obtain a record of a sequence of logic states in a form suitable for analysis.

Ease of operation was a primary design goal for Model 1602A. Thus, the keyboard (Fig. 2) was designed to be as self-explanatory as possible. For example, the FORMAT group of keys at the left enables the first-time user to select the logic polarity, the edge on which the clock triggers, and the format in which the data is to be displayed without referring to an instruction manual. However, operating procedures are outlined on an instruction card mounted in the flip-up lid (Fig. 3). The card gives all the information necessary for the first-time user to apply the machine to problems in digital program execution. The card also lists a number of messages from the machine to the operator, such as "no clock," that are identified by code numbers displayed by the machine at appropriate times.

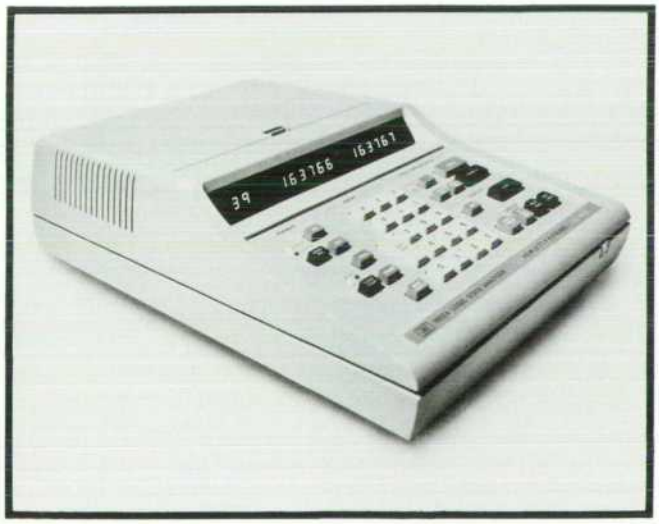


Fig. 1. *Model 1602A Logic State Analyzer captures and stores for analysis sequences of 64 digital words up to 16 bits wide occurring at rates up to 10 MHz. Although designed for ease of use, it is capable of seeking and capturing the desired information in complicated programs.*

Briefcase Size

In the interests of portability, Model 1602A is housed in a briefcase size package. To keep things small, the stored information is displayed on a row of 18 seven-segment LEDs, rather than on a CRT. When displaying stored data, the two LEDs at the far left of the display show the address of the analyzer's memory contents that are displayed in the center. If there is room in the display, which is the usual case except when the binary display format is used, the next word in memory is displayed on the right (Fig. 2). The 64-word memory can be rolled forward and backward through the display with the NEXT WORD and PRIOR WORD keys.

The captured data, stored in binary form, can be displayed in decimal, octal, hexadecimal, or binary format. When test parameters such as the trigger word are being entered, the LEDs display the entered data for verification in the same format selected for the data.

Versatile Probes

Another design feature related to ease of operation is the input probe. The single probe has inputs for 16 data lines, a clock input, a clock qualifier, and ground. The input end of the probe has a standard

© Copr. 1949-1998 Hewlett-Packard Co.

Untangling the Probing Problem

As logic state analyzers grow in versatility, the number of simultaneous signals they should monitor also grows. As a means of inputting all these signals, the conventional miniature grabber probes serve very well in a research environment but in a production test environment where the same measurement may be made on many products in succession, reconnecting 16 or more leads becomes a time-consuming chore that most companies can ill afford.

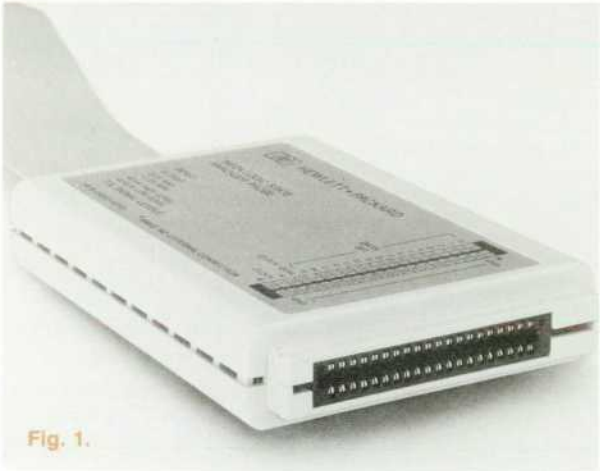


Fig. 1.

One solution to this problem is shown in Fig. 1. By terminating the Model 1602A Logic State Analyzer probe pod with a circuit-board edge connector, a multiplicity of convenient, low-cost methods of connecting up to 16 signals to the analyzer becomes available, as shown in Fig. 2.

The fastest and most direct way to make these connections is to include 20-pin edge connectors on the circuit boards in the system to be tested. The desired signals are routed to the appropriate pins on the connectors. The probe pod can then be attached quickly with the added advantage of eliminating any danger of misplaced signal leads.

For the development lab, the new probe pod is easily adapted to conventional miniature probe connections by the multilead

Fig. 2.

Fig. 3.

connector shown at the bottom right of Fig. 2. More likely, the design engineer may wish to make his own lead set for soldering into a circuit semipermanently or for connecting to a clothes-pin style IC clip. For these applications, the connector is available in a connector body without any leads. Individual leads or flat cable can then be attached to the connector as desired. The engineer can also make his own 20-pin pc connectors and wire them into circuits to be tested.

Working with the HP Interface Bus

To facilitate connections to the HP interface bus, a probe connector is available attached to a ribbon cable that is terminated in an HP-IB connector shown center right in Fig. 2 and also in Fig. 3. This can be plugged on to any of the piggy-back connectors in an HP-IB system. The logic state analyzer is then enabled to monitor activity on the bus, encoding the eight data lines for display in the octal number base while the eight control lines are displayed in binary.

Where the Model 1602A is to be used extensively for work with HP-IB systems, an input probe pod (Fig. 3) has been designed to work specifically with the bus. It serves as a pre-processor for the logic state analyzer by providing three sources of clock input as follows: (1) the DAV (data valid) line, which causes data to be loaded into the logic state analyzer whenever any device on the HP-IB system places data on the bus; (2) the NDAC (not-data-accepted) line, which clocks data into the analyzer whenever the intended recipient receives data; or (3) the parallel-poll state (ATN and EOI lines low), which enables the analyzer to monitor the response of HP-IB devices to the HP-IB controller's request for device status. The probe pod also has a pushbutton for manually entering the status of the HP interface bus at any time.

The clocking signal can also be qualified so only HP-IB commands are monitored or only HP-IB data. In addition, logic circuits in the probe pod monitor every HP-IB three-wire handshake sequence and flash a LED indicator whenever an abnormal sequence is detected. The abnormal sequence also causes a trigger pulse to be supplied to an oscilloscope probe socket on the probe body. Using this trigger pulse, an oscilloscope or logic timing analyzer can help track down the timing problems that may be involved in an abnormal handshake sequence.

printed-circuit-board edge connector that allows various arrangements of probing leads to be quickly attached and detached (see box above). During the design phase of a digital system, several lead sets can

© Copr. 1949-1998 Hewlett-Packard Co.

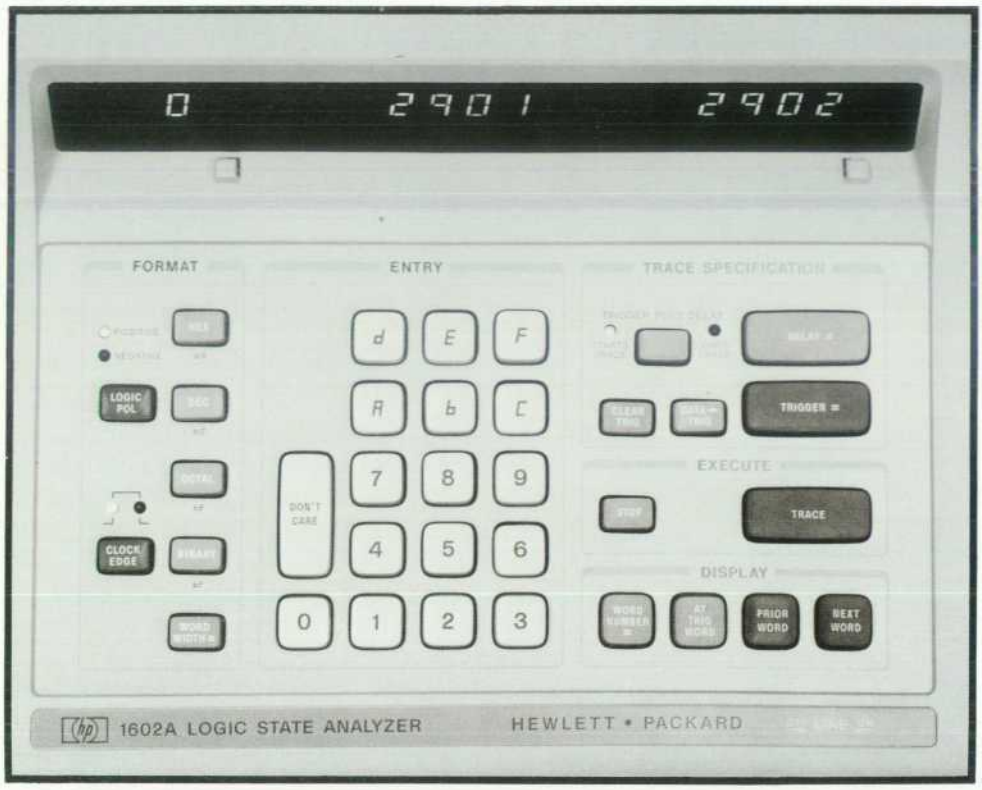


Fig. 2. *Model 1602A's keyboard is self-explanatory. The 18-digit LED display shows two consecutive words stored in memory, when operating in the octal, decimal, or hexadecimal mode, with the address of the center word shown at left. With binary words more than 6 bits wide, it displays one word and its address.*

be attached to the system with each set dedicated to a given logic state measurement. Thus, a quick and easy shift from one measurement to another is possible. This avoids individual wire hook-ups, which are tedious when 16 leads are involved and whose connection sequence may be suspect. In addition, if standard edge-connector pads for the probe are included on a PC board being designed, a quick and reliable probing interface becomes permanently available.

The threshhold level of the input lines is fixed at 1.5 volts, compatible with TTL circuits. The instrument accepts data at clock rates up to 10 MHz. Set-up and hold times are 35 and 0 ns respectively. The 16 data inputs can be connected to data buses, qualifiers, or any other lines that have information of interest. In the majority of applications, the user monitors the address bus of a system as this clearly delineates program flow.

With the probe connected to the appropriate points in the system, operation is straightforward. To select the segment of program flow to be monitored, the operator presses the TRIGGER= key then enters the appropriate trigger word using the part of the key pad appropriate to the numerical base chosen. Data capture can be delayed beyond the trigger word by entering the desired number of states of delay (up to 65,535) with the DELAY= and the decimal keys. The operator can elect to start data capture on the trigger (plus any specified delay), or stop data capture with the trigger word (plus delay) thereby preserving the data leading up to the trigger point. He can then examine the acquired sequence using the NEXT WORD or PRIOR WORD keys to cause the display to roll through the stored data, or the WORD NUMBER= key to find a particular word in memory. The AT TRIG WORD key returns the display to a known point.

Beyond the Elementary

The capabilities described so far are those of a basic logic state analyzer. As a matter of fact, the new Model 1602A has all the capabilities of the first logic state analyzer, HP Model 1601A,[1] with the added advantages of accepting up to 16 rather than 12 inputs and storing 64 rather than 16 states. However, in the hands of the knowledgable user Model 1602A is capable of far more sophisticated testing. For example, it can be used to monitor the termination of a program loop by being able to delay data capture until the nth pass through the loop. This is done by entering an E after pressing the DELAY= key. This causes the instrument to delay data capture until the trigger word has been encountered n more times, where n is the number entered following E. This mode is known as the delay-by-events mode.

Related to the delay-by-events mode is the trace-events mode. This mode is invoked by entering E immediately after the TRACE key is pressed. The instrument will then store only those words that meet the trigger specifications (i.e., trigger plus delay). This mode is used to trace all events that satisfy a particular condition, such as all accesses to a particular page of memory. In this case, part of the trigger

© Copr. 1949-1998 Hewlett-Packard Co.

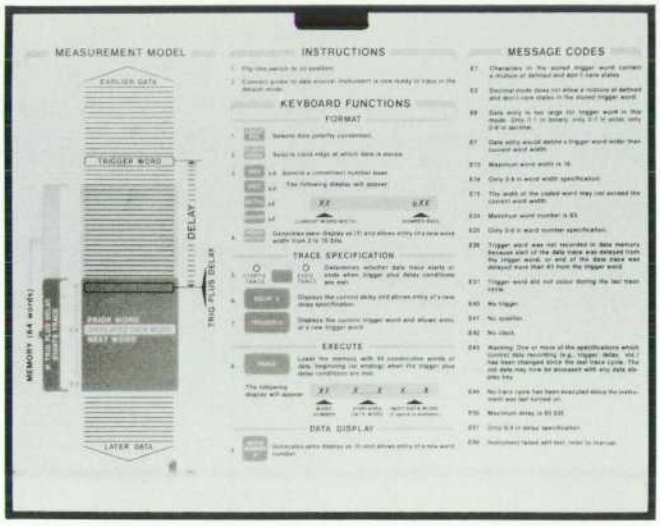


Fig. 3. *The instruction card inside the flip-up lid gives complete instructions for operating the instrument and also explains the operator messages that the analyzer presents in code.*

word is common to all accesses and this part is entered into the trigger specification. The remaining part of the trigger specification is filled with DON'T CARES and the data of interest is input on these lines.

The trace-events mode is particularly useful when used with a trigger-plus-delay specification. For example, this could be used to monitor changes in a program step in an iterative loop. The program step captured would be n steps (the selected delay) downstream from an unchanging word in the loop that is used as the trigger word.

Additional qualification of the trigger word can be obtained by use of the rear-panel TRIGGER QUALIFIER input. Anything that holds this input in the low state prevents the instrument from responding to the trigger word. This single qualifier can be expanded to four qualifiers by use of the Model 10250A Trigger Probe, which is essentially a four-input AND gate.

A similar arrangement is used to provide additional qualifiers for the CLOCK input. Anything connected to the rear-panel CLOCK QUALIFIER input that holds this input low prevents the instrument from loading data, recognizing trigger words, or counting delay cycles. This function can be used for sorting data. For example, if the ATN line of the HP interface bus is connected to this input, the 1602A will not recognize clock edges when commands are on the data lines (ATN is low), but when data is on these lines (ATN high), the 1602A will load the data into its trace memory when trigger conditions are met.

Mixed Display Mode

To aid in the interpretation of data acquired simultaneously from two different types of inputs, the instrument is able to encode part of each data word into the hexadecimal, decimal, or octal format and the remainder in binary (Fig. 4). This mixed display mode is obtained by first pressing the WORD WIDTH= key followed by the total number of bits in the word (if less than 16), then pressing the HEX, DEC, or OCTAL key followed by the number of bits to be encoded. These bits, which will be the lowest numbered bits, will be encoded for display while the remainder will be displayed in binary.

The mixed display mode is useful for simultaneous monitoring of a data bus and control lines. Information on the data bus can be encoded for easier interpretation while the states of the control lines are presented in binary as high or low (1 or 0). These lines can be identified on the Model 1602A by writing their names on labels that can be placed on tabs located in front of the display window. The tabs allow the labels to be changed quickly and conveniently when the measurement set-up is changed.

Automated Test Systems

With an optional plug-in HP-IB interface card installed, Model 1602A can serve as a "transducer" for a system controller in an automatic test system, feeding sequences of logic states it acquires to the system controller for analysis (Fig. 5). This enables very rapid functional testing of digital systems in a production or service environment.

When a test program is being prepared, Model 1602A can be set up manually for a particular test and then allowed to perform the test on a known, good unit. Following this, the system controller can be instructed to interrogate Model 1602A, which will then transfer all its control settings and the contents of its memory to the system controller, where the information becomes part of the test program. Subsequently, during execution of the test program, the controller will set up the 1602A exactly as before and when the 1602A acquires data according to this set-up, the controller will compare the new data to the previously stored data. The test program can be devised to indicate to the operator the steps to be taken appropriate to the result of the comparison. With the proper test procedures, the causes of functional problems can be isolated very quickly this way, not only on the production line but in field service as well.

An application program has been prepared for the

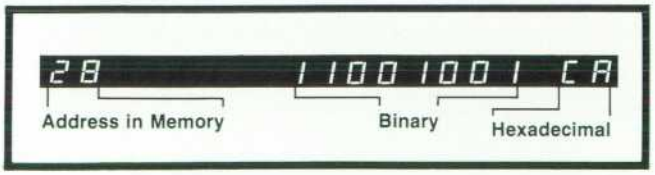


Fig. 4. *Mixed display mode encodes the lowest-numbered bits into the octal, decimal, or hexadecimal format (CA here) and displays the remaining bits in binary.*

© Copr. 1949-1998 Hewlett-Packard Co.

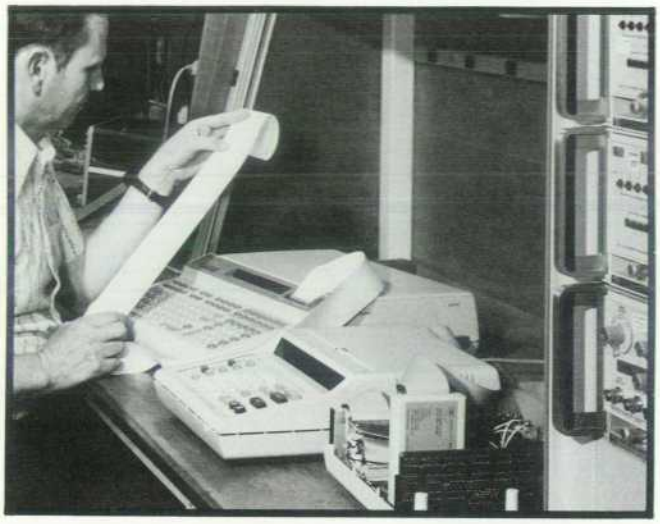


Fig. 5. *Model 1602A, shown here with a Model 9825A Desktop Computer, is programmable by way of the HP Interface Bus and is thus easily adapted to automatic test systems.*

Model 9825A Desktop Computer working with the Model 1602A Logic State Analyzer. Available on a tape cartridge, this is a conversational program that directs the user to set up the 1602A for the various test procedures he wants, and automatically inserts the control settings and data for comparisons into the program. A complete test program, stored on tape for subsequent use, is generated very quickly this way without requiring the user to know the formal HPL language of the Model 9825A Desktop Computer.

Production versions of the Model 1602A are tested by such a system. Use is made of the HP-IB port to make the inner workings of the 1602A readily accessible to the test system. A complete test is completed in less than an hour, including any repair and retesting time needed for units that don't work when first turned on. This compares with the more than four hours of production test time that was required for earlier logic state analyzers.

Instrument Organization

A simplified block diagram of a basic logic state analyzer is shown in Fig. 6. The fundamental element is the random-access memory (RAM). It stores a sequence of program steps or states appearing on the lines being monitored, usually the address bus of a digital system. With this configuration, loading the memory is inhibited until the pattern trigger circuit recognizes the occurrence of a particular bit pattern on the monitored lines and closes the clock-line switch, starting the counter.

The counter supplies the addresses for loading the memory and when it reaches terminal count, it stops. The sequence of states loaded into memory is thus retained for display.

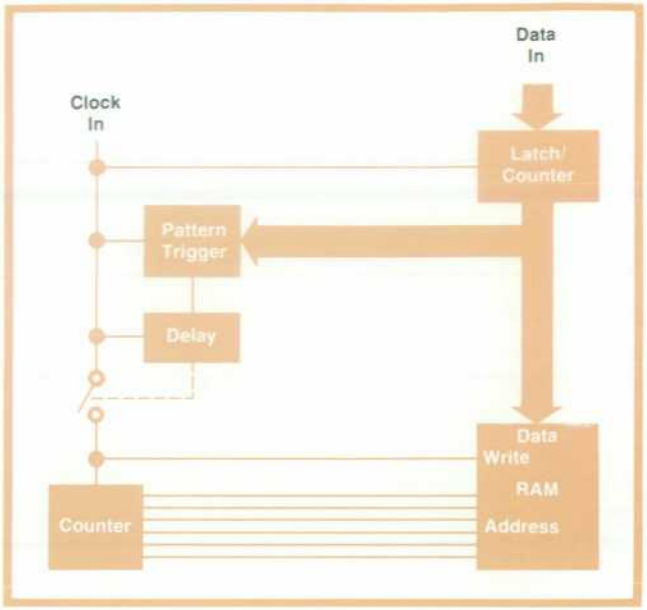


Fig. 6. *Block diagram of a basic logic state analyzer.*

Alternatively, the clock-line switch could be closed initially and then opened when the trigger pattern occurs. The state sequence leading up to the trigger pattern is thus retained for display. Or, a digital delay can be added between the pattern trigger and the clock-line switch so memory loading start or stop is delayed by a selected number of clock periods.

A simplified block diagram of Model 1602A is shown in Fig. 7. Trigger pattern recognition in this instrument is done with two 256×1-bit memories whose outputs are ANDed to provide 16-bit pattern capability. Data from the input probe is latched on the trigger memory address lines by the external clock. If a 1 is stored in each memory at its 8-bit part of the address, a trigger pulse is generated.

Once a trigger pulse is generated, the delay generator is enabled and the trigger memory is inhibited by setting the memory chip-select lines high. If more than zero delay has been selected, the delay generator, which is essentially a counter, is incremented by each subsequent qualified clock pulse. When the delay counts out, if the analyzer is in the trigger-starts-trace mode the data memory address counter is enabled. This fact is reported to the microprocessor. The address counter is then incremented by each qualified clock pulse until the 63 states following the trigger-plus-delay specification are stored in the data memory.

When the completion of a trace is reported to the microprocessor, it takes control of the data memory address counter and accesses the word selected for display by incrementing the address counter. It then generates the LED control signals for displaying the characters one at a time, completing a display scan

© Copr. 1949-1998 Hewlett-Packard Co.

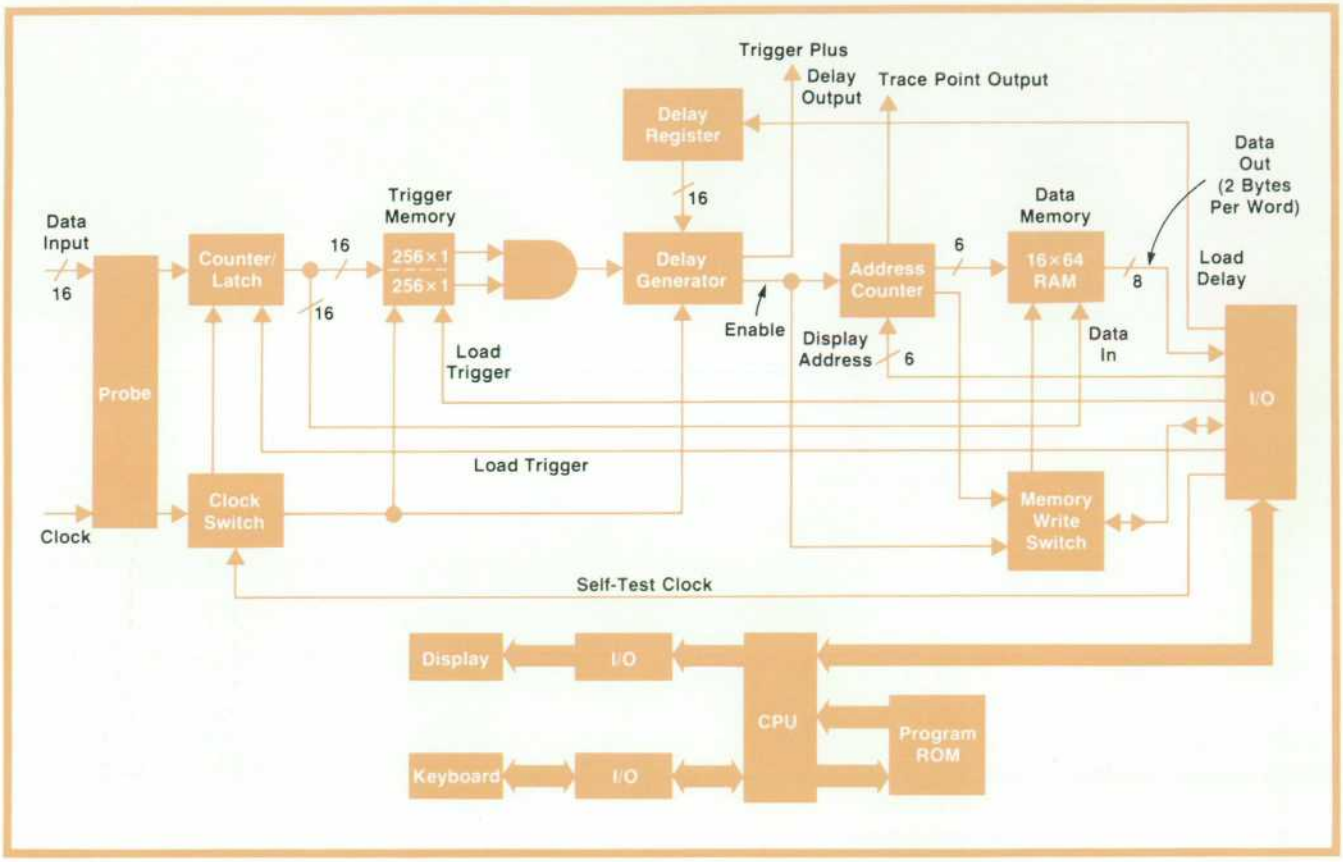


Fig. 7. *Simplified block diagram of Model 1602A Logic State Analyzer. For the sake of clarity, switches for changing the operating mode and various gates for qualifying the clock and trigger pulses have been omitted.*

every 18 ms. It also scans the keyboard every 18 ms.

In the delay-by-events mode, the delay generator is incremented by the trigger pattern only when the trigger pattern occurs. The trigger memories are not inhibited by the delay generator but remain enabled. (The switches for reconfiguring the instrument in this and other modes are not shown in Fig. 7).

In the trigger-plus-delay-ends-trace mode, the data is constantly clocked into the data memory but trigger recognition is inhibited until enough data has been acquired to fill the data memory. Data acquisition then continues, each new state replacing the oldest state in memory until the trigger-plus-delay specification is met. Data acquisition then stops and the microprocessor takes charge of the memory for readout.

The trigger memories are loaded with the trigger pattern by configuring the input data latch as a counter to drive the memory address lines. The microprocessor initializes the counter and supplies clock pulses to increment it. When the counter output is the same as a desired trigger pattern, the microprocessor loads a 1 into the memory at that address. Otherwise it loads a 0.

The delay register is a serial-in-parallel-out register that is reloaded by the microprocessor whenever the delay number is changed. Whenever the delay is counted out, the delay generator is reloaded from the delay register and the trigger memory chip-select lines are set low so trigger pattern recognition is again enabled. The trigger-recognition-delay-count sequence repeats independently of the rest of the instrument, except during data memory loading, so the TRIGGER-PLUS-DELAY output can be used to trigger an oscilloscope or other instrument repetitively. This part of the instrument functions, in effect, as a breakpoint register.

Rear-Panel Outputs

The "trigger plus delay output" indicated in Fig. 7 is a rear-panel output labelled TRIG OUT. It supplies a TTL-level pulse each time trigger-plus-delay conditions are met. It is useful as an oscilloscope trigger when investigating system timing problems.

The TRACE POINT output is also on the rear panel. This output is normally high, going low when the TRACE key is pressed and returning to the high state when trigger conditions are met. This can be used as an interrupt signal and it also enables two or more Model 1602A's to operate in parallel.

© Copr. 1949-1998 Hewlett-Packard Co.

Operating two Model 1602A's in parallel to effect a 32-bit logic state analyzer is accomplished by using the TRACE POINT output of one, designated the "master," to drive the TRIGGER QUALIFIER input of the other, designated the "slave." The trigger word on the slave is then set to all "don't cares" so triggering can be controlled by the master.

Both the TRIGGER and TRACE POINT outputs lag the input clocks by about 150 ns. This means that in parallel operation the data in the slave unit is one clock period behind the master unit, and it also limits the maximum clock rate to about 6 MHz.

Self-Test

Each time Model 1602A is turned on, 22 tests are performed on the data acquisition hardware (except for the probe) using the input latch as a counter driven by a 2-MHz clock to supply test patterns. Also, the microprocessor ROMs are checksummed, the I/O ports are toggled and read back, and operation of the microprocessor RAM is verified. During the tests, the resulting information is compared to references in the microprocessor's ROM. If the test fails, E99 is displayed. The user then refers to the manual to find instructions on how to step through the self test and interpret the displayed messages, which indicate where the problem lies. If the self test finds no problems, the display shows all 8's.

Acknowledgments

Package design was by Bill Fisher. Don Miller did the product design and the probing system. Thanks are due Bill Farnbach for help in defining the instrument and to John Scharrer for starting the hardware design.

Charles T. Small

Chuck Small joined HP in 1966 as a test technician, moved to the R and D lab two years later to build and debug prototype pulse generators, and then joined the logic state analyzer group in 1972, contributing to the design of the Models 1601L and 1600A before becoming project leader for the Model 1602A. Chuck has an AE degree from the Portland (Oregon) Community College and obtained a BS degree in EE and Computer Science from the University of Colorado in 1977. He also taught a logic-circuits lab at UC, using a Model 1602A to teach state analysis. Chuck's a model railroader (HO) and does some woodworking. He and his wife have two children, 1 and 5.

Alan J. DeVilbiss

Al DeVilbiss joined HP in 1965 after working four years on spacecraft sequencers and computers. At HP he contributed to the 1300A Large-Screen Display and the 183A 250-MHz Oscilloscope, and worked with the HP Corporate Engineering Group on heat-transfer problems before doing the HP-IB interface hardware and software design for Model 1602A. Al obtained a BSEE degree from Louisiana Polytechnic University (1960) and an MSEE from the California Institute of Technology (1961). For fun, Al designed a miniature computer for use in enduro motorcycling and sports car rallies (he drives a TR4). He also designed his own home, now being built. He and his wife have a boy, 10, and a girl, 12.

SPECIFICATIONS

HP Model 1602A Logic State Analyzer

CLOCK, DATA, AND QUALIFIER PROBE INPUTS
- REPETITION RATES: to 10 MHz.
- INPUT LOAD: one low power Schottky gate (<400 μA source).
- INPUT THRESHOLD: TTL, fixed at approximately 1.5 V.
- MAXIMUM INPUT: <+5.5 V.
- MINIMUM INPUT
  - LEVEL: >−0.5 V.
  - SWING: from ≤+0.4 V (low) to ≥+2.4 V (high).
  - CLOCK PULSE WIDTH: >25 ns at threshold.
  - DATA SETUP TIME: 35 ns threshold.
  - HOLD TIME: 0 ns.

TRIGGER AND CLOCK QUALIFIER INPUTS(Rear Panel)
- INPUT LOAD: 8 mA maximum source.
- MAXIMUM INPUT: <+5.5 V.
- MINIMUM INPUT
  - LEVEL: >−0.5 V.
  - SWING: from ≤+0.4 V (low) to ≥+2.5 V (high).
  - SETUP TIME: 40 ns with 10250A probe, 10 ns without probe.
  - HOLD TIME: 15 ns with 10250A probe, 30 ns without probe.

TRIGGER AND TRACE POINT OUTPUTS
- HIGH: >2 V into 50Ω.
- LOW: ≤0.4 V into 50Ω.
- PULSE DURATION (width)
  - TRIGGER: high for approximately one clock period.
  - TRACE POINT: sets low when TRACE key is pressed, returns high when trace specification is met.
  - DELAY FROM INPUT CLOCK: <150 ns.

GENERAL

POWER: 100, 120, 220, and 240 Vac; −10% +5%, 48 to 66 Hz; 50 VA maximum.
DIMENSIONS: 107 mm H × 275 mm W × 421 mm D (4.219 × 10.813 × 16.563 in.).
ACCESSORIES SUPPLIED: one external probe pod, one connector with individual clock, ground, and data probe leads with tips.
ACCESSORIES AVAILABLE
- Connectors with and without leads.
- ADAPTER, 10050A: when used as passive connection to 1602A, loads each HP-IB signal line with one Schottky TTL gate (<400 μA source) except DAV which is loaded with two low-power Schottky TTL gates (<800 μA source).
- HP-IB TEST PROBE, 10051A
  - INPUT LOAD: one low-power Schottky TTL gate (<400 μA source) on each HP-IB signal line.
  - INPUT THRESHOLD: TTL fixed at approximately 1.5 volts except DAV, NRFD, NDAC, ATN, EOI which are buffered with low-power Schottky TTL hysteresis gates (positive-going threshold approximately 1.7 V, negative-going threshold approximately 0.9 V).
  - MAXIMUM INPUT: ≤+5.5 V.
  - MINIMUM INPUT: >−0.5 V.
  - DIFFERENTIAL SIGNAL DELAY: signals on ATN and EOI lines are delayed approximately 30 ns more than DIO 1-8, SRQ, IFC, and REN which are applied to 1602A data inputs without buffering.
  - DATA SETUP TIME: 35 ns.
  - DATA HOLD TIME: 50 ns.

OPTIONS: 001: HPIB Interface.
PRICES IN U.S.A.: $1600, Opt 001, add $300. 10050A HP-IB Adapter, $35. 10051 HP-IB Test Probe, $185.
MANUFACTURING DIVISION: COLORADO SPRINGS DIVISION
P.O. Box 2197
Colorado Springs, Colorado 80901 U.S.A.

© Copr. 1949-1998 Hewlett-Packard Co.

Adapting the 1611A Logic State Analyzer to Work with the F8 Microprocessor Family

Microprocessors are not all alike. Adapting a dedicated instrument to test several different kinds of microprocessors poses some interesting challenges but also provides opportunities to broaden capability.

by Deborah J. Ogden

GREATLY SIMPLIFIED ANALYSIS of program flow in microprocessor systems was made possible by the use of "personality" modules in the Model 1611A Logic State Analyzer[1] (Fig. 1). Each personality module customizes the logic state analyzer for a particular microprocessor by providing input probes whose pin connections, trigger levels, and clock slope match the microprocessor's, and by providing a capability for converting the machine language of the microprocessor into mnemonic language (Fig. 2). The probes greatly simplify set-up of the instrument while display of the microprocessor's instructions in mnemonic language makes it much easier to interpret the "snapshot" of executing program steps captured and retained for display by the logic state analyzer.

Two personality modules, one for 8080 and one for 6800 microprocessors, were available at the time of the 1611A's introduction. Personality modules for Z80 and F8 microprocessors have since been designed. Adapting the analyzer to this variety of microprocessors posed a variety of problems.

Personality Requirements

To fit into the existing 1611A Logic State Analyzer structure, a personality module needs four parts: a probe, a front-panel control organization, a personality board, and a ROM board (Fig. 3). The probe interfaces with the microprocessor system, buffering the signals supplied to the logic state analyzer. The personality board latches microprocessor data, addresses, and status information with the help of

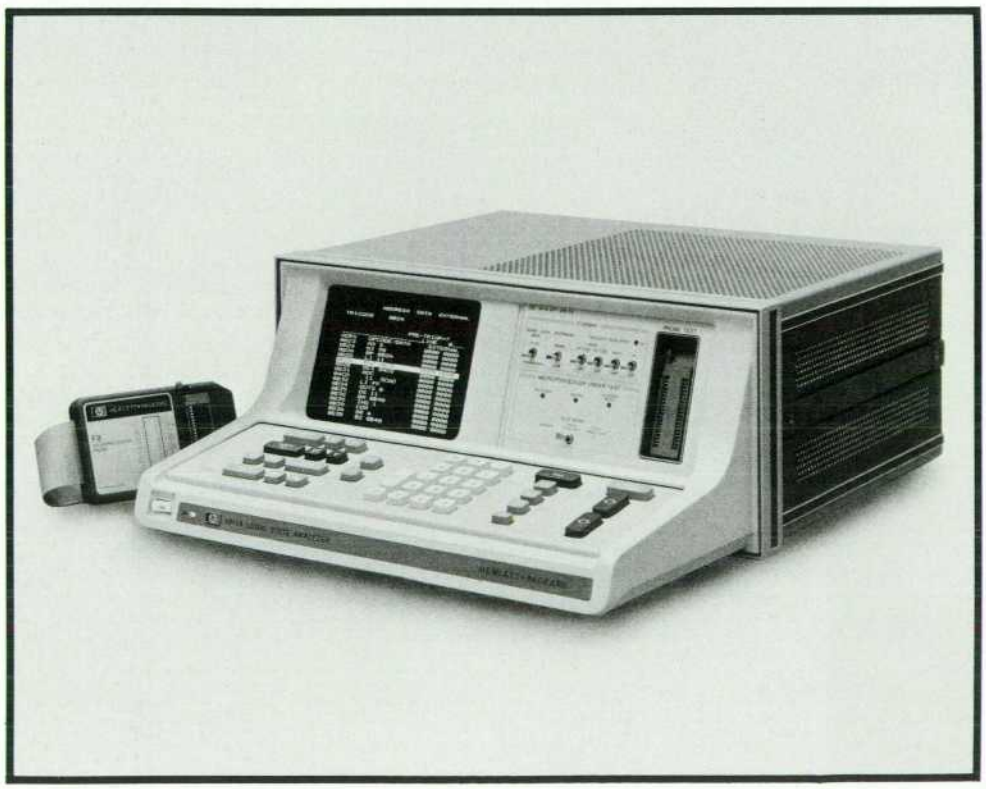


Fig. 1. *Model 1611A Logic State Analyzer uses "personality" modules to configure the instrument for testing digital systems designed around particular microprocessors. Each personality module establishes the appropriate pin connections for the probe, sets the proper clock slope and threshold levels, and encodes the microprocessor's data bus contents into the mnemonic code used for that microprocessor.*

© Copr. 1949-1998 Hewlett-Packard Co.

Fig. 2. *Each personality module enables Model 1611A to display the contents of a microprocessor's data bus in the microprocessor's mnemonic code. This allows the user to rapidly scan large blocks of code to determine if a program is executing properly.*

clocks and control signals from the microprocessor, and at the end of each microprocessor machine cycle it provides clocks to the logic state analyzer for storing the output of these latches. Other functions of the personality board are to control halting of the microprocessor, provide status information to the front panel, and generate a self-test routine for the logic state analyzer.

The front panel displays status information, such as NO CLOCK , and it has various switches such as those that select normal or single-step program tracing and the hexadecimal or octal display format. The ROM board has all the software needed by the logic state analyzer including the inverse assembler that translates the microprocessor machine code into mnemonics.

All personality modules are required to be passive to the system under test. This is because a major advantage of a logic state analyzer over development systems is that it can help in design and debugging a system without having any effect on the system, other than adding a small amount of capacitance to the lines being monitored and drawing a small amount of current for driving the personality module's buffers. The only active role the logic state analyzer plays is halting and single-stepping the microprocessor. Otherwise, it merely "takes snapshots" of the program flow for display.

Attributes of the F8

The personality module for the F8 series proved to be an interesting challenge. This is because the F8 is structured differently from other microprocessors. It was designed so only a few components are needed for simple systems but expansion is possible for more complex systems. Particularly challenging from the logic state analyzer point of view was the absence of a separate address bus for the F8 CPU and the lack of a wait, halt, or bus-request line for suspending microprocessor operation. Yet a design goal for the F8 personality module was similarity to other modules so a user familiar with logic state analyzer operation with other modules would not have to relearn instrument operation.

The CPU of an F8 system has a 64-byte RAM that is used as a scratch-pad memory but all other memory (RAM or ROM) is in external components. Each of the ROM components, called program storage units (PSU), has a program counter and a data counter for accessing its own internal data. Data is transferred on a bidirectional 8-line data bus with data flow controlled by a 5-line bus (ROMC) that identifies the type of information on the data lines and what the satellite components must do with it during each instruction cycle, such as write data-bus contents to memory, place contents of I/O port on data bus, add data-bus contents to data counter, and so on.

Since a separate address bus is not required, each of the CPU and PSU components has 16 pins available for two 8-bit I/O ports. The simplest F8 system therefore needs only two components, a CPU and one PSU. If more than 1K of ROM is needed, more PSUs can be

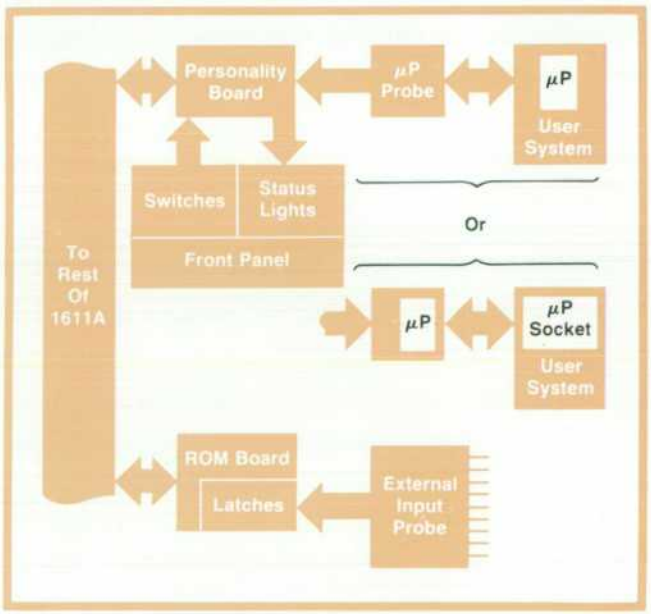


Fig. 3. *Block diagram of a personality module for the Model 1611A Logic State Analyzer. The microprocessor probe can clip onto the microprocessor in the user system with a "clothes-pin" clip, or the probe can plug into the microprocessor socket with the microprocessor plugged into a socket on the probe body.*

© Copr. 1949-1998 Hewlett-Packard Co.

added, each of these contributing two more I/O ports. Other F8 devices, the static memory interface (SMI) and dynamic memory interface (DMI), output addresses to external RAMs and ROMs. These also have internal program and data counters, but no separate I/O ports. A multi-unit F8 system may then have several program and data counters, but they must all contain the same values. This synchronization is performed by the ROMC lines under control of the CPU.

New Features

The absence of a separate address bus allowed a useful feature to be added to the logic state analyzer's F8 personality module. Since the CPU package has 16 pins for the two I/O ports, data on these pins can be supplied through the analyzer's microprocessor probe directly to the analyzer's eight external input lines. If the user wishes to monitor these I/O pins, he does not have to make individual connections to them with the analyzer's external probe. A three-position front-panel switch determines whether I/O port 0, I/O port 1, or the external probe supplies the data that is written in the external field on the analyzer display.

A bank of trigger qualifier switches is another new feature. A set of flags derived from the ROMC lines by the personality module can qualify trigger words so triggering occurs only when the selected qualifiers are true. The qualifiers are READ OP CODE, which allows triggering only on op-code fetch machine cycles, READ $\overline{\text{OPCODE}}$, which allows triggering on any read cycle except op-codes, WRITE, which allows triggering only on memory-write cycles, and I/O, which allows triggering only on I/O cycles. These trigger qualifiers have proved so useful that they have been designed into the other 1611A personality modules.

Personality Module Operation

The basic timing of the personality board, whose main function is grabbing data, address, and status information, was designed around the timing specifications for the F8 CPU. During program execution, each event in the microprocessor system is triggered by a CPU-generated clock pulse called WRITE. When an op-code is placed on the data lines, all the ROMC lines are at the zero level and the information on the data lines is valid on the falling edge of the WRITE clock. When ROMC $\neq 0$, information on the data lines is valid on the rising edge of the WRITE clock. The personality module must therefore monitor the ROMC lines to decide on which edge to clock the data, then latch it during the small time window where it is guaranteed valid. Flags are also generated by the ROMC lines to indicate the transaction type. The circuitry for doing all this is diagrammed in Fig. 4.

Because the 1611A Logic-State Analyzer normally monitors microprocessor address and data lines simultaneously, some means of tracking the F8 addresses is needed. This is accomplished by using one of the F8 system's own SMI (static memory interface) chips within the F8 personality module to monitor the data lines and generate outputs according to the states of the ROMC lines. The flags generated by the ROMC lines (Fig. 4) then tell the personality module when to latch the information on the data lines as data and when to latch the output of the SMI as an address. The control inputs to the SMI are set so it cannot drive the data bus or respond to an interrupt; it serves merely as a passive monitor.

The 1611A Logic State Analyzer normally provides means of halting a microprocessor at the end of a trace

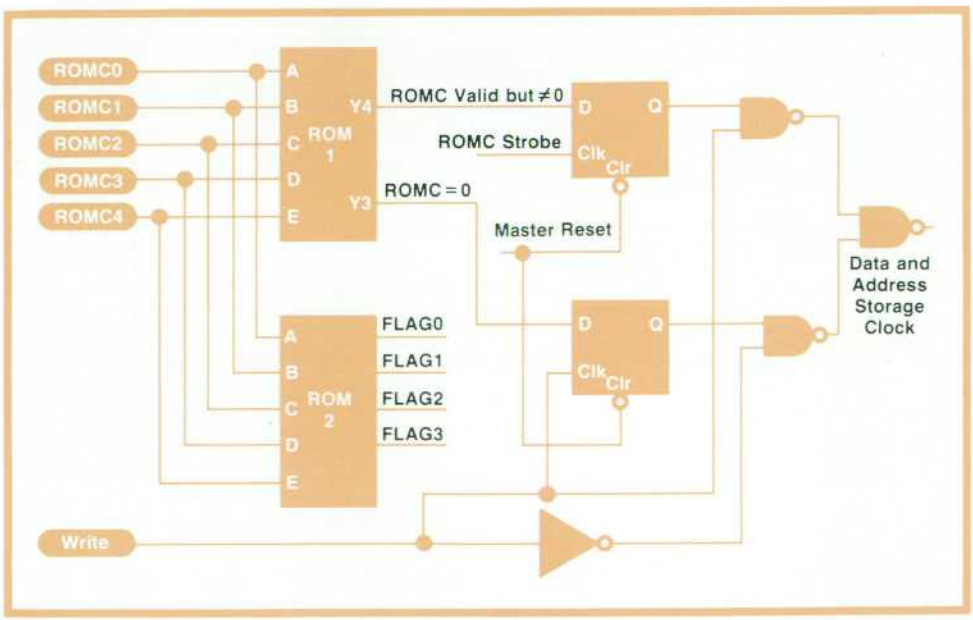


Fig. 4. *Logic state analyzer clock is derived from the external microprocessor WRITE pulse by these logic circuits that detect whether clocking is to occur on the rising or falling edge of the WRITE pulse.*

© Copr. 1949-1998 Hewlett-Packard Co.

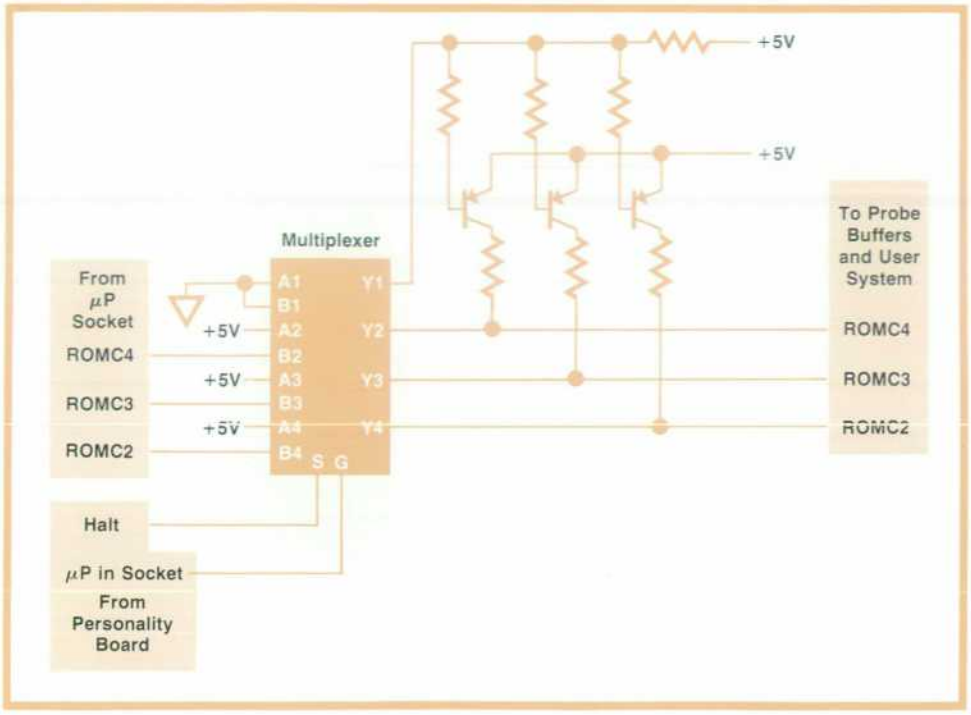


Fig. 5. *Halt circuits intercept the ROMC signals to supply a no-op (11100_2) to the rest of the external, microprocessor system. A no-op instruction is also placed on the data bus.*

(data capture cycle). This gives the user an opportunity to examine program execution in blocks of up to 64 steps before asking the microprocessor to continue. The halt capability, used in the single-step mode, also enables the user to go through a program one step at a time.

Since the F8 CPU has no halt input, the F8 personality module accomplishes halt indirectly. To enable this capability, the CPU must be plugged into the probe body and the probe tip plugged into the microprocessor's socket in the system being tested. This enables the personality module to break into the ROMC lines and supply a no-op (ROMC = 11100_2) to the rest of the system when an opcode fetch (ROMC = 00000_2) is output by the CPU. About 100 ns after this break, which gives time for all the external units to place their data lines in the high-impedance state, the personality module places a no-op instruction on the data lines. Until the halt is released, the CPU executes a no-op instruction repetitively but the program counters in the rest of the system will not be incremented because of the no-op on the ROMC lines.

Fig. 5 is a diagram of the halting circuitry. Only the three most significant bits of the ROMC lines are affected because the two least significant bits are zero for both an op-fetch and a no-op transaction. The three lines are run through a multiplexer that switches either the ROMC lines (B inputs) or a "high" state (A inputs) to the Y outputs, depending on the state of the HALT line (S input). In either state, the Y1 output grounds the bases of transistors that pull up any "high" output line to 3.6V for the benefit of the F8 CMOS circuits. This pull-up is needed because the multiplexer, which is low-power Schottky, outputs a high of only 2.4V. Low-power Schottky is used here to obtain a sufficient speed advantage over CMOS.

If the F8 CPU is not in the probe socket, the μP-IN/SOCKET line turns off the multiplexer. The Y outputs then go to a high-impedance state and the pull-up transistors turn off so as not to interfere with any activity on the ROMC lines. The μP-IN-SOCKET line also disables the halting circuitry and displays an error message (HALTING NOT ALLOWED) if the front-panel test mode switch is set to TRACE THEN HALT or TRACE SINGLE STEP. The μP-IN-SOCKET line is enabled by a circuit that senses the voltage drop across a 1-ohm resistor in series with the +5V supply line to the microprocessor

Deborah J. Ogden

An early interest in mathematics led Debbie Ogden to a BS degree in computer science at North Carolina State University and a job writing programs for manipulating three-dimensional computer graphics in the University's EE lab. This generated an interest in computer hardware problems which led to an MSEE degree (1976) and a job at Hewlett-Packard's Colorado Springs Division. At HP, Debbie's first assignment was the F8 personality module. Outside of working hours, Debbie enjoys outdoor activities like skiing or back-packing, and recently she's taken up woodworking in a local adult educational program with the goal of making furniture.

© Copr. 1949-1998 Hewlett-Packard Co.

socket. Current is drawn through this resistor only when a microprocessor is in the socket of the probe body and the probe is correctly connected to a powered system.

Acknowledgments

Roger Molnar did the product design for the F8 personality module (Model 10259A). Sheila McCullough provided the pc board layout. Jeff Smith, Bill Farnbach, and Justin Morrill contributed valuable engineering ideas, and Chuck House, logic analyzer department manager, inspired everyone involved to make the extra effort needed to bring this product to the marketplace in good time.

Reference

1. J.H. Smith, "A Logic State Analyzer for Microprocessor Systems," Hewlett-Packard Journal, January 1977.

SPECIFICATIONS

Option 0F8 for HP Model 1611A Logic State Analyzer

NOTE: Option 0F8 (Model 10259A) is compatible with any microprocessor that meets the specifications of the Fairchild F8.

CLOCK AND WRITE

CLOCK RATE: 100 kHz to 2 MHz.
WIDTH: 180 ns minimum for either high or low state.
INPUT CURRENT: approximately 50 μA, logic high and low.
INPUT CAPACITANCE: approximately 25 pF.
THRESHOLD: 2.4 V to 5.5 V, logic 1 (high); −0.8 to 0.8 V, logic 0 (low).
WRITE PERIOD: either 4 or 6 times the clock period.
WRITE PULSE WIDTH: maximum = clock period, minimum = clock period − 100 ns.

ROMC

INPUT CURRENT: approximately 220 μA, logic 0 (low); approximately 40 μA, logic (high).
INPUT CAPACITANCE: approximately 25 pF.
THRESHOLD: 2 V minimum logic 1 (high); 0.7 V maximum logic 0 (low).
SETUP TIME: 200 ns minimum relative to second falling edge of ϕ after WRITE goes low.
HOLD TIME: 80 ns minimum relative to falling edge of WRITE.

DATA, I/O0, I/O1, EXT RES

INPUT CURRENT: approximately 200 μA, logic 0 (low); approximately 20 μA, logic 1 (high).
INPUT CAPACITANCE: approximately 25 pF including 30.4 cm (12 in.) cable.
THRESHOLD: 2 V minimum logic 1 (high); 0.7 V maximum logic 0 (low).

DATA SETUP AND HOLD TIMES

If ROMC = 0, times are relative to falling edge of WRITE; setup, 200 ns minimum; hold, 50 ns minimum.
If ROMC ≠ 0, times are relative to rising edge of WRITE; setup, 350 ns minimum; hold, 50 ns minimum.
1/O0 AND 1/O1: setup, 300 ns minimum; hold, 50 ns minimum.

EXTERNAL PROBE INPUTS

INPUT CURRENT: approximately 50 μA, logic 0 or logic 1.
INPUT CAPACITANCE: approximately 25 pF at probe tip.
THRESHOLD: 2.4 V to 5.5 V, logic 1 (high); −0.8 V to 0.8 V, logic 0 (low).
SETUP TIME: 150 ns minimum relative to rising edge of WRITE for ROMC ≠ 0, or to falling edge of WRITE for ROMC = 0.
HOLD TIME: zero relative to rising edge of WRITE for ROMC = 0, or to falling edge of WRITE for ROMC = 0.
NOTE: all inputs have hysteresis.

HALTING: If 1611A Test Mode Switch is in TRACE THEN HALT or TRACE SINGLE STEP and F8 CPU is not in socket on probe with probe connected to an energized user system, then halt is not possible. 1611A CRT will display: Halting Not Allowed.

OUTPUTS

LOW: <0.4 V into 50Ω.
HIGH: >2.0 V into 50Ω (nominally 3.9 V into open circuit).
TRIGGER: duration, approximately 75 ns (RZ format); delay, approximately 350 ns after rising edge of WRITE if ROMC ≠ 0, and approximately 350 ns after falling edge of WRITE if ROMC = 0 during cycles that define a valid trigger.
TRACE POINT (⎍): provides positive edge approximately 350 ns after rising or falling edge of WRITE (as explained for Trigger Output) during cycle that defines specific valid trigger to be displayed on 1611A. If 1611A delay is set such that trigger word is not displayed, TRACE POINT OUTPUT occurs for cycle that defines valid word immediately preceding first displayed word.
TRACE POINT (⎍): complement of TRACE POINT (⎍).

1611A General

MEMORY DEPTH: 64 data transactions; 16 transactions are displayed at one time; roll keys permit viewing all 64 transactions.
TIME INTERVAL: accuracy, 0.1% ±1 μs. Maximum time, $2^{24}-1$ μs (16.7 s).
EVENTS COUNT: $2^{24}-1$ events (16.7 million) maximum.
LOGIC PROBE OUTPUT POWER: 5 Vdc at 0.1 A maximum.
POWER: 100, 120, 220, 240 Vac; −10% +5%; 48 to 440 Hz, 120 VA maximum.
DIMENSIONS: 206 mm H × 426 mm W × 522 mm D (8.125 × 16.75 × 25.5 in.).
WEIGHT: 15 kg (33 lb).
ACCESSORIES SUPPLIED: one microprocessor probe, external 8-bit probe; one 40-pin clip with 30.5 cm (12 in.) cable, one 40-pin male socket with 30.5 cm (12 in.) cable; one 40-pin male socket with 7.6 cm (3 in.) cable.
PRICE IN U.S.A.: Model 1611A Logic State Analyzer with Opt 0F8, $5200. 10259A Personality Module for Field Installation, $1250.
MANUFACTURING DIVISION: COLORADO SPRINGS DIVISION
P.O. Box 2197
Colorado Springs, Colorado 80901 U.S.A.

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

HEWLETT-PACKARD JOURNAL

FEBRUARY 1978 Volume 29 • Number 6

Technical information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
Amstelveen-1134 The Netherlands
Yokogawa-Hewlett-Packard Ltd., Shibuya-Ku
Tokyo 151 Japan

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

CHANGE OF ADDRESS: To change your address or delete your name from our mailing list please send us your old address label (it peels off). Send changes to Hewlett-Packard Journal, 1501 Page Mill Road, Palo Alto, California 94304 U.S.A. Allow 60 days.

© Copr. 1949-1998 Hewlett-Packard Co.