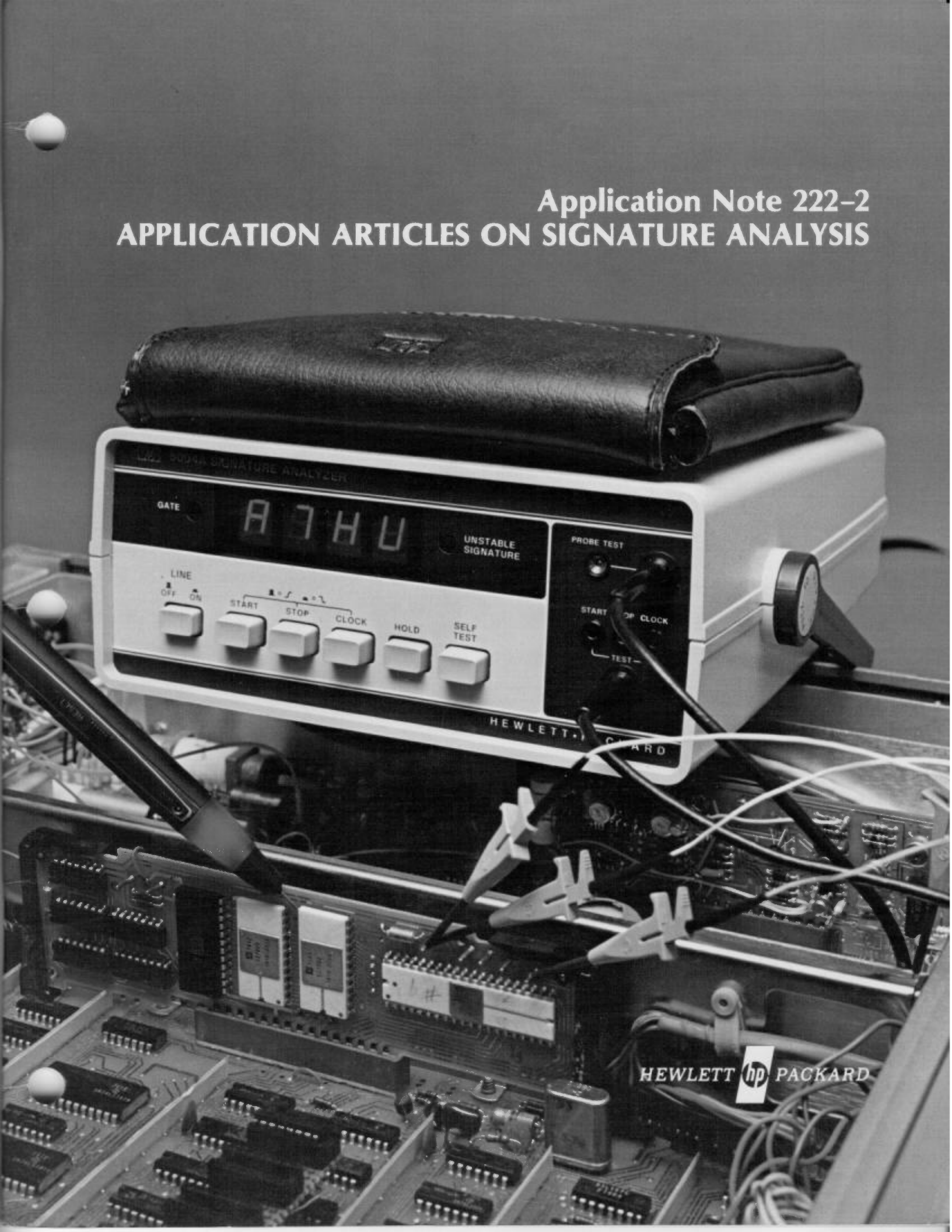
Application Note 222-2
APPLICATION ARTICLES ON SIGNATURE ANALYSIS



HEWLETT hp PACKARD

APPLICATION NOTE 222-2

APPLICATION ARTICLES ON SIGNATURE ANALYSIS

W1RG

FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS PUBLICATION

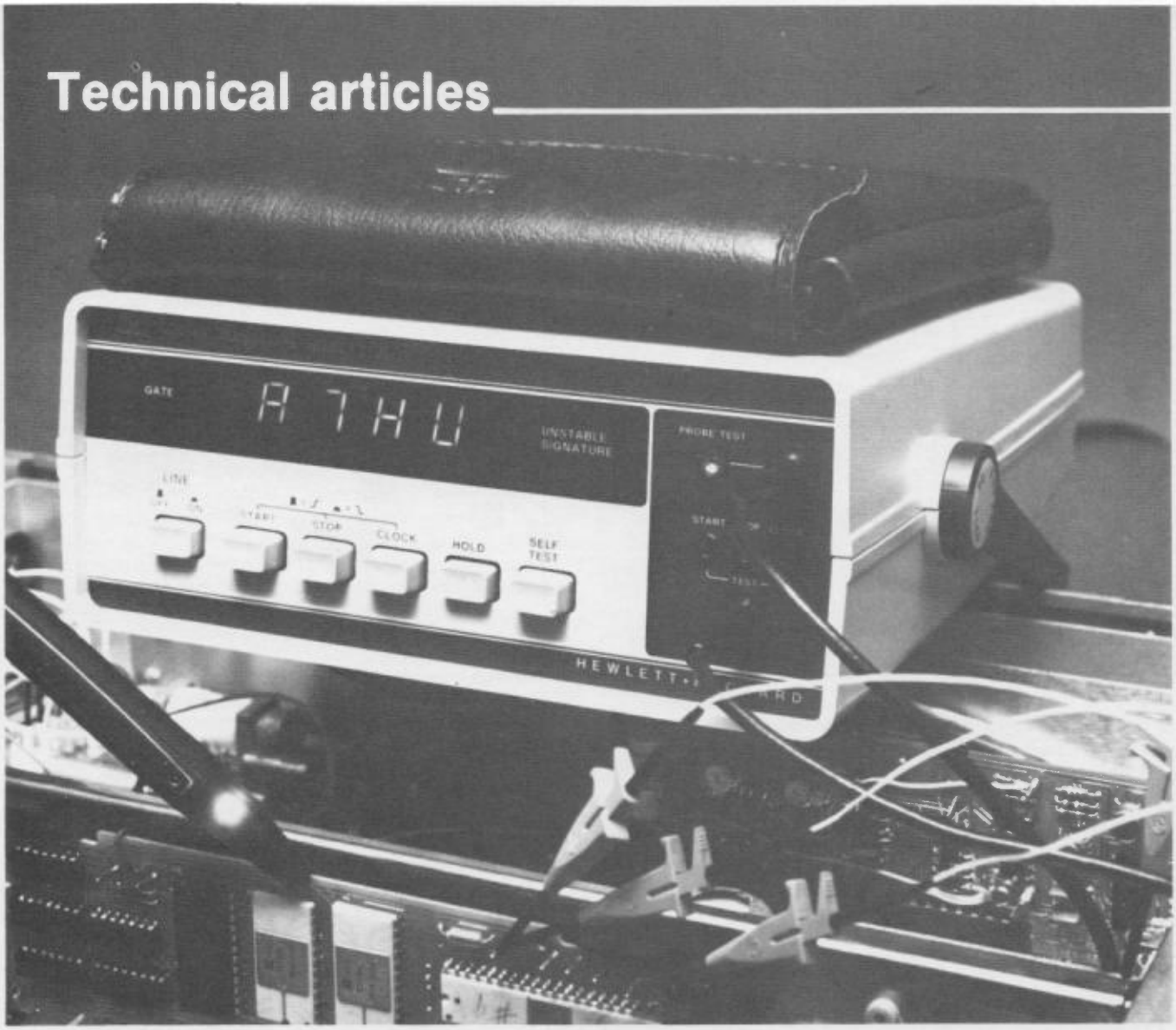
This is a collection of eight technical articles on Signature Analysis applications. It is intended to assist you in designing or retrofitting your digital products for S.A. troubleshooting. Check the annotated Table of Contents and choose those articles which are of interest in your application.

ABOUT OTHER PUBLICATIONS

For additional background on Signature Analysis, check these HP publications:

1. 5004A Signature Analyzer Data Sheet, HP publication 02-5952-7464.
2. 3060A Board Test System Data Sheet, HP publication 5952-8776.
3. Application Note 222, *A Designer's Guide to Signature Analysis*, HP publication 02-5952-7465.
4. Application Note 222-1, *Implementing Signature Analysis for Production Testing*, HP publication 5952-8785.

TABLE OF CONTENTS



Hexadecimal signatures identify troublespots in microprocessor systems

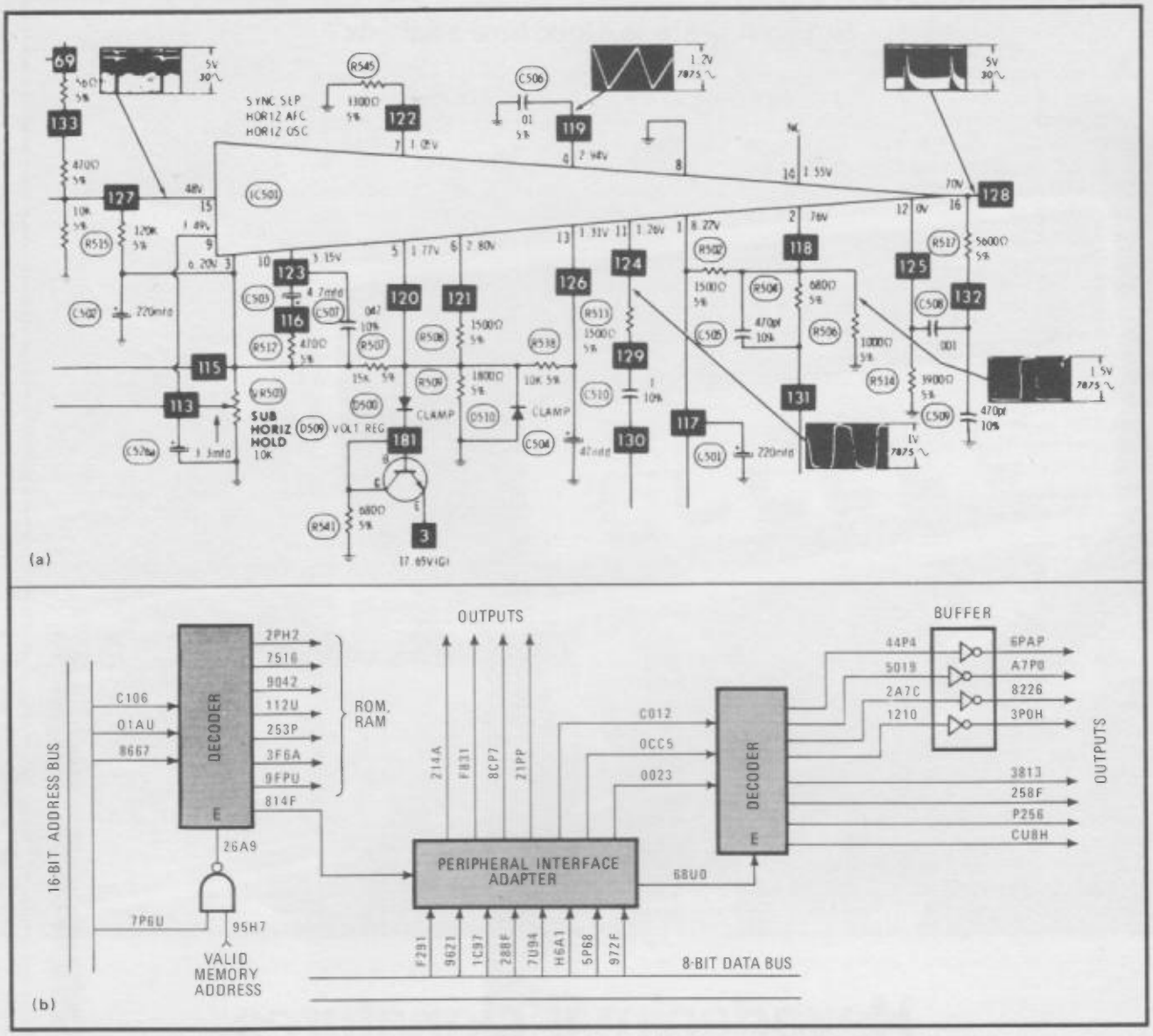
by Gary Gordon and Hans Nadig, *Hewlett-Packard Corp., Santa Clara, Calif.*

☐ The number of microprocessor-based products manufactured each month is approaching the total of all installed computers and minicomputers, raising the question: "How will they be serviced?" Traditional digital servicing, in which defective modules are swapped for good ones, creates substantial inventory and handling costs. A much more economical alternative is a new technique called signature analysis, with which a product can readily be serviced down to the component level.

Signature analysis is based on the time-honored technique of signal tracing. When its requirements are designed into a product, a new test instrument can map lengthy bit streams from the product into short four-digit hexadecimal "signatures." These the technician compares with the correct signatures noted on the system's circuit diagram. If a bit stream is faulty, he traces it back through gates and memory elements until he can isolate an element with correct inputs but faulty outputs. The method has a 99.998% certainty of spotting a faulty bit stream, regardless of its length or the subtlety of its faults.

Signature analysis is more than a new measurement technique. It is a wholly new service philosophy, for the decision whether to adopt it requires a thorough evalua-

Reprinted from **Electronics**, March 3, 1977



1. A new form of digital ID. Diagrams of analog circuits, as for a television set (a), have voltage levels and waveforms noted on them to aid service technicians (circuit is taken from a Howard W. Sams Photofacts service manual). But in digital circuits, all waveforms look alike, and the schematic shown here (b) is annotated instead with hexadecimal signatures that characterize the bit streams unique to each circuit node. A new test instrument derives actual signatures for the technician to compare with the correct ones on the schematic.

tion of its applicability while a product is still in the development stage.

The final decision will usually be based on economics. On the plus side, the technique:

- Increases a product's value to the customer by lowering his cost of ownership.
- Lowers warranty costs for the manufacturer.
- Lowers production-line troubleshooting costs, because it simplifies the job of isolating the speed-related failures, which cannot be caught by present board testers.
- Lowers materials costs, because the product no longer needs to be divided into replaceable modules.

On the minus side, designing signature analysis into a system does add back some other material expenses and also lengthens development time.

On balance, though, the pluses outweighed the minuses in Hewlett-Packard's first applications of the technique to two digital voltmeters. As will be described later, development time and costs rose roughly 1%. But the return on this investment was substantial. Factory costs dropped, since the 1% or so increase in parts cost caused by adding jumpers and extra read-only-memory space was more than offset by the roughly 5% decrease in other materials costs. As a matter of fact, HP has opted for signature analysis for the majority of its upcoming microprocessor-based instruments and other data products.

Analog roots

Signature analysis reverts to the advantages of analog-equipment servicing. There, too, the serviceman uses an annotated schematic (Fig. 1a), which clearly shows voltages and waveforms and allows him to make repairs without a detailed understanding of the circuits. By

How accurate is signature analysis?

For any technique intended to pick up errors in bit streams, it is important to have a measure of the accuracy with which it performs that function. To calculate the accuracy of signature analysis, the first step is to define an error bit stream as a hypothetical sequence related to an erroneous data bit stream in the following way: in the error bit stream, the bit or bits that are in error in the data stream show up as 1s, while bits not in error—regardless of whether they are 1s or 0s in the data stream—show up as 0s. Then, in, say, a 500-bit data sequence, if bit 42 is in error—whether it is a 1 or a 0—bit 42 of the corresponding error bit stream will be a 1 and all 499 other bits will be 0s.

The second step is to invoke the principle of superposition, which is applicable because the feedback shift register is linear. This states that the response of the register to the sum of two inputs is the same as the sum of its responses to the individual inputs. (Note that superposition is used only for this analysis and is not used in the actual instrument.)

From this it follows that if the register input is considered to be the sum (modulo 2) of two sequences—a data bit stream and an error bit stream—then the signature it should display will be equivalent to the sum (modulo 2) of the individual signatures.

Consider this case of summing the two signatures. If there are one or more errors in the data bit stream, the error bit stream will contain 1s in those locations. Then, the sum of their signatures should be different from the signature of the data bit stream itself. It follows that the signature of the error bit stream must be anything but 0 for errors to be detected. This deduction then leads us to examine the conditions under which the signature of the error bit stream becomes 0—the case where errors would go undetected.

With a 16-bit shift register the error bit stream's signature is never 0 for streams of 16 bits or less that contain a 1. This happens because the first 1 to enter the register never has time to leave it before the signature is complete and can never be canceled by a fed-back bit. Thus all errors are caught.

For length 17, one error-bit-stream sequence will be missed; that which starts with a 1, and then has a 1 present at each bit-time where the first 1 is fed back, thus canceling each subsequent error bit. Then, as the 17th bit enters the register, the first and only remaining error bit exits from the register. Thus the signature will be 0 even though there were 1s in the original error bit stream. This means that with 17-bit-sequence inputs, one of the 2^{17} possible combinations may be in error and will not be caught. For length 18, three are missed; for length 19, seven are missed; and so on.

In general, the percentage probability of catching an error in a sequence length m with register length n is:

$$100 - 100[H(m-n)][(2^{m-n}-1)/(2^m-1)]$$

where H is the step function (required to make the function zero for sequences of n or less).

In more useful terms, with register length n equal to 16, the error is always less than 1 in 2^{16}, regardless of m, the length of the input stream (as m gets very large, the error term approaches 2^{-n}). This gives rise to the certainty of 99.998% that an error, if present, can be spotted.

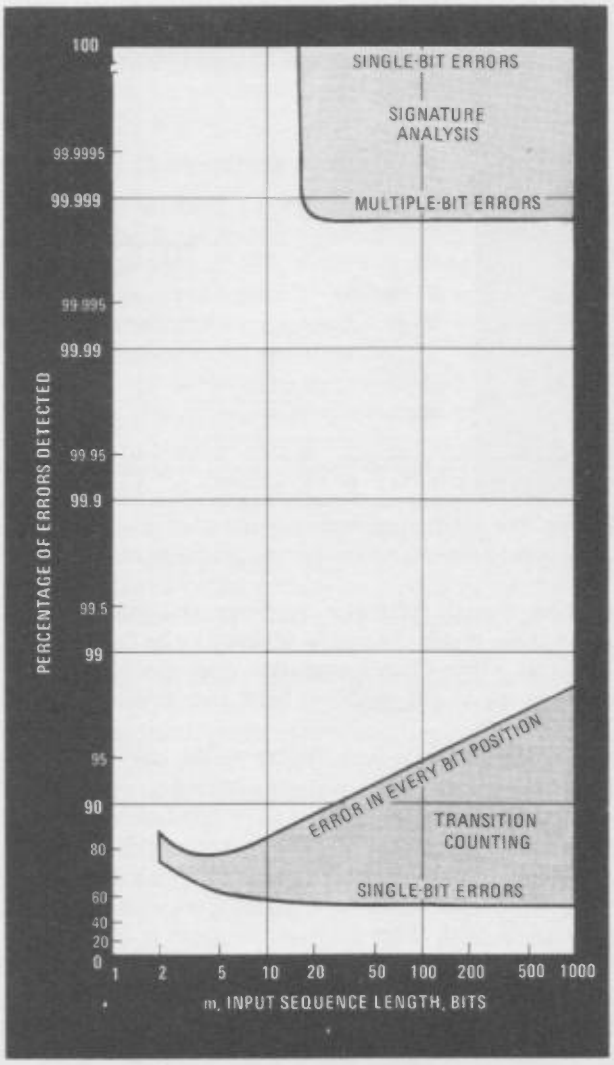
For transition counting, the corresponding probability that it will detect an error in a sequence of length m is expressed by the equation:

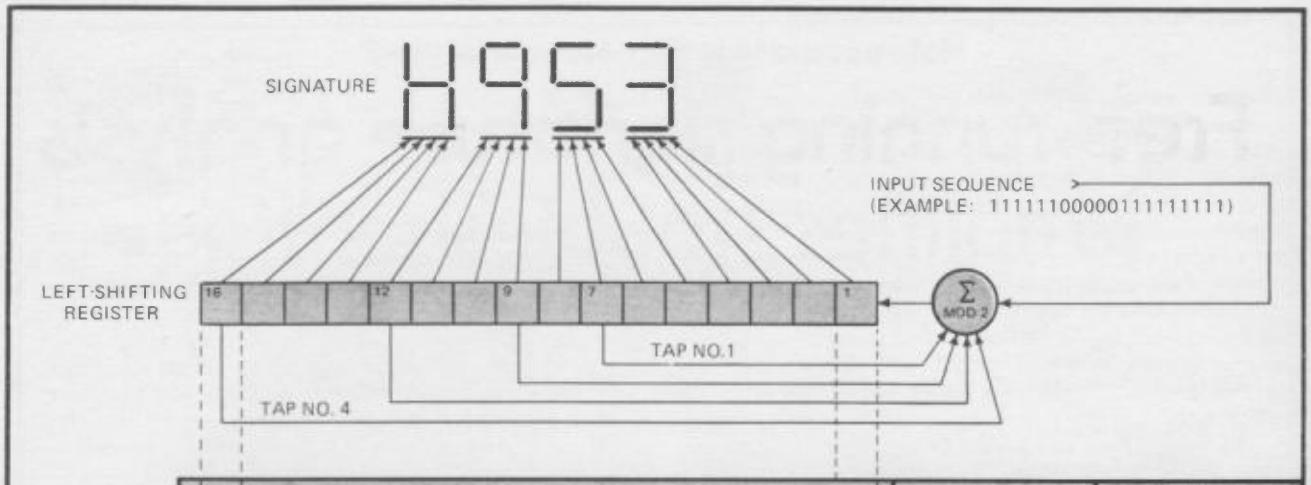
$$P(\%) = 100 - \left[\frac{100}{2^m(2^m-1)}\right] \sum_{r=0}^{m} \left[\frac{m!}{(m-r)!r!}\right]\left[\frac{m!}{(m-r)!r!} - 1\right]$$

where r is the dummy variable. This function increases with increasing m (see graph).

Although these two equations are valid when errors affect the entire bit sequence, that situation is seldom encountered, even in board testers with algorithmic stimulus generators. More typically, digital errors are subtle and affect only a few bits. This is especially true when one is looking for faulty bit streams some number of gates away from the source of a fault, as happens in the course of backtracing.

Thus an equally meaningful figure of merit for signatures is their ability to catch the most subtle of all faults—the single bit error. Signature analysis really excels here, catching every such error. Transition counting, in comparison, misses (m-1)/2m of the errors, nearly 50% for lengthy sequences. Logarithmic graph paper, in fact, is required in the figure to show the formidable superiority of signature analysis.





	CONTENTS OF SHIFT REGISTER																NEXT INPUT TO SHIFT REGISTER	NEXT INPUT SEQUENCE BIT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
7	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0
8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0
9	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0
10	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0
11	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	1	1
12	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1
13	0	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	1
14	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1
15	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	1
16	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1
17	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1
18	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1
19	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1	1
20	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1	—	—

TIME

2. Shifting. The basic component in the signature analyzer is a linear shift register with feedback. This converts the bit stream from a particular circuit node into a four-digit hexadecimal signature. The table shows how a 20-bit input sequence is processed.

comparing actual voltmeter readings and oscilloscope traces to those displayed on the schematic, he determines the point at which circuit operation becomes faulty and from there traces the problem back to a failed component in the unit.

But in the programmed digital world, service schematics are devoid of waveform, voltage, and other service information, for the not very comforting reason that all bit streams look pretty much alike on an oscilloscope. The problem is compounded with microprocessors, state machines, and controllers, for a more subtle reason: with them there is no longer a one-to-one association between product features and particular sections of hardware. For example, if a keyboard-debouncing function fails in an older random-logic product, a service manual might advise checking the integrated circuits that control that function. With microprocessors, on the other hand, key debouncing is more likely a time-shared function tying up the whole processor for a brief moment. When it fails, any one of a large number of ICs could be faulty.

The price of board exchange

As a solution, subdividing the circuitry into replaceable modules has worked reasonably well till now. For one thing, board exchange places relatively few demands on the technical abilities of the serviceman. For another, it is and will remain the fastest way to make repairs when down time is costly, as in large computers and process-control systems. A further advantage is the economy of scale inherent in centralizing repair at the manufacturing site.

But these advantages carry a price. The economy of scale is offset by substantial administrative and inven-

tory costs. A manufacturer may have up to 5% of his assets tied up in service-module inventory, which includes both replacement-board kits and "float" boards in round trip to the factory or waiting in bins. Administrative and handling costs for such a program can also be quite high, particularly as a product approaches obsolescence in the marketplace.

Also, the problem looms of faulty boards in the loop. "Soft" or system-related failures are difficult to detect at repair centers, some of which have reported "no problem found" on 50% of certain returned boards.

Finally, board-level repair is particularly unattractive for supporting products overseas. Turnaround times for modules stretch way out, and import duties of several times the price of the module are often encountered.

For these reasons, centralized board-exchange programs are being widely reevaluated. A partial answer is to move service closer to the customer. For very high-volume products, where board exchange and automated test remain popular, many companies are moving their repair to outlying depots. For lower-volume products where up time is again critical, signature analysis is viable for depot repair of exchanged boards.

Signature analysis was primarily conceived, however, as a more radical change in service strategy. It is a way to substantially reduce repair costs on microprocessor-based products and ROM-based controllers that can stand a few extra hours of down time. Most instruments, computer peripherals, point-of-sale terminals, desk-top calculators, video games, and future citizens' band and television applications fall into this category. The list also includes equipment for which backups are often available: controllers, communications or military equipment, and some of the newer digital products as diverse as taxi-meter and gas-pump controls.

Here the administrative simplicity and cost savings of signature analysis are quite compelling. Troubleshooting by this method requires only a universal $1,000 test instrument, the HP 5004A signature analyzer (see p. 95), and the service manual of the product, which of course must have been specially designed to contain the necessary modest self-test program.

Deriving the signature

Everything depends on the signature. A kind of compressed "fingerprint" of the data present on a node, it is compared with the correct signature printed on the schematic of the product (Fig. 1b) so that any discrepancy may be noted and traced to the source. Clearly, the major figure of merit for any such signature must be the accuracy with which it allows faulty bit streams to be spotted.

HP realized some years ago the potential of the signature-tracing method and investigated numerous ways in which bit streams could be compressed or mapped into signatures. Possibilities are transition counts, 1s counts to generate check sums, and even entropy, the communications measure of information. But a linear-feedback shift register, the method eventually chosen, does a superior job in this regard (see "How accurate is signature analysis?" p. 91).

Linear-feedback shift registers have been used as generators of pseudorandom binary sequences in cryptography, mechanical-vibration control, communications channel testing, and digital radar. But for signature analysis, the register is configured as shown in Fig. 2. Bit sequences being measured are summed in modulo 2 with the register feedback. The register is clocked by the same clock as the bit stream under measurement. Input sequences may be any length, but at the end of the measurement only the residue remaining in the register is looked at. These 16 bits, when displayed in a hexadecimal format, comprise the "signature" of the measured bit stream. A nonstandard hexadecimal character set (0123456789ACFHPU) was chosen for easy readability and compatibility with 7-segment displays.

The table in Fig. 2 shows how the register generates a signature from the 20-bit sequence 11111100000-111111111. Initially (time 0 thru 7) the register acts merely as a shift register. At time 7, the first 1 of the input sequence has reached the first feedback tap (tap 1, Fig. 2). It is fed back and mixed with the input 0, with the result that a 1, not a 0, is next clocked into the register (time 8). This behavior continues until the end of the measurement when a residue of 16 bits, 1101100101010011 (time 20), is all that is left from the 20-bit input sequence. (Note the total dissimilarity in appearance between this residue and the original 11111100000111111111 input sequence.) This residue is displayed in hexadecimal format as H953, the signature of the 20-bit sequence.

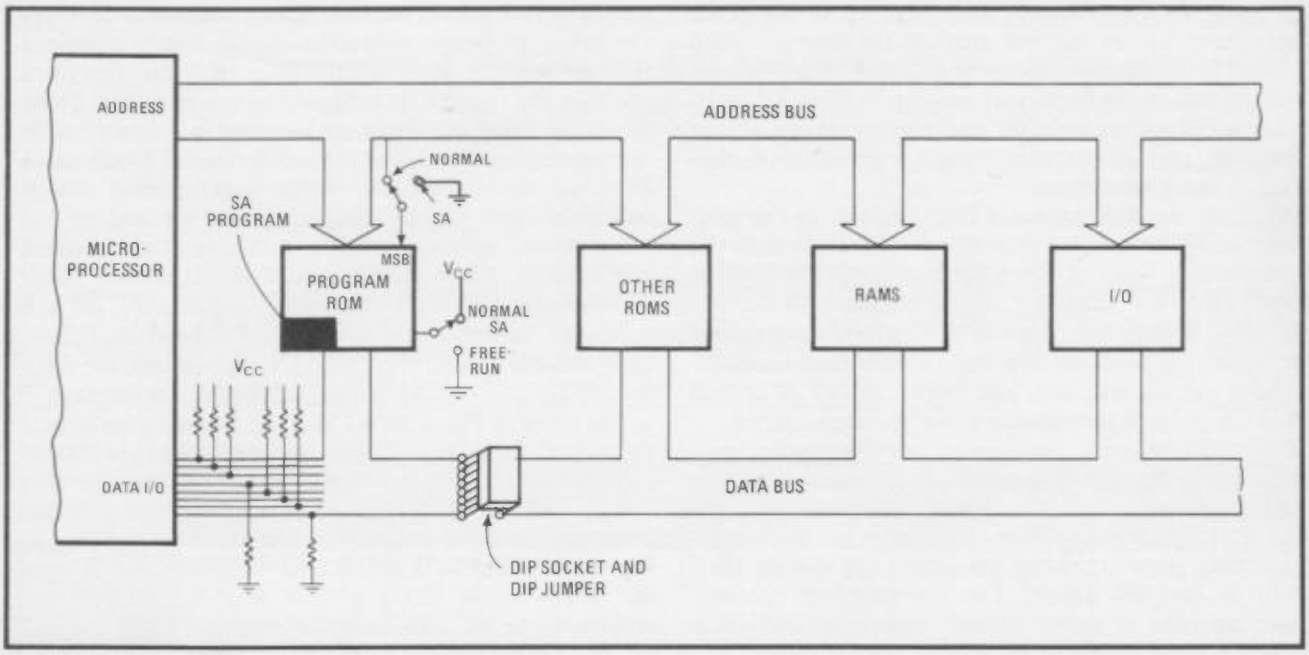
Designing in signature analysis

In an actual circuit designed to incorporate signature analysis, the bit sequences may be any length, probably much longer than 20 bits. A portion of the circuit's read-only memory—perhaps 5%—contains a special program for stimulating the various nodes in the circuit (Fig. 3). The stimulation serves more to "wiggle" or force a state change on the nodes than to generate meaningful data. Frequently this stimulus program may be merged with the product's performance-verification program. The correct signatures are developed by simply exercising the various parts of a circuit that is known to be good and noting the results on the circuit diagram.

A second requirement, beyond stimulating nodes, is to break feedback paths within the circuit either by using hardware switches, jumpers, or connectors, or by disabling gates with software. This requirement is necessary to prevent a fault from being fed back around and perturbing all data nodes. In practice, adding the ability to break feedback paths incurs an incremental hardware expense of less than 1%. (The cost is more than offset, however, by the savings resulting from no longer needing to subdivide the product into small, replaceable modules.) When these two requirements are met, back-tracing a fault to its source is a straightforward process of tracing faulty signatures.

In using signature analysis, the procedure varies slightly depending on whether the fault lies in the kernel (the minimum configuration of microprocessor and ROM necessary to run the simplest test program) or in the outlying circuitry.

If the fault is in the outlying circuitry (a keyboard or



3. Adapting the product. To incorporate signature analysis in a microprocessor-based product, the designer has to add extra program steps to the read-only memory, some switches to put the system into the signature-analysis mode, resistors to force a no-operation instruction when the ROM is disabled, and jumpers that can be removed to isolate portions of the circuit. Here, a 16-pin jumper in a dual in-line package (type AMP 435704-8 or equivalent) is inserted in the data bus.

display scanner, input/output latch, etc.), the field engineer simply switches the circuit to the diagnostic mode. Then, guided by a troubleshooting tree, he uses the test instrument to trace faults back to their source.

But what if the problem lies in the kernel, and even the ROM stimulus program will not run? Here, the microprocessor itself can provide a stimulus if its address counter is allowed to sequence through the address field. To do this it is only necessary to open the data/instruction bus and force the no-operation instruction onto it. This stimulus program checks out all the address lines and the individually enabled ROMs as well. All of these nodes are readily characterized with signatures.

Since signature analysis relies on the ability of a system to control itself in a synchronous manner, asynchronous circuits, like monostables, direct-memory access, dynamic memory, or interrupts, need to be carefully controlled. Generally, simple provisions in the hardware can be made to force them into a synchronous or disabled condition when that is required for a particular test.

The technique in use

As an example, consider the first HP instrument to use signature analysis—the 3455A system voltmeter from the HP Loveland Instrument division in Colorado. The digital portion is quite extensive. It is microprocessor-controlled and contains a self-test program stored in ROM. If the self-test fails, a jumper inside the enclosure is removed, breaking feedback loops and also enabling the signature-analysis routine which is used now to diagnose the instrument.

The decision to go with signature analysis influenced the design in several ways, all of which make it easier to troubleshoot down to the component level. The whole digital portion is on one board. The elimination of connectors and a multitude of smaller pc subassemblies reduced the production cost and also made all the parts easily accessible for testing without the use of special extender boards.

Naturally, some extra design time, a few more ROM locations, and the extra jumper wire were the price paid for this kind of serviceability. The cost evaluation proved to be interesting: the production cost actually fell, and the extra design time amounted to approximately 1% of the overall development time.

Manual aid

Besides the design engineer, the writer of the service manual made an important contribution to the successful application of the signature analysis to the 3455A voltmeter. After learning the internal algorithms of the product almost as well as the designer and having no precedent to fall back upon, he developed a number of innovative ideas for the service approach that were enthusiastically received by the field engineers.

The service manual is written in such a way that a person unfamiliar with the signature analyzer can walk up to a sick voltmeter, read the instructions, and within a short time locate the fault. One element in the manual is a troubleshooting flowchart or tree (Fig. 4), which systematically guides the technician through the fault-finding process.

The initial tests may rely very little on signature analysis, yet may allow isolation of the fault down to a specific area. Diagnostic programs cannot carry on from here, since they do not have access to individual nodes, but it is from here on down to the components that signature analysis excels.

At this level, the repairman uses the annotated sche-

The signature analyzer

The 5004A signature analyzer checks out a compatibly designed digital product by detecting the bit streams at various circuit nodes and displaying them as hexadecimal signatures, which may or may not agree with the correct values noted on the schematic. It is a lightweight, portable instrument, built around the feedback-shift-register circuitry that produces the signatures, and it is equipped with an active probe for data input.

The probe has dual threshold levels that are compatible with transistor-transistor logic. It also serves as a TTL probe, rather like the HP 545A, and in this capacity provides additional troubleshooting information by indicating high, low, bad-level, and pulsing states.

Through an active "pod" on the 5004A's input cable, the product under test supplies the instrument with three gating signals: start, stop, and clock. Start signals the beginning of a measurement window, preparing the shift register to receive information from the data probe. Stop closes the measurement window. Clock is the system clock of the product under test and assures synchronous acceptance of data and gating signals by the shift register. The active edge of each of these gating signals can be selected at the front panel, giving the designer more latitude in applying signature analysis to his product without adding hardware.

The front panel also includes a gate light and an unstable-signature light. The gate light indicates proper start/stop gating operation, remaining on during the measurement window, with stretching to make it visible to the operator during very short on times. The unstable-signature light indicates a difference between signatures in adjacent windows, alerting the user to intermittent faults that may not be apparent from the display.

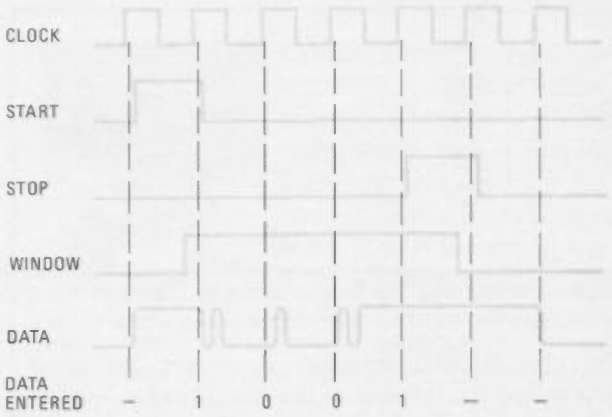
Two useful controls are the hold and reset switches. The hold feature allows observation of single-event (one-shot) signatures, such as a power-on-restart routine. The instrument will display only the signature associated with the first valid window and will hold the display until the probe reset switch is pressed. The hold/reset controls are also useful for taking signatures in awkward locations where it is impossible to simultaneously probe and watch the display.

Since the 5004A is synchronized to the system clock of the product under test (up to 10 megahertz), setup times become important. Setup time is the period during which data must be stable before arrival of the selected clock edge. In the 5004A, the maximum data setup time is 15 nanoseconds (but typically 8 ns). This leaves the balance of the clock cycle for the settling times of components in the product under test. No hold time is required after the selected clock edge.

Any data state changes between clock edges are disregarded. The first data bit accepted during a measurement window is the one coinciding with the synchronized start-signal edge. The last bit accepted is that preceding the synchronized stop-signal edge (see figure).

Tri-state bus architectures are common in many types of equipment and pose the problem of how to interpret the floating state for the purposes of consistent signature detection. Pullup resistors in the circuit under test would force a bus high in the third state, but would slow down the state transitions and possibly cause inconsistent signatures. Instead, the 5004A data probe pulls to the 1.4-volt level, through a 50-kilohm input resistor, and employs hysteresis. This causes the floating state of a tri-state bus to be entered without ambiguity into the signature as the same state (0 or 1) as the most recent valid bit.

To increase the confidence of on-site service, the front panel self-test feature allows a go/no-go checkout of the entire 5004A, including probe, pod, and cables. An internal program exercises the analyzer and displays the result. If this display indicates a malfunction, the 5004A itself can be switched to its own test mode and diagnosed to the component level with another signature analyzer.

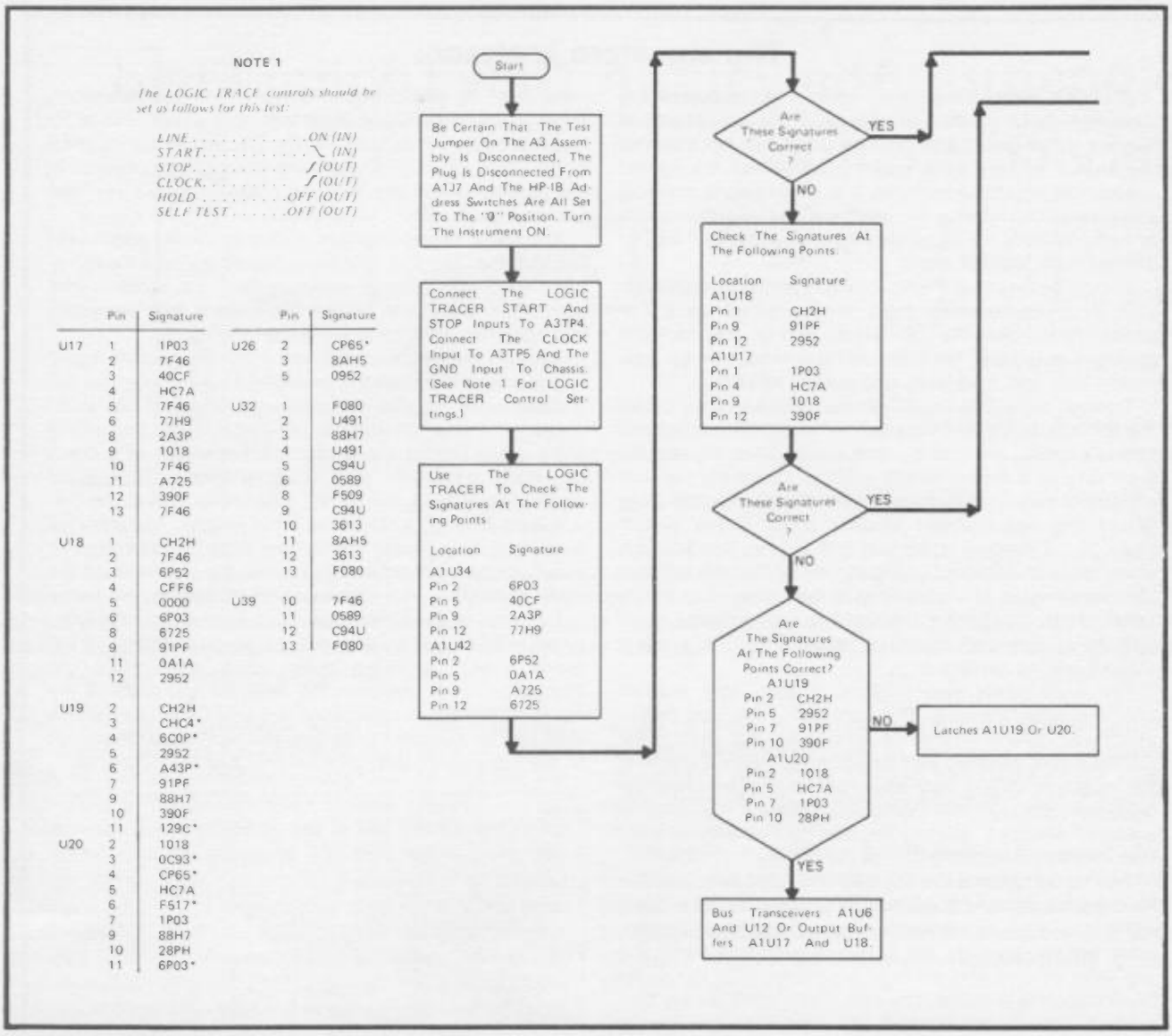


matics and graphs of board layouts, together with the flow chart, to find the bad node. In some cases, the manual includes instructions as to which IC to replace. In other cases the use of a logic probe, which is an integral part of the logic tracer, may be required. A current sensor, such as the HP 547A, helps to find short circuits between traces or to ground and is particularly helpful when bus problems are encountered.

In the case of the voltmeter, the first test checks its kernel, which consists of the microprocessor, the clock circuit, the power supply, and a number of external gates. After the proper functioning of the kernel has been verified, the test setup is changed (one control wire of the logic tracer is moved to another pin in the 3455A), and the remaining portions of the circuit are tested. A special portion of the ROM control loads and reads the random-access memories. Some asynchronous portions require a third test setup. Again, the connection of the start wire is simply moved to the next pin designated for this purpose, and troubleshooting can continue.

Several methods of documentation have been tried successfully. The 3455A service manual shows pictures of the board and defines the setup for each test (Fig. 4). Each picture shows only the signatures related to the particular test, directing the field engineer's effort towards the important areas on the board. The ROM program even simulates interrupt signals, ensuring, however, that they occur predictably at the same spot within a window so that stable signatures result.

The signature analyzer has its own self-check. Each test setup is tested by touching the power-supply voltage with the instrument's probe to input a sequence of all 1s.



	Pin	Signature		Pin	Signature
U17	1	1P03	U26	2	CP65*
	2	7F46		3	8AH5
	3	40CF		5	0952
	4	HC7A			
	5	7F46	U32	1	F080
	6	77H9		2	U491
	8	2A3P		3	88H7
	9	1018		4	U491
	10	7F46		5	C94U
	11	A725		6	0589
	12	390F		8	F509
	13	7F46		9	C94U
				10	3613
U18	1	CH2H		11	8AH5
	2	7F46		12	3613
	3	6P52		13	F080
	4	CFF6			
	5	0000	U39	10	7F46
	6	6P03		11	0589
	8	6725		12	C94U
	9	91PF		13	F080
	11	0A1A			
	12	2952			
U19	2	CH2H			
	3	CHC4*			
	4	6C0P*			
	5	2952			
	6	A43P*			
	7	91PF			
	9	88H7			
	10	390F			
	11	129C*			
U20	2	1018			
	3	0C93*			
	4	CP65*			
	5	HC7A			
	6	F517*			
	7	1P03			
	9	88H7			
	10	28PH			
	11	6P03*			

4. Troubleshooting tree. The service manual for the HP 3455A digital voltmeter, the first instrument to use signature analysis, contains a troubleshooting tree and a list of signatures that should be found at the designated pin numbers of the various devices.

If this characteristic signature is correct, the setup conditions and the framing of the measurement window are verified. Specifically, this tells the user that the switches on the signature analyzer, as well as all the jumpers, switches, and control buttons in front and rear of the voltmeter are correctly set. Thus, the confidence level of the user is very high at the start of a test.

This application of signature analysis went particularly smoothly, because the engineer developing service techniques worked closely with the design engineer. Generally, the design engineer wrote the stimulus routines while the service engineer involved himself with documentation and overall test strategy. This early involvement also ensured that the designer, with the many demands on his time, did not neglect to think about serviceability and, for example, put off allocating read-only-memory space till inconveniently late in the design cycle.

The fact that signature analysis is built into the 3455A voltmeter also made final testing on the production line much easier. A signature analyzer is now a favorite piece of production-line test equipment for the 3455A.

Thus, the signature analyzer promises to have a significant impact on present service costs. With the industry presently spending roughly a billion dollars annually to find the 10 million ICs that fail each year in the field, such a technique is needed. Although IC pretesting and burn-in programs have gone a long way toward weeding out weak devices, changes are nevertheless needed in service strategy as well. The signature analyzer offers a new option for those who are planning service strategies. □

Signature Analysis: A New Digital Field Service Method

In a digital instrument designed for troubleshooting by signature analysis, this method can find the components responsible for well over 99% of all failures, even intermittent ones, without removing circuit boards from the instrument.

by Robert A. Frohwerk

WITH THE ADVENT OF MICROPROCESSORS and highly complex LSI (large-scale integrated) circuits, the engineer troubleshooting digital systems finds himself dealing more with long digital data patterns than with waveforms. As packaging density increases and the use of more LSI circuits leaves fewer test points available, the data streams at the available test points can become very complex. The problem is how to apply some suitable stimulus to the circuit and analyze the resulting data patterns to locate the faulty component so that it can be replaced and the circuit board returned to service.

The search for an optimal troubleshooting algorithm to find failing components on digital circuit boards has taken many directions, but all of the approaches tried have had at least one shortcoming. Some simply do not test a realistic set of input conditions, while others perform well at detecting logical errors and stuck nodes but fail to detect timing-related problems. Test systems capable of detecting one-half to two-thirds of all possible errors occurring in a circuit have been considered quite good. These systems tend to be large, for factory-based use only, and computer-driven, requiring program support and software packets and hardware interfaces for each type of board to be tested. Field troubleshooting, beyond the logic-probe capability to detect stuck nodes, has been virtually neglected in favor of board exchange programs.

The problem seems to be that test systems have too often been an afterthought. The instrument designer leaves the test procedure to a production test engineer, who seeks a general-purpose solution because he lacks the time to handle each case individually.

Obviously it would be better if the instrument designer provided for field troubleshooting in his original design. Who knows a circuit better than its original designer? Who has the greatest insight as to how to test it? And what better time to modify a circuit to accommodate easy testing than before the circuit is in production?

New Tools Needed

But here another problem arises: what do we offer the circuit designer for tools? A truly portable test instrument, since field troubleshooting is our goal, would be a passive device that merely looked at a circuit and told us why it was failing. The tool would provide no stimulus, require little software support, and have accuracy at least as great as that of computer-driven factory-based test systems.

Cover: *Those strange-looking strings of four alphanumeric characters on the instrument's display and the schematic diagram are signatures, and the instrument is the 5004A Signature Analyzer, a troubleshooting tool for field repair of digital systems. With a failing system operating in a self-stimulating test mode, the service person probes various test points, looking for incorrect signature displays that can point to faulty components.*

If a tester provides no stimulus, then the circuit under test must be self-stimulating. Whereas this seemed either impossible or at best very expensive in the past, a self-stimulating circuit is not out of the question now. More and more designs are microprocessor-oriented or ROM-driven, so self-stimulus, in the form of read-only memory, is readily available and relatively inexpensive.

By forcing a limitation on software, we have eliminated the capability to stop on the first failure and must use a burst-mode test. Another restriction we will impose is that the device under test must be synchronous, in the sense that at the time the selected clock signal occurs the data is valid; not an unfair condition by any means, and it will be justified in the article beginning on page 15.

There are only a few known methods for compressing the data for a multiple-bit burst into a form that can be handled easily by a portable tester without an undue amount of software. One method used in large systems is transition counting. Another method, a much more efficient data compression technique borrowed from the telecommunications field, is the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator.

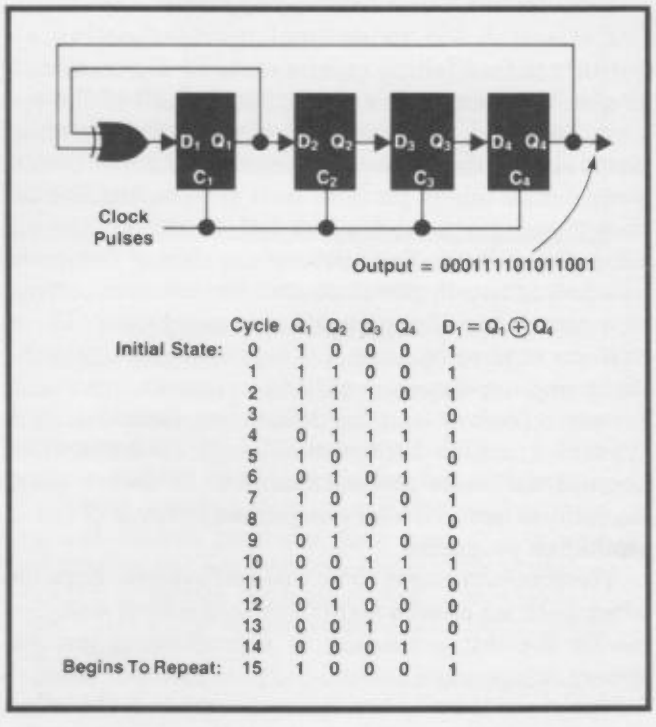
A troubleshooting method and a portable instrument based on this concept turns out to be the answer we are seeking. We call the method signature analysis and the instrument the 5004A Signature Analyzer. The instrument is described in the article on page 9. Here we will present the theory of the method and show that it works, and works very well.

Pseudorandom Binary Sequences

A pseudorandom binary sequence is, as implied, a pattern of binary ones and zeros that appears to be random. However, after some sequence length the pattern repeats. The random-like selection of bits provides nearly ideal statistical characteristics, yet the sequences are usable because of their predictability. A PRBS based upon an n-bit generator may have any length up to 2^n-1 bits before repeating. A generator that repeats after exactly 2^n-1 bits is termed maximal length. Such a generator will produce all possible n-bit sequences, excluding a string of n zeros. As an example, let us take the sequence: 000111101011001. This is a fifteen-bit pattern produced by a four-bit maximal-length generator ($15=2^4-1$). If we were to wrap this sequence around on itself, we would notice that all possible non-zero four-bit patterns occur once and only once, and then the sequence repeats.

To construct a PRBS generator we look to the realm of linear sequential circuits, which is where the simplest generators reside mathematically. Here there exist only two types of operating elements. The first is a modulo-2 adder, also known as an exclusive-OR gate. The other element is a simple D-type flip-flop, which being a memory element behaves merely as a time delay of one clock period. By connecting flip-flops in series we construct a shift-register as in Fig. 1, and by taking the outputs of various flip-flops, exclusive-ORing them, and feeding the result back to the register input, we make it a feedback shift register that will produce a pseudorandom sequence. With properly chosen feedback taps, the sequence will be maximal length. The fifteen-bit sequence above was produced by the generator in Fig. 1, with the flip-flops initially in the 0001 state since the all-zero state is disallowed. The table in Fig. 1 shows the sequence in detail. The list contains each of the sixteen ways of arranging four bits, except four zeros.

If we take the same feedback shift register and provide it with an external input, as in Fig. 2, we can overlay data onto the pseudorandom sequence. The overlaid data disturbs the internal sequence of the generator. If we begin with an initial state of all zeros and supply a data impulse of 1000..., the result is the same sequence as in Fig. 1 delayed by one clock period.



	Cycle	Q_1	Q_2	Q_3	Q_4	$D_1 = Q_1 \oplus Q_4$
Initial State:	0	1	0	0	0	1
	1	1	1	0	0	1
	2	1	1	1	0	1
	3	1	1	1	1	0
	4	0	1	1	1	1
	5	1	0	1	1	0
	6	0	1	0	1	1
	7	1	0	1	0	1
	8	1	1	0	1	0
	9	0	1	1	0	0
	10	0	0	1	1	1
	11	1	0	0	1	0
	12	0	1	0	0	0
	13	0	0	1	0	0
	14	0	0	0	1	1
Begins To Repeat:	15	1	0	0	0	1

Fig. 1. *Signature analysis is a troubleshooting technique that makes use of the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator. Shown here is a feedback shift register that generates a 15-bit PRBS. The outputs of the four flip-flops go through all possible non-zero four-bit patterns and then the sequence repeats.*

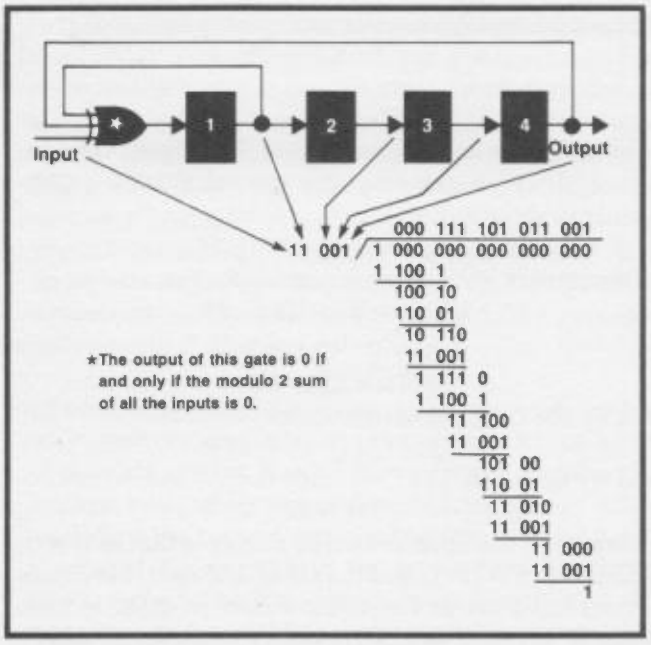


Fig. 2. *When the feedback shift register of Fig. 1 is provided with an external input, data can be overlaid on the PRBS generated by the circuit. Feeding data into a PRBS generator is the same as dividing the data by the characteristic polynomial of the generator.*

Shift Register Mathematics

A shift register may be described using a transform operator, D, defined such that $X(t) = DX(t-1)$. Multiplying by D is equivalent to delaying data by one unit of time. (Recall that we are concerned only about synchronous logic circuits.) In Fig. 2 the data entering the register is the sum of samples taken after one clock period and four clock periods along with the input data itself. Thus, the feedback equation may be written as $D^4X(t) + DX(t) + X(t)$ or simply X^4+X+1.

It happens that feeding a data stream into a PRBS generator is equivalent to dividing the data stream by the characteristic polynomial of the generator. For the particular implementation of the feedback shift register considered here the characteristic polynomial is X^4+X^3+1, which is the reverse of the feedback equation. Fig. 2 shows the register along with longhand division of the impulse data stream (100...). Keep in mind that in modulo-2 arithmetic, addition and subtraction are the same and there is no carry. It can be seen that the quotient is identical to the pattern in Fig. 1 and repeats after fifteen bits (the "1" in the remainder starts the sequence again).

Because the shift register with exclusive-OR feedback is a linear sequential circuit it gives the same weight to each input bit. A nonlinear circuit, on the other hand, would contain such combinatorial devices as AND gates, which are not modulo-2 operators and which would cancel some inputs based upon prior bits. In other words a linear polynomial is one for which $P(X+Y) = P(X) + P(Y)$. Take the example of Fig. 3, where the three different bit streams X, Y, and X+Y are fed to the same PRBS generator. Notice that the output sequences follow the above relationship, that is, $Q(X+Y) = Q(X) + Q(Y)$. Also, notice that Y is a single impulse bit delayed in time with respect to the other sequences and the only difference between X and X+Y is that single bit. Yet, Q(X+Y) looks nothing like Q(X). Indeed, if we stop after entering only twenty bits of the sequences and compare the remainders, or the residues in the shift register, they would be: $R(X+Y) = 0100$, $R(X) = 0111$.

Error Detection by PRBS Generator

Looking at this example in another manner, we can think of X as a valid input data stream and X + Y as an erroneous input with Y being the error sequence. We will prove later that any single-bit error, regardless of when it occurs, will always be detected by stopping the register at any time and comparing the remainder bits (four in this case) with what they should be. This error detection capability is independent of the length of the input sequence. In the example of Fig. 3, R(X+Y) differs from the correct R(X), and the effect of the error remains even though the error has disappeared many clock periods ago.

Let us stop for a moment to recall our original goal. We are searching for a simple data compression algorithm that would be efficient enough to be usable in a field service instrument tester. As such it was to require only minimal hardware and software support.

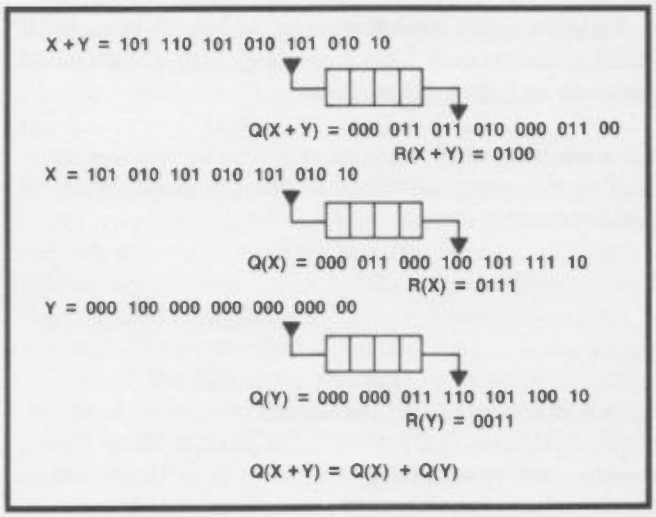


Fig. 3. *Three different input data sequences fed to the same PRBS generator produce very different output sequences even though the input sequences differ by only one bit. If the generators are stopped at some time and the patterns remaining in the flip-flops are compared, they are also different. These remainder patterns are called* signatures. *They show the effects of an error sequence Y added to a data stream X even when the error occurs only once in a long measurement window.*

We have now found such an algorithm. If the circuit designer arranges his synchronous circuit so as to provide clock and gate signals that produce a repeatable cycle for testing, then the feedback shift register is the passive device that we need to accumulate the data from a node in the instrument under test. By tracing through an instrument known to be good, the designer merely annotates his schematic, labeling each test point with the contents of the shift register at the end of the measurement cycle, and uses this information later to analyze a failing circuit. Because this PRBS residue depends on every bit that has entered the generator, it is an identifying characteristic of the data stream. We have chosen to call it a *signature*. The process of annotating schematics with good signatures as an aid in troubleshooting circuits that produce bad signatures has been termed *signature analysis*.

Errors Detected by Signature Analysis

We have claimed that any single-bit error will always be detected by a PRBS generator. But how about multiple errors? Also, our goal was to maintain error detection capability at least as good as existing methods. Earlier mention was made of transition counting, which appears to be the only other method that could easily be made portable. To show how signature analyis stands up against transition counting requires a mathematical discussion of the error detection capabilities of these methods. Take first the PRBS.

Assume X is a data stream of m bits, P is an n-bit PRBS generator, P^{-1} its inverse ($P^{-1}P = 1$), Q is a quotient and R the remainder.

$$P(X) = Q(X)\cdot 2^n + R(X). \quad (1)$$

Take another m-bit sequence Y that is not the same as X and must therefore differ by another m-bit error sequence E such that

$$Y = X + E.$$

Now,

$$P(Y) = Q(Y)\cdot 2^n + R(Y)$$

so,

$$P(X+E) = Q(X+E)\cdot 2^n + R(X+E).$$

But all operators here are linear, so

$$P(X) + P(E) = Q(X)\cdot 2^n + Q(E)\cdot 2^n + R(X) + R(E).$$

Subtracting (or adding, modulo 2) with equation 1 above,

$$P(E) = Q(E)\cdot 2^n + R(E). \quad (2)$$

However, if Y is to contain undetectable errors,

$$R(Y) = R(X).$$

It follows that

$$R(Y) = R(X+E) = R(X) + R(E) = R(X),$$

$$R(E) = 0.$$

Substituting into equation 2,

$$P(E) = Q(E)\cdot 2^n,$$

and all undetectable errors are found by

$$E = P^{-1}Q(E)\cdot 2^n. \quad (3)$$

For a single-bit error

$$E = D^a(1)$$

where D is the delay operator, a is the period of the delay, and "1" is the impulse sequence 1000... Substituting into (3),

$$D^a(1) = P^{-1}Q(D^a(1))\cdot 2^n.$$

D commutes with other linear operators, so

$$D^a(1) = D^aP^{-1}Q(1)\cdot 2^n$$

$$1 = P^{-1}Q(1)\cdot 2^n$$

$$P(1) = Q(1)\cdot 2^n.$$

But by the original assumptions,

$$P(1) = Q(1)\cdot 2^n + R(1)$$

and by addition

$$R(1) = 0.$$

However, it has been shown by example that $R(1) \neq 0$. Therefore, $E \neq D^a(1)$ and the set of undetectable errors E does not include single-bit errors; in other words, a single-bit error is always detectable. (An intuitive argument might conclude that a single-bit error would always be detected because there would never be another error bit to cancel the feedback.)

To examine all undetectable errors as defined by equation 3, it helps to consider a diagrammatical representation, Fig. 4, of:

$$E = P^{-1}Q(E)\cdot 2^n.$$

Since X, Y, and E are all m-bit sequences, it follows that $Q\cdot 2^n$ must be an m-bit sequence containing n final zeros. Q therefore contains (m−n) bits. Hence, there are 2^{m-n} sequences that map into the same residue as the correct sequence, and there are $2^{m-n}-1$ error sequences that are undetectable because they leave the same residue as the correct sequence. 2^m sequences can be generated using m bits and only one of these is correct, so the probability of failing to detect an error by a PRBS is

$$\text{Prob (PRBS, fail)} = \frac{\text{Undetectable Errors}}{\text{Total Errors}} = \frac{2^{m-n}-1}{2^m-1}.$$

For long sequences, large m,

$$\text{Prob (PRBS, fail)} \approx 1/2^n.$$

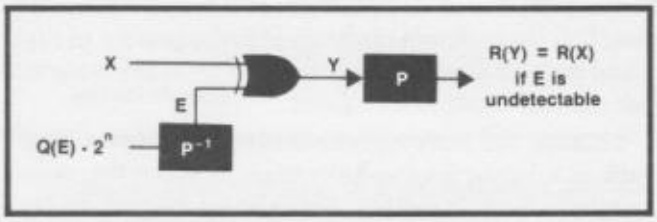


Fig. 4. *A diagrammatical representation of errors undetectable by signature analysis. For long data sequences the probability of not detecting an error approaches $1/2^n$, where n is the number of flip-flops in the feedback shift register.*

In summary, a feedback shift register of length n will detect all errors in data streams of n or fewer bits, because the entire sequence will remain in the register, R(X) = P(X). For data streams of greater than n bits in length, the probability of detecting an error using a PRBS is very near certainty even for generators of modest length. The errors not detected are predictable and can be generated by taking all m-bit sequences with n trailing zeros and acting upon such sequences by the inverse of the n-bit PRBS generator polynomial P, that is

$$E = P^{-1}(Q \cdot 2^n).$$

Furthermore, such error detection methods will always detect a single-bit error regardless of the length of the data stream. It can also be proved that the only undetectable error sequence containing two errors such that the second cancels the effect of the first is produced by separating the two errors by exactly 2^n-1 zeros.[1] The one sequence of length n+1 that contains undetectable errors begins with an error and then contains other errors that cancel each time the original error is fed back.

Errors Detected by Transition Counting

It appears that signature analysis using a PRBS generator is a difficult act to follow, but let us give transition counting a chance. A transition counter assumes an initial state of zero and increments at each clock time for which the present data bit differs from the previous bit. With a transition counter the probability of an undetected error, given that there is some error, is:

$$\text{Prob (Trnsn, Fail)} = N_u/N_t,$$

where N_u = number of undetected errors and N_t = total number of errors. But

$$N_u = \sum_{r=0}^{m} p_{ur}$$

where p_{ur} = Prob (undetected errors given r transitions). However,

$$p_{ur} = N_{ur} \cdot p_r$$

where N_{ur} = number of undetected errors given r transitions, and p_r = Prob (counting r transitions). Reducing further,

$$N_{ur} = N_r - N_c,$$
$$p_r = N_r/N_s,$$

where N_c = number of ways of counting correctly (=1), N_s = total number of m-bit sequences, and N_r = number of ways of counting r transitions:

$$N_r = \binom{m}{r} = \frac{m!}{r!(m-r)!}$$

The binomial coefficient $\binom{m}{r}$ expresses the number of ways of selecting from m things r at a time. Looking back to the original denominator,

$$N_t = N_s - N_c.$$

Putting all of this together,

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} (N_r - N_c)(N_r/N_s)}{N_s - N_c}.$$

Or,

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} \left[\binom{m}{r} - 1\right] \binom{m}{r}/2^m}{2^m - 1}$$
$$\approx 1/\sqrt{m\pi}.$$

This is the probability of a transition counter's failing to detect an error in an m-bit sequence.

A similar argument finds the probability of the specific case where a single-bit error is not detected by a transition counter. There are 2^m sequences of m bits and any one of the m bits can be altered to produce a single-bit error, so that there are $m \cdot 2^m$ possible single-bit errors. To determine how many undetected single-bit errors exist, we must look at how to generate them.

Upon considering the various ways of generating single-bit errors that are undetectable, a few observations become obvious. We can never alter the final bit of a sequence, because that would change the transition count by plus or minus one, which would be detected. The only time we can alter a bit without getting caught is when a transition is adjacent to a double bit; that is, flipping the center bit in the patterns 001, 011, 100, or 110 will not affect the transition count. In other words, the transition count for ...0X1... and ...1X0... does not depend on the value of X.

Since our transition counter assumes an initial 0 state, the first bit of the sequence, regardless of its state, can be flipped without affecting the transition count, provided that the second bit is a one. In this case only the second of m bits is predetermined, i.e., $b_2 = 1$, and there are 2^{m-1} ways of completing the sequence. Any bit other than the first or last, that is, the m−2 bits from b_2 through b_{m-1}, can be altered without affecting the transition count if the bit in question is flanked by a zero on one side and a one on the other. For a given bit b_i we have free choice of m−1 bits, since as soon as we select b_{i-1} then b_{i+1} is forced to the opposite state. There are $(m-2) \cdot 2^{m-1}$ of these midstream errors. Adding the 2^{m-1} sequences where b_1 can be changed we have a total of $(m-1) \cdot 2^{m-1}$ sequences containing single-bit errors that cannot be detected by a transition counter. But earlier we showed that the total number of single-bit errors was $m \cdot 2^m$, hence the probability of failing to detect a single-bit error is

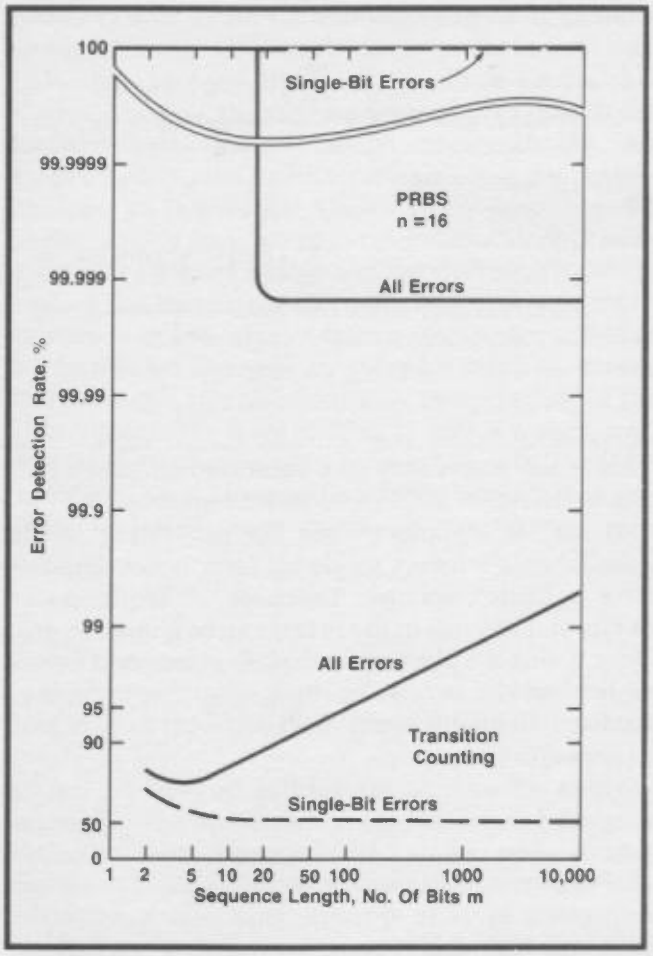


Fig. 5. *Probability of detecting errors for signature analysis and transition counting as a function of the length of the data sequence. n=16 for the PRBS generator.*

$$\text{Prob (Trnsn, Fail, single-bit)} = \frac{(m-1)\cdot 2^{m-1}}{m\cdot 2^m} = \frac{m-1}{2m}$$

$$\approx 1/2.$$

It may be noted that the failure rate is actually somewhat higher, because a counter of limited length will overflow for long sequences, leaving some ambiguity. It can be shown that because of this overflow an n-bit transition counter will never detect more than $1/2^n$ of all errors.

Signature Analysis versus Transition Counting

We can now plot the probabilities of detecting any error using a transition counter versus a PRBS generator (see Fig. 5). It is interesting to note that the transition count method looks worst on single-bit errors, exactly where the feedback shift register never fails. Overall the transition counter looks pretty good, detecting at least half of all errors, but even a one-bit shift register could do that. The four-bit PRBS generator used in earlier examples will always detect better than $(100-100/2^4)=93\%$ of all errors. It seems conclusive that the PRBS method puts on a good performance, and if we want it to do better we merely add one more bit to the register to halve the rate of misses.

How Close Do We Want to Get?

We set out to find a means of instrument testing at least as good as present computer-based methods. These existing systems generally perform as well as the engineer who adapts them to the circuit under test. The task of adapting a circuit to be tested by signature analysis is very much the same as adapting to any other tester—engineering errors are assumed constant. If the PRBS technique is used for backtracing to find faulty components in field service, then the largest remaining block of human error is the ability of the service person to recognize a faulty signature.

It seems that a four-character signature is easily recognized, while the incidence of correct pattern recognition falls off with the addition of a fifth character. We tried this on a statistically small sample of people and found it to be so. Electronically, four hexadecimal characters is sixteen bits. A few bits more or less is not likely to complicate a shift register, but it would have an adverse effect on the user. Sixteen bits gives a detector failure rate of less than sixteen parts per million (one in 65,535), adequate for most purposes, so we settled on a four-character signature.

Since the signature offers no diagnostic information

Last In → A	B	C	D ← First In	Display
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9
0	1	0	1	A
1	1	0	1	C
0	0	1	1	F
1	0	1	1	H
0	1	1	1	P
1	1	1	1	U

Fig. 6. *In the HP 5004A Signature Analyzer, n=16 and the remainder, or signature, is displayed as four non-standard hexadecimal characters. Each character represents the outputs of a group of four flip-flops as shown here.*

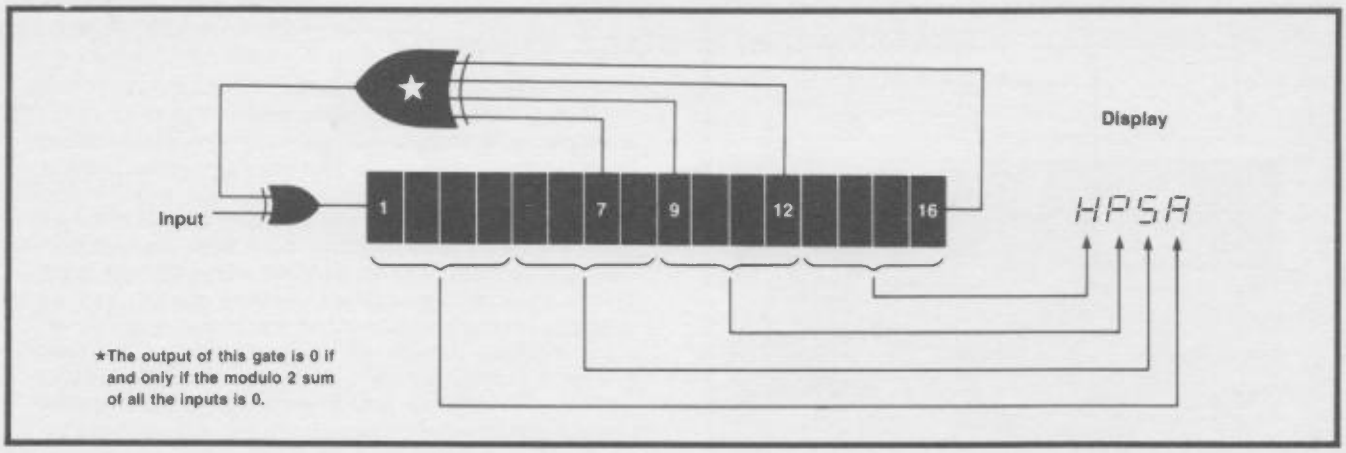


Fig. 7. *The 16-flip-flop PRBS generator used in the 5004A Signature Analyzer.*

whatsoever, but is purely go/no-go, the character set was not restricted, except to be readable. Numbers are quite readable but there are not enough of them. Another consideration was that for an inexpensive tool, seven-segment displays are desirable. The chosen character set (Fig. 6) is easily reproduced by a seven-segment display and the alpha characters are easily distinguishable even when read upside down. A further psychological advantage of this non-standard ("funny hex") character set is that it does not tempt the user to try to translate back to the binary residue in search of diagnostic information.

Register Polynomial

We have decided on a four-character display for a sixteen-bit register, but it remains to select the feedback taps to guarantee a maximal length sequence. It happens that this can be done in any of 2048 ways.[2] The computer industry uses two:

$$\text{CRC-16} = X^{16}+X^{15}+X^2+1,$$

and

$$\text{SDLC(or CCITT-16)} = X^{16}+X^{12}+X^5+1.$$

But each of these is reducible:

$$\text{CRC} = (X+1)(X^{15}+X+1),$$

and

$$\text{SDLC} = (X+1)(X^{15}+X^{14}+X^{13}+X^{12}+X^4+X^3+X^2+X+1).$$

The X+1 factor was included in both to act as a parity check; it means that all undetectable error sequences will have even parity. This is apparent by looking at the original polynomials and noting that they each have an even number of feedback taps, so an even number of error bits is required to cancel an error. For our purposes this clustering of undetectable errors seems undesirable. We would like a polynomial that scatters the missed errors as much as possible. For this reason we would also like to avoid selecting feedback taps that are evenly spaced or four or eight bits apart because the types of instruments, microprocessor-controlled, that we will most frequently be testing tend to repeat patterns at four and eight-bit intervals. The chosen feedback equation is:

$$X^{16}+X^{12}+X^9+X^7+1,$$

which corresponds to the characteristic polynomial

$$P(X) = X^{16}+X^9+X^7+X^4+1.$$

This is an irreducible maximal length generator with taps spaced unevenly (see Fig. 7). Our relatively limited experience with this PRBS generator has shown no problems with regard to the selection of feedback taps. The test of time will tell; even the CRC-16 generator seems to have fallen out of favor with respect to that of SDLC after having served the large-computer industry for well over a decade.

References

1. W.W. Peterson, and E.J. Weldon, Jr., "Error-Correcting Codes," The MIT Press, Cambridge, Massachusetts, 1972.
2. S.W. Golomb, "Shift-Register Sequences," Holden-Day, Inc., San Francisco, 1967.

Robert A. Frohwerk

Bob Frohwerk did the theoretical work on signature analysis. He received his BS degree in engineering from California Institute of Technology in 1970, and joined HP in 1973 with experience as a test engineer for a semiconductor manufacturer and as a test supervisor and quality assurance engineer/manager for an audio tape recorder firm. He's a member of the Audio Engineering Society. Bob was born in Portland, Oregon. Having recently bought a home in Los Altos, California, he and his wife are busy setting up a workshop for Bob's woodworking and metal sculpture, putting in a large organic garden, and planning furniture projects and a solar heating system. Bob also goes in for nature photography, sound systems, and meteorology (he built his wife's weather station).

Signature Analysis in the 5342A

Incorporating microprocessor control into the 5342A Microwave Frequency Counter made it possible to develop a powerful measuring instrument at a substantial reduction in cost. Besides providing many operational benefits, such as keyboard entry of frequency and amplitude offsets, resolution selection, and offset recall, microprocessor control enhances the serviceability of the 5342A by providing powerful diagnostic routines, also selectable from the front-panel keyboard, that aid the service person in fault isolation and instrument verification (see Fig. 1). Other microprocessor routines, exercised every time the instrument is turned on, check the health of ROMs and RAM and display error codes if all is not well.

Despite the diagnostic aids provided by the microprocessor, placing a microcomputer inside a sophisticated measuring instrument also introduces some serviceability problems. After the first prototype was constructed, we discovered it was impossible to isolate certain failures to a particular assembly using traditional troubleshooting equipment and techniques.

Failures involving the microprocessor assembly and the individual assemblies that interface to the microprocessor assembly were extremely difficult to troubleshoot. Even after hours of troubleshooting, it was uncertain whether the fault was a control failure originating on the microprocessor assembly, an interface failure originating on an assembly's interface with the microprocessor, or a failure in some other part of the instrument, causing the measurement algorithm to hang up or branch to an incorrect program segment. We needed a quick way to verify proper operation of the microprocessor control assembly.

Fortunately, there was a solution which, even though the instrument had advanced to the prototype stage, was inexpensive to implement and permitted microprocessor verification and fault isolation to the component level. This technique, called signature analysis, relies on a relatively inexpensive troubleshooting instrument—the 5004A Signature Analyzer.[1]

Signature Analysis

Signature analysis, as implemented in the 5004A Signature Analyzer, employs a data compression technique to reduce long, complex data streams at circuit nodes to four-digit hexadecimal signatures. By taking the signature of a suspected circuit node and comparing it to the correct signature, which is empirically determined and documented in the operating and service manual, proper circuit operation is quickly verified. By probing designated nodes, observing good and bad signatures, and then tracing back along the signal flow from outputs to inputs, the cause of an incorrect signature may be discovered and corrected.

In operation, four signals must be supplied to the signature analyzer. A START signal initiates the measurement window. During this time window, DATA from a circuit node is clocked into the signature analyzer. A CLOCK signal synchronizes the data. A STOP signal terminates the measurement window.

There are two ways to implement signature analysis and meet the requirements just mentioned in a microprocessor-based product: free running and software driven. In the free running method, the microprocessor is forced into an operating mode in which it cycles continuously through its entire address field. START/STOP signals are derived from the address bus lines. In software driven signature analysis, a stimulus program is stored in ROM. The stimulus program generates START/STOP signals and can also write repeatable DATA streams onto the data bus for testing other assemblies connected to the microprocessor. Free running signature analysis has the advantage of not requiring any ROM space for storing the stimulus program. Software driven signature analysis has the advantage of being able to exercise a greater portion of the instrument's circuitry. For thorough testing, both techniques could be implemented in the same instrument.

Fig. 1. *Nine diagnostic modes are available with the counter in AUTO mode. The SET key is pushed twice and is followed by the appropriate digit key.*

SET, SET, 0: Indicates that the main synthesizer is sweeping (SP) and that the signal has been placed in the IF (23) and finally that the harmonic determination has been made (Hd). This display is shown in the photograph.

SET, SET, 1: Displays the main synthesizer frequency, the location of the harmonic comb line (e.g., if −, harmonic is below f_x so must add IF result), and the harmonic number N.

SET, SET, 2: Displays results of counter A accumulation during acquisition.

SET, SET, 3: Displays results of counter B accumulation during acquisition.

SET, SET, 4: Displays intermediate frequency being counted.

SET, SET, 5: If Option 002, amplitude measurement, is installed, a single corrected amplitude measurement is made and held.

SET, SET, 6: If Option 002, amplitude measurement is installed, a continuous measure of uncorrected amplitude is displayed.

SET, SET, 7: When the signal is removed from the microwave port, the main synthesizer sweeps over its full range in 100-kHz steps.

SET, SET, 8: This mode is a keyboard check.

In the 5342A, lack of ROM space ruled out the software driven implementation. To implement the free running approach in the 5342A, all that was required was the addition of some switches and pull-up resistors to the microprocessor assembly. Fig. 9 on page 9 shows a block diagram of the 5342A microprocessor assembly. The shaded area contains the components added to the assembly to implement free running signature analysis.

To check out the microprocessor assembly, the microprocessor is forced into a free run mode by opening the data bus switches S1 (this prevents data out of the ROMs from altering the forced free run instruction) and grounding the free

run switch S2. When S2 is grounded, a clear B instruction is presented to the microprocessor data input (clear B was chosen to minimize the number of diodes needed). This causes the B accumulator to be cleared and the address to be incremented by 1. Consequently, the address lines from the microprocessor repeatedly cycle over the entire address field of the microprocessor from 0000 to FFFF. By using the most significant address line as both START and STOP for the 5004A, and one phase of the microprocessor clock as the 5004A CLOCK input, repeatable, stable signatures are obtained for the microprocessor address lines, ROM outputs, device select outputs, and most circuit nodes on the microprocessor assembly. By checking the assembly's outputs for correct signatures (documented in the manual), it is possible to verify with a high degree of confidence that the assembly is functioning properly. If a signature is incorrect, then signatures are checked back along the signal flow paths, from outputs to inputs. When a device is found where the output signature is bad but the input signatures are good, that device is replaced.

Reference

1. A.Y. Chan, "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits," Hewlett-Packard Journal, May 1977.

-Martin Neil

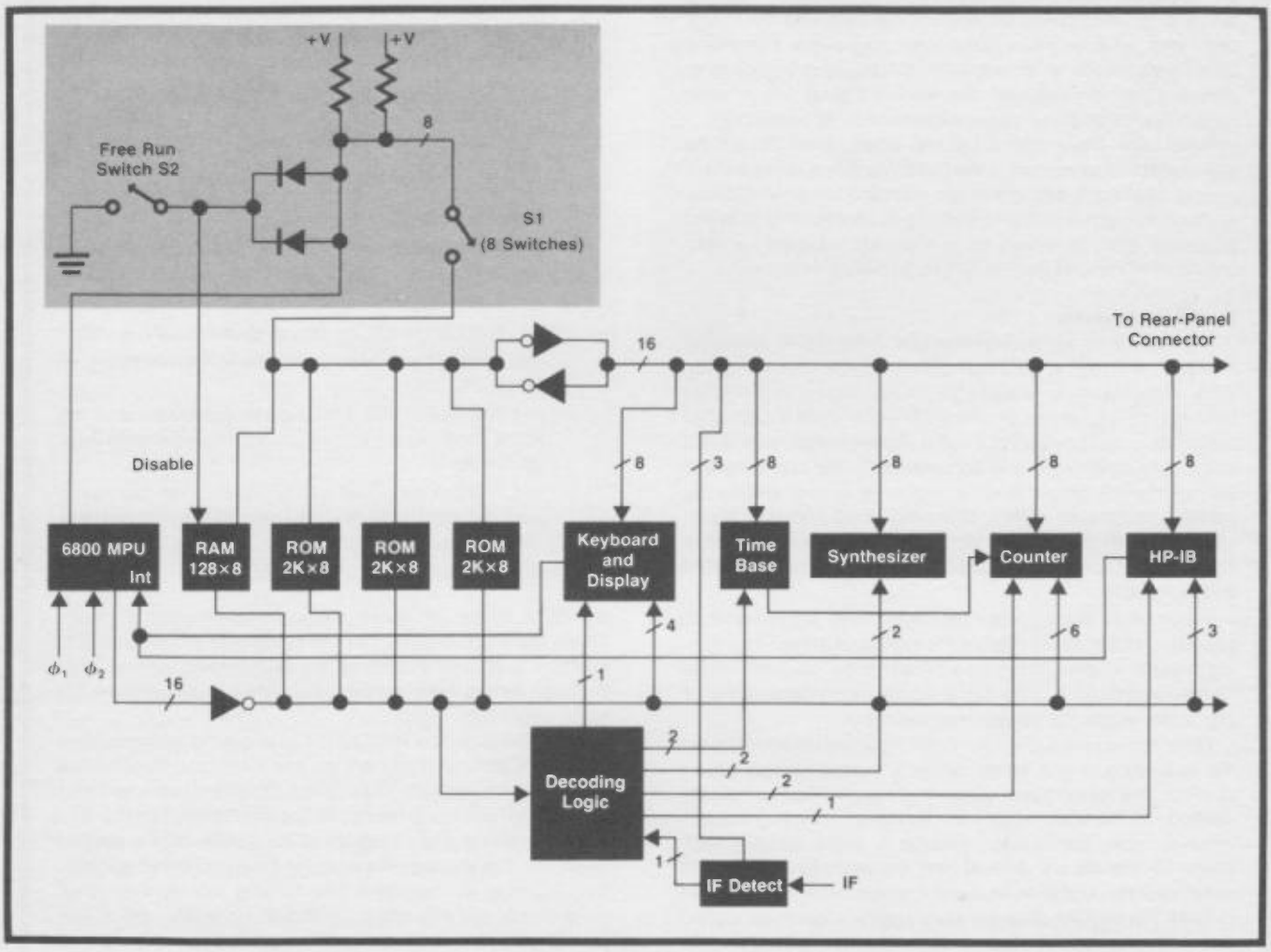


Fig. 9. *5342A microprocessor assembly. Components in the shaded area were added for troubleshooting by signature analysis. Only a few switches and pull-up resistors were required.*

Designing Serviceability into the Model 8568A Spectrum Analyzer

by David D. Sharrit

THE COMPLEXITY OF THE MODEL 8568A Spectrum Analyzer presented several challenges to the serviceability goals set for the instrument. Microcomputer control, the keyboard front panel, the digitally-stored display, and the pilot-signal phase-lock loop are new and very different from previous spectrum analyzers with which production and field repair people are familiar. For this instrument to be

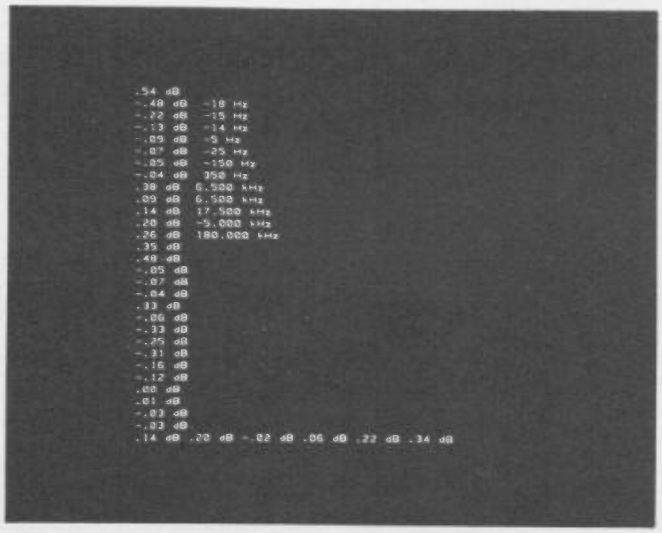


Fig. 1. *Correction factors generated by the error-correction routine can be listed on the CRT to give a check on IF system performance.*

serviced and repaired to the component level in a reasonable amount of time, serviceability had to be designed in, beginning with the first prototype.

Serviceability was implemented by designing self checks into the digital processors, by taking advantage of the processing power of the analyzer to assist in analog troubleshooting, and by designing for signature-analysis[1] troubleshooting of the digital circuitry.

Troubleshooting from the Front Panel

The system was designed so that most faults occurring in the analyzer's frequency-tuning portions are automatically indicated on the CRT display. Each of three phase-lock loops within the instrument has a lock indicator circuit that generates a flag. If any of these flags indicates an unlock condition when the loop should be locked, the appropriate error message is displayed on the CRT, such as 275 UNLOCK, 249 UNLOCK, and YTO UNLOCK. If either of two counter-locked frequencies controlled by the microprocessor cannot be tuned close enough, then either VTO UNCAL or YTO ERROR is displayed. In most cases, these messages make it possible to isolate the fault to two or three internal assemblies.

The instrument's design is such that normal front-panel operations can be used for quick and accurate troubleshooting of analog portions of the instrument. Because exact center frequencies can be keyed in, because markers can be used to read out frequency, frequency difference, amplitude, and amplitude difference, and also because the internal counter can measure and display the input frequency, such tasks as verifying tuning accuracy and bandwidth accuracy are much simpler than before. In addition, the correction factors generated by the amplitude error-correction routine can be displayed on the CRT, giving the user a quick check on IF section performance (Fig. 1).

Each front-panel key has a secondary function that is accessed with the SHIFT key. For example, pressing the SHIFT key before the FREQUENCY SPAN key is pressed activates the error-correction routine. Several of the shift functions were designed into the analyzer to facilitate troubleshooting. One tells the microprocessor to count and display the actual sweep time, using the internal 10-MHz standard as a clock. It is thus possible to verify the various sweep times without using any external test equipment. This function simultaneously checks the operation of the internal counter.

Other shift functions enable the direct frequency counting of the signal IF, pilot IF, and VTO signals (the VTO generates the frequency offset for the pilot IF path). Another permits direct front-panel control of the digital-to-analog converters (DACs) that normally are controlled by the frequency-tuning algorithms. As an example of how this might be used, the DACs that control the VTO can be set to the end points, 0 and 1023, and the VTO frequency counted at each end point. This enables verification of the VTO oscillator, its tuning range, and the DACs that tune it, all from the front panel.

Making the Unknown Visible

Because of the complexity of the frequency-tuning algorithms, it is not easy for the technician to use the tuning equations to determine many of the internal control settings at a given center frequency and frequency span. One shift function, FREQUENCY DIAGNOSTICS ON, displays most of these settings. These include the tuning DAC settings, the harmonic number, the divide-by-M phase lock numbers, and the calculated frequencies for the VTO and the pilot third LO. Once these numbers are obtained, it is possible to probe the internal circuitry to verify proper operation.

Because the tuning algorithm for the YTO (YIG-tuned first local oscillator) is an iterative process, some failures cause the microprocessor to spend a long time trying to phase lock the YTO before deciding a failure has occurred, in which case it would display an error message and then initiate a sweep. To avoid this time delay during troubleshooting, a shift key function and an internal test point are provided to perform a phase-lock, flag-inhibit function. This tells the microprocessor to ignore errors when tuning the YTO and to sweep as though everything were all right. This essentially corresponds to opening the loop, permitting the analyzer to be tuned during troubleshooting without the microprocessor's continually trying to correct for the YTO tuning failure.

The power of the internal microprocessor can be further enhanced by connecting an external controller such as the Model 9825A Desktop Computer to the analyzer through the HP interface bus. This permits automated testing and alignment procedures (see box).

Troubleshooting the Digital Section

All the troubleshooting aids just described assume that the digital sections are operating properly. The design of the main microprocessor, the display processor, and the HP-IB interface microprocessor permits independent verification and troubleshooting of each. Each has its own test software and can operate independently for testing purposes.

To provide an overall system go/no-go test, two red LEDs were added to the front panel (INSTR CHECK). Whenever the instrument is turned on or the INSTR PRESET key is pressed, the two LEDs are turned on. The main processor then goes through a self check of internal working registers, performs a checksum of the program memory, pattern checks the display memory, and reads the keyboard to verify that there are no stuck keys or stuck I/O lines. If all these checks pass, then the two LEDs are turned off. If a bad bit is detected in the display memory, LED I stays on; if the I/O-keyboard check failed, LED II stays on. Both LEDs' staying on indicates the probability of a failure in the main microprocessor or the program memory.

Grounding an internal test point forces the processing circuits to cycle repeatedly through this test plus an additional RAM test. Monitoring the amount of time spent in each check provides an indication of the particular ROM, RAM, or display storage bit that failed.

When a problem with the display processor or display memory is suspected (LED I remains on), jumpering two test points enables a special set of test ROMs and disables the normal ROMs. A special CRT test pattern is then displayed, verifying the display processor and memory independently from the main microprocessor, the interface circuits, and the interconnect cable.

Troubleshooting with Signature Analysis

Once a digital failure has been detected and the suspect processor has been identified by the self-check routines, the problem becomes one of finding the faulty IC. This is done with signature analysis, using the Model 5004 Signature Analyzer.[2]

Designing for signature analysis requires very little hardware, but the few items that are required are essential. One of these is a jumper plug to open the feedback from the ROM outputs to the processor so the processor can free-run through all memory locations. The signature analyzer can then be used to verify all ROM outputs by comparing the signature at each output with the known good signature.

Once the ROM and the processor's program counter have been verified and the jumper plug replaced, the ROM programs become the stimulus for testing other devices on the processor bus. Several different stimulus programs are stored in ROM. The first program simply outputs various bit patterns on the bus to check the processor and the output bus. The remaining test programs check, usually one at a time, the other circuits on the bus such as the ALUs, RAMs, displays, keyboard scanners, counters, and so on.

Each test program supplies a synchronous, completely defined, repetitive stimulus that checks the functions of each circuit under test. The stimulus program cannot, of course, rely on any feedback from the device being tested or any device not previously verified; otherwise, it would not be possible to guarantee that the stimulus signals are valid. However, once the ALU is verified, it can be used to generate the test pattern for the RAMs and this basic "kernel" grows until the entire system is verified or the original fault found.

For each stimulus program, then, feedback paths need to be opened. This can sometimes be done in software by simply ignoring certain bus outputs, but it often requires grounding test points that have been designed in to permit qualifiers, interrupts, and so on to be inhibited. In addition, all asynchronous timing is either inhibited or tested only at times when it can be guaranteed to appear synchronous.

Inputs to the system from external sources have to be defined and preferably stimulated by the processor. Two special test extender boards were designed to break internal feedback paths and connect processor-controlled outputs to the external inputs for each board. This permits a complete check of the entire board, including the interface circuitry.

The only additional hardware required to properly implement signature analysis was the test points for connecting the start, stop, and clock inputs of the signature analyzer.

Troubleshooting Aids

The signatures are documented on multicolored diagrams that are removable from the service manual. Printed in black on these diagrams are the good signatures for the IC pins. A green verification path shows the output signatures that must be checked to verify that the board is operating properly. When a bad signature is located, information in red indicates what IC pins should be probed next to backtrack the fault to its origin. Printed in red next to the signature of an input pin is the IC number and pin number of the source of that input, so backtracking can be performed without continually referring to the schematic. Also printed on the diagram are the setup requirements for the test, such as the jumpers required to enable the stimulus test program and the start, stop, and clock test points for the signature analyzer.

Reference

1. R.A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," Hewlett-Packard Journal, May 1977.
2. A.Y. Chan, "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits," Hewlett-Packard Journal, May 1977.

David D. Sharrit
A native of Phoenix, Arizona, Dave Sharrit worked 3 years on radar signal processing before joining HP in 1973. Initially he worked on the signal-processing circuits for the 8505A Network Analyzer, which earned him two patents, before going to work on the 8568A. Dave has a BSEE degree from Arizona State University (1970) and has done graduate work at Stanford University. In off hours, Dave enjoys outdoor activities, primarily hiking and bicycling.

Team up a μP with signature analysis and ease troubleshooting in the field

Design troubleshooting aids into your microprocessor-based product right at the start, and you can be sure it will be serviceable later on. In fact, one powerful aid is already there—the μP itself. But adding to that an HP-developed technique called signature analysis—which compresses a long data stream into a unique, readily recognizable "signature" of four hex characters (see box)—will enable you to isolate system faults right down to a single node.

With a μP as part of the design, direct connections between the front-panel controls and circuitry often give way to digitally scanned keyboards, processors, ROMs, RAMs, a/d and d/a converters and multiplexed buses. Without the proper tools, then troubleshooting to the component level can become expensive, time-consuming and frustrating. But thanks to the μP, you can add special test routines and functions that take just 5 to 10% of program space, yet verify performance, simplify adjustments and troubleshooting, and detect some internal failures.

For example, in one HP spectrum analyzer, a front-panel function performs a calibration check on the i-f section, and measures the gain and center frequency of all the bandwidth settings. Deviations are displayed on the analyzer's CRT. Another function measures and displays the analyzer's actual sweep time, so the sweep can be checked against the setting. If any of the internal phase-locked loops fall out of lock, the processor automatically displays that condition on the CRT. Other functions permit the internal d/a converters to be programmed directly for testing, internal control settings to be displayed or internal frequencies to be directly counted.

Check the processor

The processor's ability to help verify and troubleshoot extends from the analog to the digital portions of an instrument. When a processor controls an instrument entirely, you must determine whether the processor is functional before attempting to isolate faults in the rest of the instrument. In a multiprocessor-based instrument, isolating the faulty processor is especially crucial.

Consequently, the primary function of a processor's self-test program is to verify the processor and its digital circuitry, or as much of it as possible. Either a self-test routine will activate every time the instrument comes on or an instrument-preset key or test switch will initiate the test routine. The routine generally includes a pattern test of the processor's internal registers, a checksum verification of the program in ROM, a pattern test of the system RAM, a visual test pattern shown on the instrument's display (CRT, LEDs, etc.), and a check of some of the I/O devices. The go/no-go results of self-test are displayed on the front panel, either by LEDs or by a turn-on message on the CRT.

In operation, a preset/power-up line forces the main processor to its reset condition while turning on red check LEDs on the front panel (Fig. 1). The processor then performs a read-write pattern check of its internal accumulators and registers, performs a checksum of each of the 16 program ROMs and a pattern check on the 16 RAMs, executes a read-write check on the display RAM memory through the digital-storage processor and, finally, reads the status of the front-panel keys from the interface board.

If all tests pass, the check LEDs (two in this case) are turned off. Both on indicates a problem with the main processor, ROM or RAM. One LED on indicates a probable problem in the digital-storage processor or its display RAM; the other LED on indicates an unexpected bit from the interface board.

The HPIB processor board has its own self-check routine and front-panel LED. In addition, the digital-storage processor verifies both itself and its RAM (through an internally-selectable self-test pattern), and generates a CRT test pattern, independently of the main processor and the instrument I/O bus. Usually, then, it's possible to both detect a fault and isolate it to one of the three processors or the I/O bus, from the front panel.

Both the front-panel keys and the self-check routine itself can be checked by keeping a key depressed during the main self-test routine. Problems will be detected when the interface board is read. And one check LED will be kept on to ensure that the two check LEDs aren't turned off improperly.

David Sharrit, Development Engineer, Hewlett-Packard, 1400 Fountain Grove Parkway, Santa Rosa, CA 95404

Reprinted from *Electronic Design*, January 4, 1979.

At times, the processor's self-check routine can also isolate faults. For example, when checking the program ROMs, the routine generates a checksum for each ROM, which enables the processor to tell you which ROM is bad. One way to tell you is to display the faulty IC number on the CRT. But this would require a known-good I/O bus, digital-storage processor and display RAM.

A simpler method requires a minimal amount of known working circuitry: Write the ROM checksum routine so that the time required to execute unambiguously indicates the bad ROM (Fig. 2). Here, checksums are generated for the upper and lower bytes of the first two kwords. If the sums agree with that stored in the first ROM, then the program continues to the next pair of ROMs; if they don't agree, the routine terminates.

Enter signature analysis

To monitor the execution time of this routine, a 4-bit ring counter hangs on the memory address bus, and is cleared and clocked by the main processor using appropriate addresses. Before starting the routine, the processor clears and clocks the counter once, then does it again when the routine is terminated. The time between the rising edges of the first two counter outputs can then be monitored.

A signature analyzer makes a good monitor here, since with a steady ONE at its data probe, the unit generates a unique signature that depends on the number of clocks between the start and stop inputs. You can then reference the signatures to the bad ROM using a fault table (Fig. 3).

For instance, a signature of 6HF5 indicates that U6 is bad; one of UCF4 indicates that all ROMs are good. Bad RAMs are similarly isolated with their own pattern-check routine and another fault table, using the next pair of counter outputs as the start/stop control lines. The complete self-check routine can be forced to repeat continually by grounding a status line into the processor. Not only does this simplify testing, it helps isolate intermittent failures.

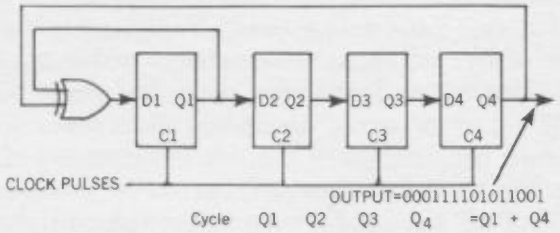
Self-test obviously has its limitations. To run the routine just to check the program ROMs requires that some part of the ROM and a good portion of the processor be operating properly.

If the routine doesn't run, you'll need an additional test, normally called a "free-run" check, to isolate the first ROM, containing the self-test program, from the processor. This is usually done by breaking the feedback path from the ROM back to the processor, thus forcing a NOP instruction, and continually causing the processor to increment the memory address. The actual outputs of the ROM, chip-select lines, and memory address lines can then be verified using the signature analyzer.

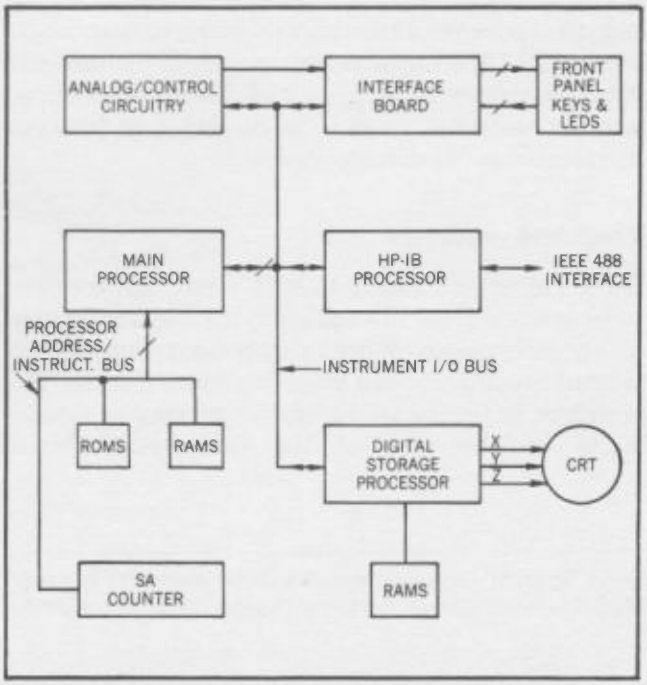
If the processor's memory-output lines don't sequence properly, you may have to do some additional troubleshooting with an oscilloscope to verify the

Revisiting signature analysis

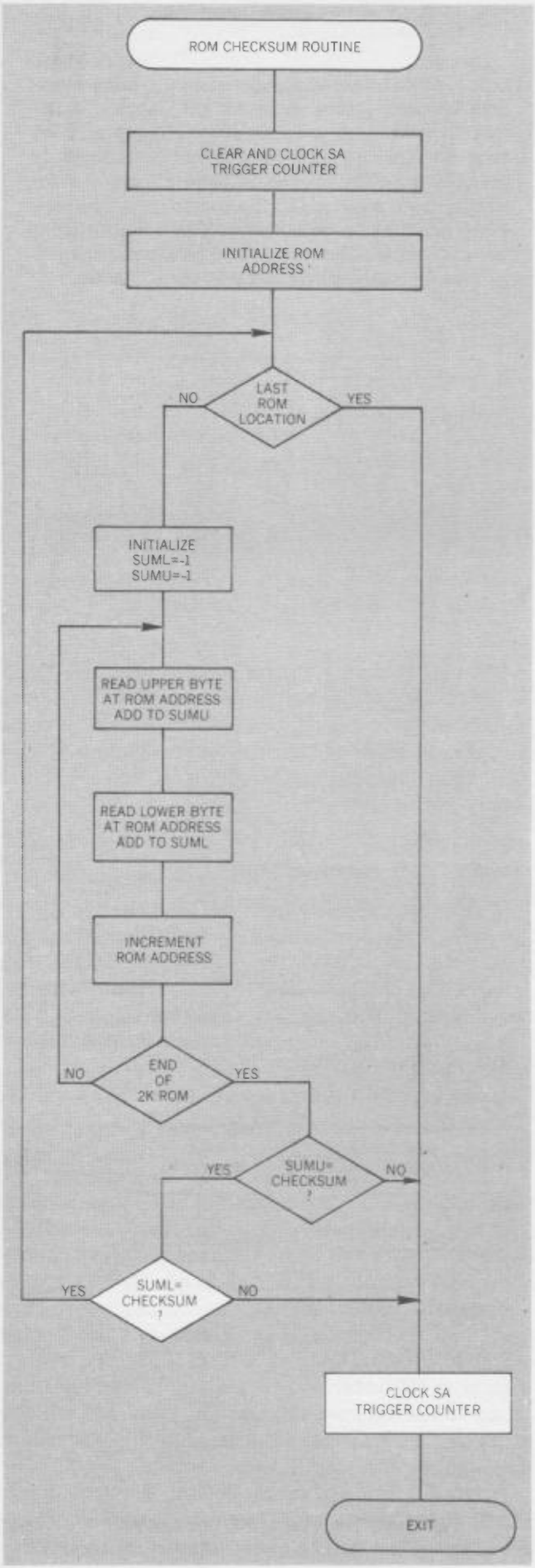
In signature analysis, a cyclic redundancy check code, a sort of checksum, is produced by a pseudorandom binary-sequence generator (see figure). In the figure, a feedback shift register produces a 15-bit sequence: The outputs of the flip flops go through all possible non-ZERO four-bit patterns, and the sequence then repeats. In effect, data at a node are compressed into a form that is handled easily by a portable tester. The compressed data are displayed as four hexadecimal characters—the signature.



	Cycle	Q1	Q2	Q3	Q_4	=Q1 + Q4
INITIAL STATE:	0	1	0	0	0	1
	1	1	1	0	0	1
	2	1	1	1	0	1
	3	1	1	1	1	0
	4	0	1	1	1	1
	5	1	0	1	1	0
	6	0	1	0	1	1
	7	1	0	1	0	1
	8	1	1	0	1	0
	9	0	1	1	0	0
	10	0	0	1	1	1
	11	1	0	0	1	0
	12	0	1	0	0	0
	13	0	0	1	0	0
	14	0	0	0	1	1
BEGINS TO REPEAT:	15	1	0	0	0	1



1. **The digital section of a spectrum analyzer** is a likely candidate for self-testing under internal μP control.



processor's clocks, reset and power-supply inputs. If those fall within specifications, you've isolated the fault to the processor chip. If the ROM outputs are good, but the self-check routine still doesn't run when the ROM outputs are reconnected, the processor is faulty.

Other aids are needed

Generally, of course, troubleshooting this way doesn't pin down exactly which processor function has failed. You'll probably call for a logic analyzer or other debug aid in the design phase to troubleshoot the actual program steps or determine precisely what a processor is (or isn't) doing. But the test will give a go/no-go indication of the processor's performance at its full operating speed and in its true environment. And once the basic "kernel" of processor and first ROM are verified, you can use the kernel to test the remaining ROM and RAM, which in turn can help you test the I/O and other digital circuitry, and so on.

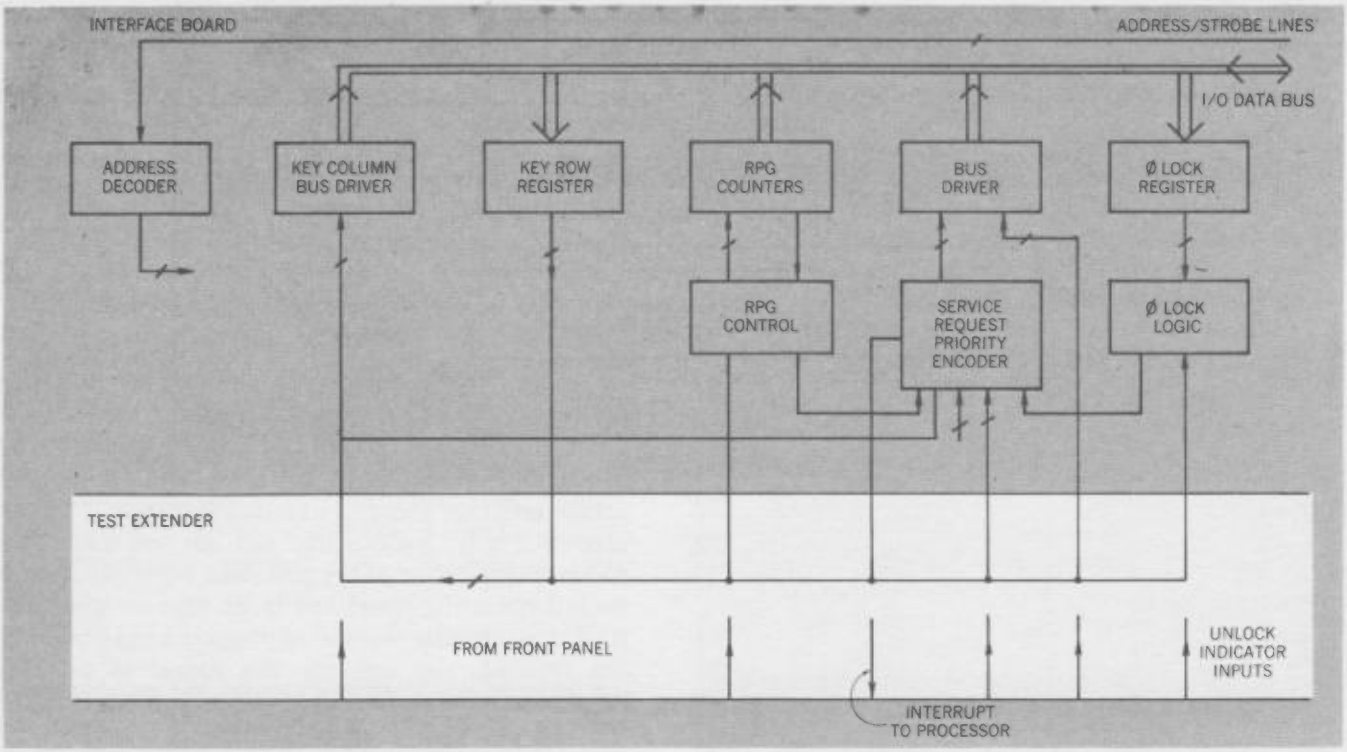
Self-testing also shows its limitations when you attempt to isolate faults in the I/O or other circuitry not directly connected to a processor bus. The processor can't always read the outputs of the I/O device —a self-test requirement—though in some cases, it's worth adding the circuitry required to read back the output of an I/O port.

In other situations, the processor can't exercise all

5 VDC SIGNATURE	COURSE OF ACTION
UCF4	ROM IS GOOD. PROCEED TO RAM CHECK.
U789	REPLACE HYBRID PROCESSOR A15U13.
05C7	U34
095A	U3
0F25	U31
2986	U35
2HP3	U29
31HP	U32
34P5	U33
394U	U5
512U	U2
5PUC	U36
61A0	U7
6HF5	U6
77A0	U4
78FP	U1
CH44	U30
CPU1	U8

3. **Suspect ROMs and RAMs can be spotted** with a look-up fault table, which lists signatures for both bad and good memories.

2. **A checksum routine verifies** that all program ROMs are OK. Here, the execution time gives away the bad ROM.



4. **Mating the technique of signature analysis with** a test program that generates circuit stimulii is a good way to check out circuitry, such as this portion of a spectrum-analyzer interface board.

the inputs required to verify an I/O device thoroughly in its normal configuration. Or it may be able to verify a section of circuitry but not easily pinpoint the faulty device. Is the input buffer bad, or the output buffer? A good way to find out is to team up the processor again with a signature analyzer.

A routine in the processor's program generates the stimulii for the various I/O peripherals; the analyzer verifies the logic circuitry and traces bad signals to their starting point. The stimulus program usually differs from the self-test program because the stimulii must not change because a device is good or bad.

Besides designing the stimulus program, you may have to modify the I/O to accommodate a test configuration. Here are two tips:

1. Either avoid asynchronous timing signals, or disable or ignore them during test modes.

2. Disable feedback paths, such as interrupts to the processor or circuit under test, since a failure anywhere in the loop will make all logic signals appear bad and make isolation impossible. Disabling is easiest when software performs the feedback functions, since a software "path" is easily opened.

To disable a hardware feedback, generally you can add a strategic test point to be grounded during testing, or a jumper to be removed. Or you can connect an unused processor-controlled output to a point in the loop that allows the feedback path to be disabled. Proper board partitioning is as always helpful in troubleshooting, since some feedback paths can be broken and bus-loading problems isolated simply by removing certain boards.

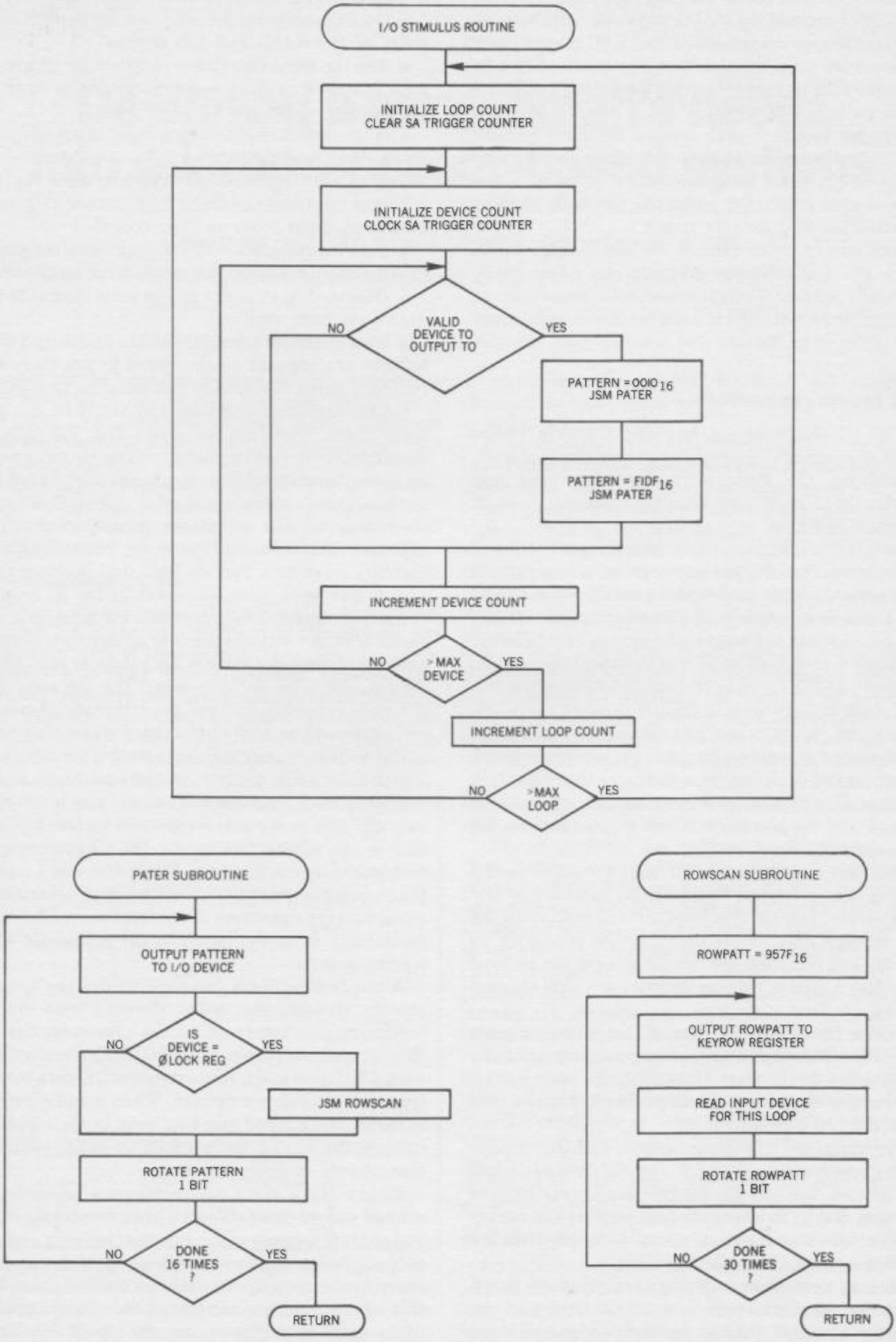
You should also provide some way to stimulate those I/O inputs that aren't under processor control. Jumpering those inputs to processor-controlled outputs (which can be independently verified) usually provides the best stimulus—yet it may be enough to manually connect a line alternately to ground and ONE, and verify both results.

Putting it all together

The spectrum-analyzer interface board is a good example of signature analysis coupled with a stimulus test program (Fig. 4). The board contains the logic required to interface the main processor with the instrument's front panel. It also handles the service-request logic, including lock/unlock indicators, key-down indication, HPIB requests, and so on. Communication with the processor takes place via the bidirectional 16-bit data bus, an address bus, a strobe line and an interrupt line.

During the self-check routine, the processor can check the board only minimally. To do so, it clears the rotary-pulse-generator (RPG) counter, reads the outputs to verify all ZEROs and reads the key-column lines to verify all highs. But to check the circuitry thoroughly requires a repetitive, synchronous stimulus to all the various inputs. Because of the number of inputs, you may need a special-purpose test extender to disconnect the normal inputs and reconnect them to the key-row outputs. The processor can now control all the inputs by outputting bit patterns to the key-row register.

And the test extender also does other things. The extender breaks several feedback loops



5. **A stimulus program** puts an I/O bus (and connected devices) through its paces with different patterns.

on the board and opens the interrupt feedback from the service-request logic to the processor. And because the clear line on the one-shot in the RPG control block is connected to a key-row line, the one-shot can be cleared by the processor to avoid the timing ambiguity of its pulse width.

Another feedback path around the RPG counter detects the overflow state and inhibits any further clocks. A grounded test point connected to an otherwise unused gate input opens the feedback, thereby isolating the two counter stages.

Once you've taken care of the test configuration, write the test-stimulus program. In many cases, especially with an 8-bit processor, all it takes to cover all possible bit patterns is a simple increment/output loop, from 0 to 255.

Test pattern requirements

With a 16-bit processor, however, the cycle time to count through all possible states becomes excessive. In addition, the service-request priority logic may require that a particular sequence of bit patterns test all possible states.

Testing the spectrum-analyzer interface board and the other devices on the bus requires a test pattern with several rotate/output patterns (Fig. 5). For most I/O-bus devices, a fairly simple test stimulus suffices, one generated in the program by cycling two different patterns through all valid I/O devices. One pattern is simply a ONE rotating through the 16 bit positions. The other pattern rotates ZEROs through the word.

To check the more complex service-request logic, a third pattern sequence stimulates the key-row register whenever the device address indicates the phase-lock register. During this sequence, an I/O input device is read into the processor; which one depends on the loop count.

The first time through the loop, the input device acts as the processor's accumulator, so no actual I/O read occurs. During that loop, all information on the bus depends only on the processor. So even with all I/O devices removed, the bus signatures can be verified. The key-row register outputs are also checked during the first loop.

During the second loop through, the processor reads the RPG/SRQ word. Any faults now appearing on the bus can be traced back through those logic blocks. During the third loop, the key-column word is read onto the I/O bus, and so on.

Generally, you'll need one setup for each driver since it isn't possible to trace a fault if two unverified devices are driving the bus during the same measurement window. But by indexing the read address and changing the start/stop inputs, it's possible for one software routine to test all the I/O devices.

It's easy to overlook verifying a test program. Don't. Does the program really test all the states of the priority encoder? Taking the time to question the capability of critical sections of the test program with a logic analyzer can save time in the long run:

- Does the program properly test the pattern sensitivity of the RAMs and I/O devices?
- Are the same signatures obtained on all known-good boards or is there some marginal timing in the test-pattern sequence?
- Is an indeterminate open-logic input affecting some signature?
- If power is cycled off and on, does the test generate the same results or does the test rely on an undefined input from another board?
- Does inducing faults by jumping signals to ground or supply show up on the verification outputs?
- Does the test stimulus rely on some hardware that hasn't yet been verified?
- Does the input stimulus remain unchanged when failures are induced on the board or are there still feedback paths that make fault isolation impossible?

Go through that checklist and you'll be one step away from signature analysis. The last stop is documentation. Good signatures must be documented so those troubleshooting the board will know if a particular node signature is good or bad. One way is to include all the signatures in a flow chart, an approach that can work well for over-all troubleshooting down to a module level, but becomes risky and cumbersome when extended to the IC level.

Another approach is to annotate the schematic with signatures, not with the familiar analog voltage levels and waveforms. An obvious advantage is that all the information is on one document: The signature tells if a node is good or bad and the schematic shows the node connections and what inputs affect that node.

Unfortunately, signature-annotating the schematic —especially on a medium to complex board or one requiring more than one test set-up—can become not only difficult to lay out, correct and update but hard to use. An alternative prints the signatures on a component-layout diagram. While this can't replace the schematic, additional color-coded information along with the signatures minimizes the need to switch continually from the board to the schematic while tracing a fault.

A verification path (arrows) guides the troubleshooter through the nodes. Checked here are the service-request interrupt to the processor, the I/O data-output lines from the RPG/SRQ circuitry and, with a different start/stop setup, the I/O data outputs from the key-column drivers. When a bad signature is found, the colored numbers next to the signature indicate the IC and the pin number of the source of that signal.

At the IC, a black pad represents an output, a colored pad an input. Colored lines connecting inputs and outputs indicate which inputs affect only a certain output. Isolated colored pads indicate inputs that control multiple outputs, such as a clock or clear. With this information, you can trace a bad output signature to its point of origin—where the inputs to a device are good but the output signature is incorrect. ■■

Designing a serviceman's needs into microprocessor-based systems

Mapping tells him where to start; signature analysis locates circuit nodes in trouble; diagnostics check operation

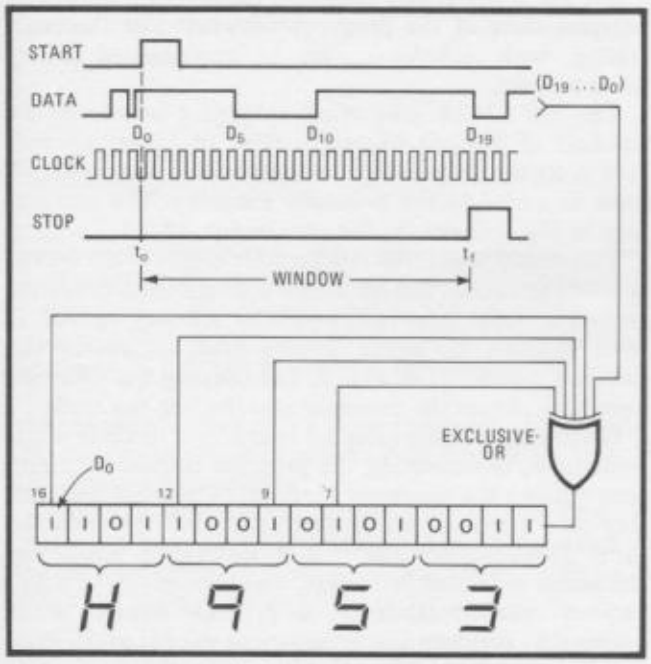
by Martin Neil and Randy Goodner, *Hewlett-Packard Co., Santa Clara Division, Santa Clara, Calif.*

☐ Mention the word "microprocessor" to a serviceman, and watch his face cloud over. He knows that the many advantages the chips offer to both manufacturer and user are coupled with new problems and challenges from his point of view.

There is a way to ease his job, and it is the familiar tack of designing serviceability into the product. However, many designers are unfamiliar with the special servicing requirements of microprocessor-based products.

Troubleshooting woes

Several characteristics associated with such designs present troubleshooting problems for traditional test equipment. One of these is that characterization of the circuitry is difficult because processor firmware often replaces hardware and its operation may be hidden in the software algorithm. A related problem is the dynamic operation of these products, where signals often are active for a few microseconds and then disappear. In a microprocessor-controlled keyboard, for example, checking for signal faults requires knowing when to look as well as where to look.



1. Signature analyzed. This 4-place hexadecimal signature is a compression of the 20-bit data stream entering the linear-feedback shift register. The unique character set, consisting of 0 . . . 9, 4, C, F, P, and U, allows use of a seven-segment display.

A third problem is that the bidirectional nature of the processor bus makes interpretations of address and data information very difficult. Compounding this is the buses's parallel structure, which has many devices ORed together, making for tedious fault detection. Furthermore, the test gear must contend with many operations, since newer instruments may have the processor going through a few thousand steps in a measurement cycle where earlier products usually required fewer than 100 operations.

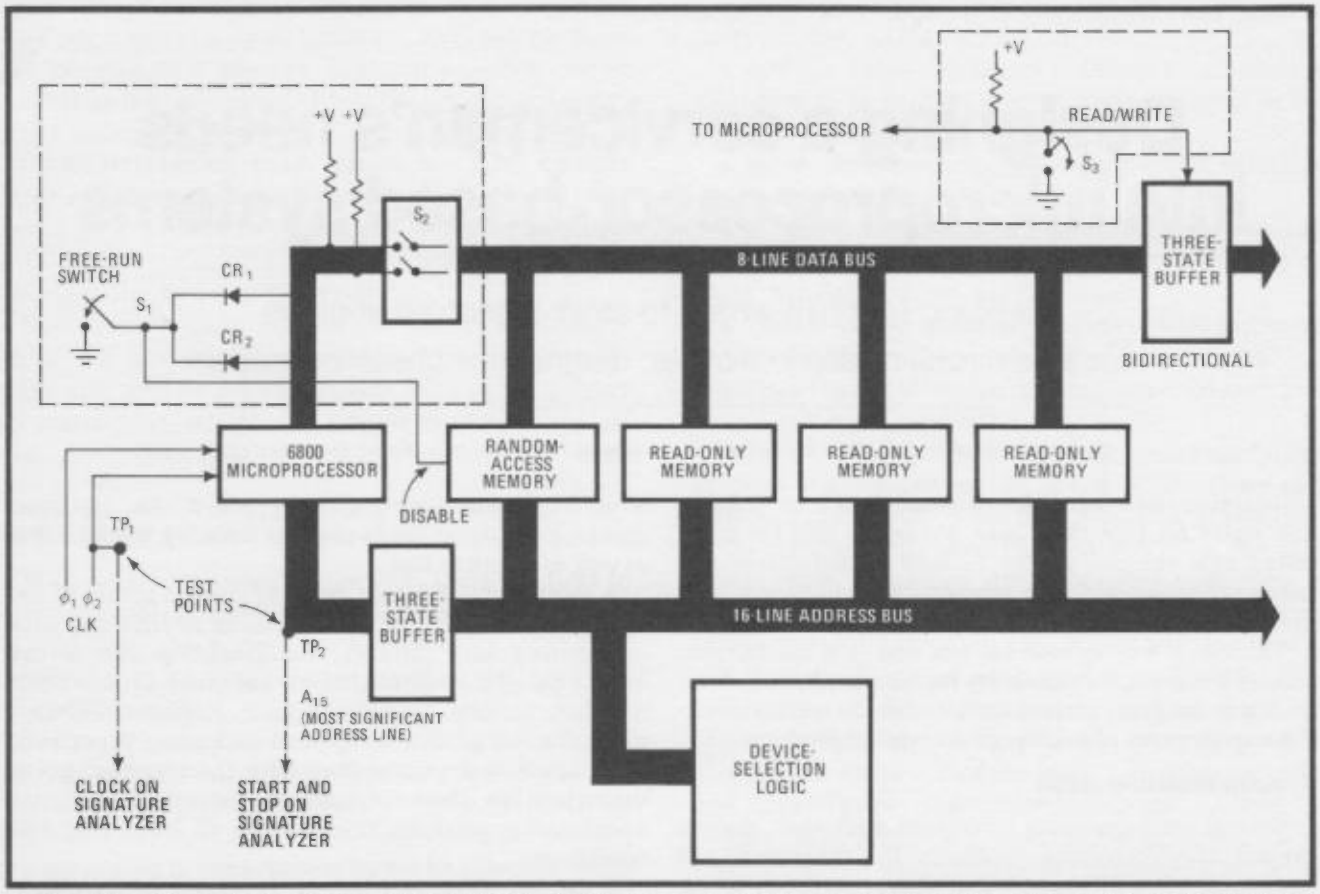
A partial solution to these problems is logic-state analyzers, which help trace the microprocessor's operating algorithm by following the sequence of machine states. Once the state in which the fault first appears is located, traditional test equipment comes into play in order to trace through the nested circuits seeking the component or components giving rise to the fault. However, this procedure can take much time and invariably requires a highly skilled serviceman.

Failures involving the microprocessor assembly and those assemblies that interface to the processor are extremely difficult to troubleshoot. In fact, it may be impossible to isolate these failures with traditional troubleshooting equipment and techniques. Hours of investigation may not determine whether the fault is a control failure originating in the microprocessor assembly, an interface failure, or a failure in some other assembly causing the measurement algorithms to hang up or branch to an incorrect program segment.

To chase the clouds away from the serviceman's face, the designers of microprocessor-based products must take a new look at designing in serviceability. They should look closely at three techniques that can overcome these troubleshooting problems: signature analysis, built-in diagnostics, and mapping. What follows is a discussion of these three, coupled with examples of how specific designs help the serviceman.

Signature analysis

Perhaps the greatest serviceability boon a designer can offer to the product's user is the capability to work with such troubleshooting tools as the HP 5004A signature analyzer. A relatively new technique, signature analysis



2. Open the loop. To implement free-running signature analysis, it is necessary to open the microprocessor assembly's feedback loop to prevent alterations to the free-run instruction. This can be done on the board itself by simply adding a few components.

uses data compression to reduce complex, serial data-stream patterns of any length at a circuit node to a unique four-digit hexadecimal signature [*Electronics*, March 3, 1977, p. 89].

Comparing the signature of a suspect circuit node with the empirically determined correct signature in the service manual will quickly verify circuit operation. The cause of an incorrect signature may be quickly discovered by probing designated nodes, observing good and bad signatures, and tracing the signal flow.

Besides the data from the circuit node, the 5004A needs three signals. A start signal initiates the measurement window during which data is clocked into a 16-bit linear-feedback shift register in the analyzer. A clock signal generated by the unit under test synchronizes the data with the signal analyzer. A stop signal terminates the measurement window, and the 16-bit residue in the shift register appears on the display as a 4-place hexadecimal signature (Fig. 1).

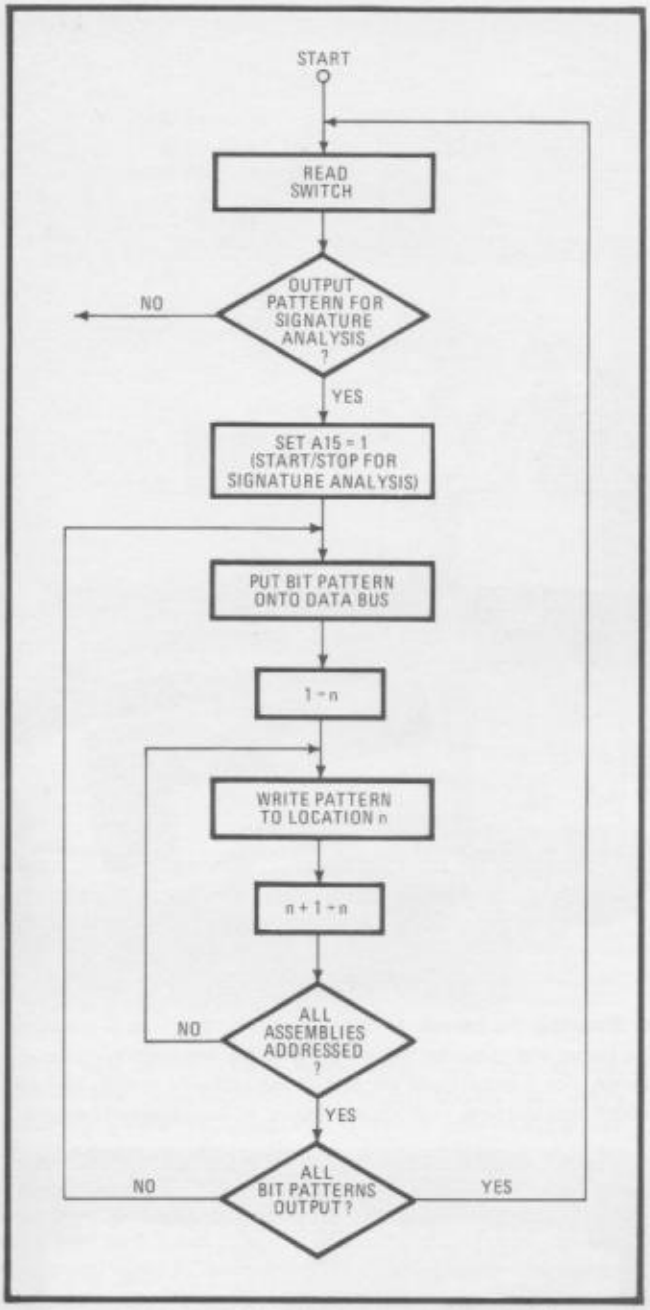
The free-running and software-driven modes are the two basic implementations of signature analysis in a microprocessor-based product. Free-running analysis forces the processor to cycle continuously through its entire address field with start/stop signals derived from the lines of its address bus. Software-drive analysis uses a stimulus program stored in read-only memory to generate the start/stop signals and to write repeatable data streams onto the data bus in order to test assemblies connected to the processor.

The free-running mode occupies no ROM space, while the software-driven mode typically requires something under 5% of this space. However, the software mode can exercise more of the product's circuitry. For thorough testing, both techniques may be implemented in the same product.

The HP 5342A microwave frequency counter is an example of efficient implementation of free-run signature analysis: only a few switches and pullup resistors need be added to the processor assembly. The outlined area in Fig. 2 shows the few components added.

Essentially, what free-run analysis checks is the operation of the kernel: the minimum configuration of microprocessor, ROM, and random-access memory needed to cycle through the entire address field. Grounding the free-run switch, S_1 in Fig. 2, and opening the data-bus switch, S_2, forces the processor into the free-run mode.

Grounding S_1 generates an instruction to clear accumulator B, incrementing the program counter by 1 and thus cycling the processor through its address field one step at a time. A NO OP instruction will perform the same function, but the CLR B instruction needs the minimum of added hardware: two diodes. Closing S_1 allows incrementation of the program counter, while opening S_2 prevents the ROM output data from altering the free-run instruction. Consequently, the processor's address lines cycle repeatedly over the entire address field from 0000 to FFFF. Using the most significant address line (A_{15}) as start/stop signals and one phase of



3. Program-driven analysis. This program stored within a ROM generates the output patterns used to test the system in the program-driven signature-analysis mode. The most significant address bit A_{15} is set high to generate its start/stop signals.

the processor's clock as the clock signal will give repeatable, stable signatures for the address lines, ROM outputs and device-select outputs.

Switch S_3 enables the write buffer so that free-run signatures may be observed there. The read portion of the buffer cannot be checked with free-run analysis; it requires a logic pulser and logic probe.

Software-driven analysis

Exercising a special pattern stored in ROM will provide a more thorough signature analysis than will the free-run mode. This software-driven mode provides a pattern for the boards that are outside the kernel and therefore have no way of generating patterns. It allows troubleshooting of the input registers of devices accepting data from the microprocessor, and will often provide signatures for the circuitry responding to data in those registers.

A typical 8-bit output test pattern would start off with all 0s, go to all 1s, then continue with a walking 1 pattern (10000000, 01000000, . . . 00000001). The processor places the first byte of the pattern on the data bus and instructs a board to accept the byte in a specific location or register. For example, if the board has 12 locations, the processor will write the byte at all 12.

Once done, the processor sends the same byte to locations on the next board. After all locations on all boards have been loaded, the processor places the next byte on the data bus, and the process repeats.

One way to generate start/stop signals is setting the most significant address bit high at the beginning of the test program. With the test program in the lower half of the memory, the most significant address line, A_{15}, can be artificially toggeled at the beginning of the cycle simply by addressing a dummy location in the upper half of the memory. Figure 3 is a flow chart of the program for output pattern generation for this situation.

Program-driven signature analysis can also test inputs from boards that are not part of the kernel. An internal service switch can set the microprocessor to generate an input exercise. Such a test allows these boards to put latched data from every storage location onto the data bus for checking by signature analysis.

There are two primary points of concern in performing such a check of the read operation: determining by sequential elimination which board among many is malfunctioning and insuring that the circuits are properly initialized. Indeed, these concerns apply to both types of program-driven analysis and to free-run analysis as well. Moreover, they are part of a design checklist for designing in signature-analysis capability.

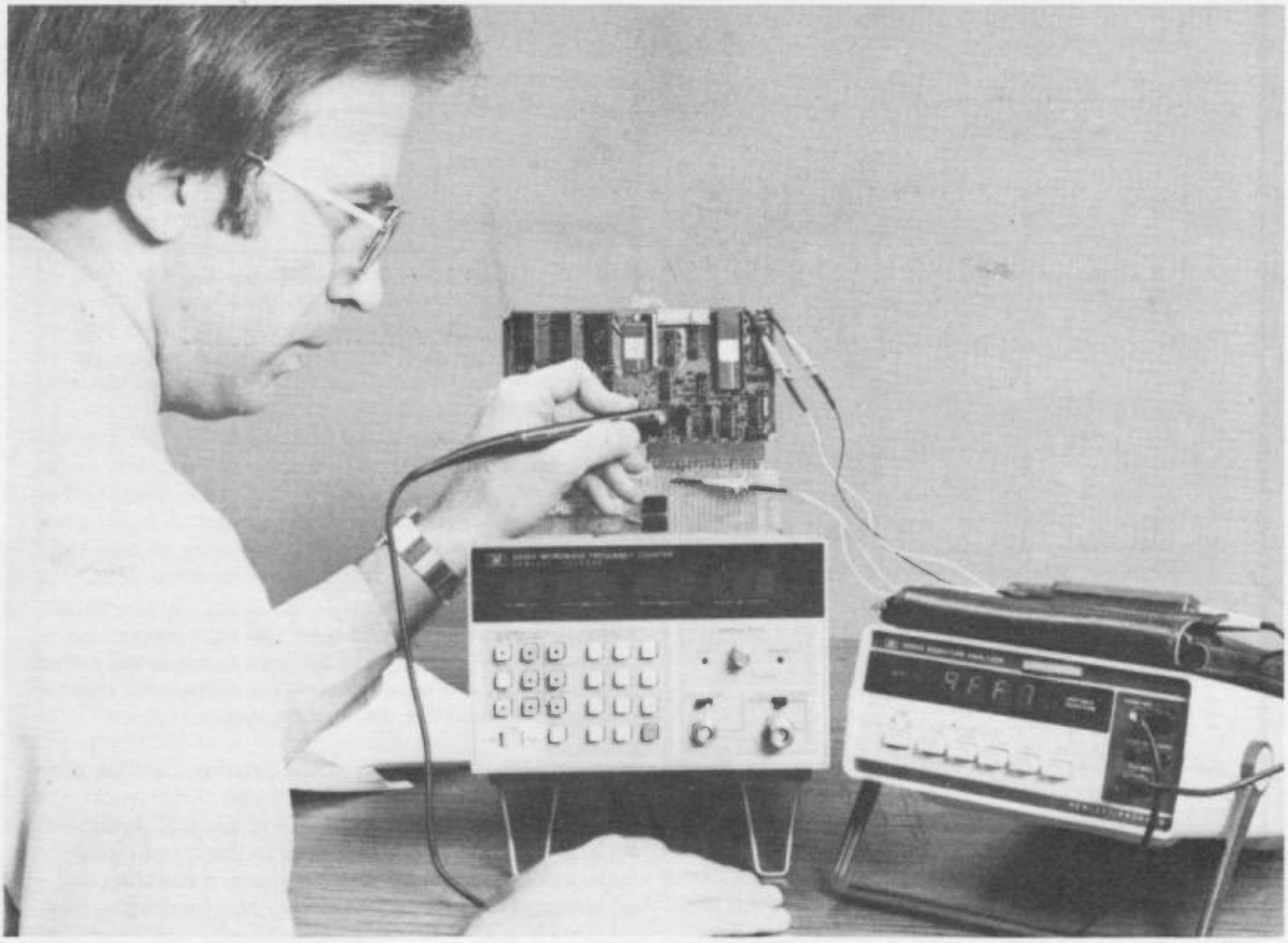
Design points to remember

Experience in designing microprocessor-based products that will work with signature analyzers suggests a list of six considerations that must be part of the design process. These six are:

- Distinguish the kernel.
- Provide a way to open local feedback paths.
- Insure initialization of the boards under test.
- Use address decoding to isolate ROM failures.
- Stabilize the signatures of three-state devices.
- Carefully document signatures.

The kernel should be on its own board, for one of the first steps in troubleshooting a malfunctioning system is to find out if the kernel free-runs. This test cannot be accomplished unless the kernel can be completely isolated (Fig. 4). If a separate board is not possible, then switches must be provided to isolate the kernel's components from the rest of the system.

A good case in point is a malfunction in which some device on the microprocessor's addresses and data buses continuously pulls a line high or low. It is not clear which device is malfunctioning. The solution is to attach to the kernel an extender board modified with isolating switches for the address and data lines (Fig. 5).



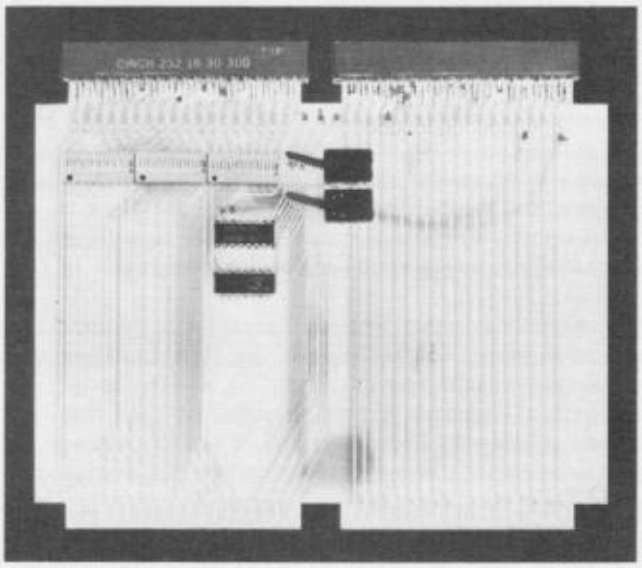
4. Shooting the kernel. An initial troubleshooting step is to isolate the kernel and check for signatures using the free-running mode, as shown. For a stuck node the rest of the system's boards may be added, one at a time, until a board causes a bad signature to appear.

To find the origin of stuck nodes on a product with an extender attached, the serviceman takes the signatures with all bus line switches closed. Then, on the lines giving incorrect signatures, he opens the extender-board switches and takes the signatures on their kernel side. A good signature on this side means that the bad signature on the other side is caused by a bad external device pulling that line high or low.

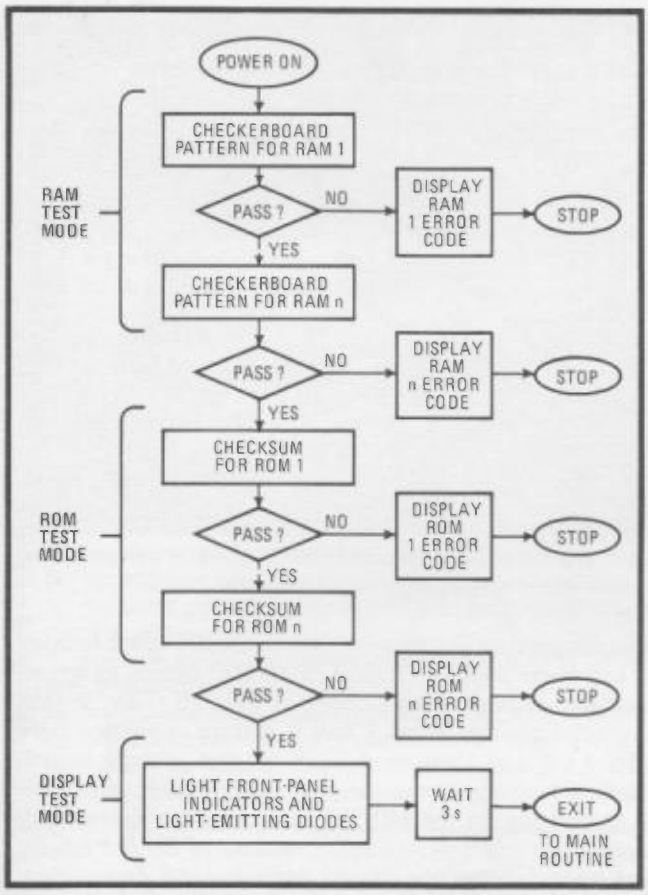
The next step is to determine which device is the bad one, and this can be a much simpler process if the designer puts each subsystem interfaced with the processor on its own board. Then the serviceman can simply add boards to the free-running kernel until the bad signature is seen again. He continues to use the extender board's switches to isolate stuck nodes.

A similar problem can occur with program-driven signature analysis that generates input tests. A read cycle involves all boards putting data on the bus, so it is necessary to test them in sequence by adding them one at a time to the kernel.

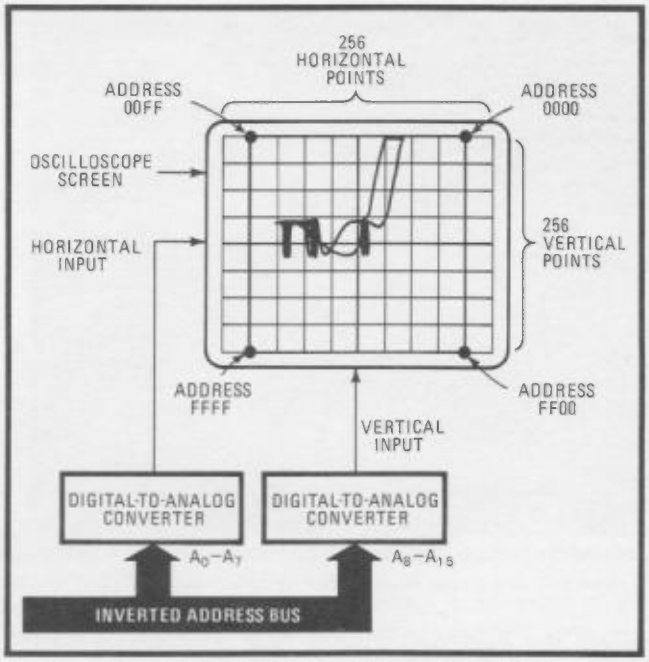
In this case, determining whether a particular board sends out data properly requires its isolation from other boards performing the same function. One tack is to pull all other boards putting out data, excluding the processor and ROM boards. Another tack is to isolate the board under test by an extender card with switches on the data lines. However, opening these switches does more than



5. Isolation. The modified extender board isolates the questionable boards' address lines and data lines from those of the bus. It also provides the serviceman with a manual reset capability and circuitry for address decoding (located near the board's center).



6. Power-on test. The diagnostic program charted describes a test routine for checking a system's RAMs, ROMs, and LEDs. The program is self-initiating at power-on and displays failure error codes. Internal switches or keyboard inputs can also start tests.



7. Mapping circuitry. Two 8-bit digital-to-analog converters provide oscilloscope mapping for a 16-bit address bus. This service aid can be built in or put on a separate board. The service manual must of course include maps of correctly operating systems.

isolate the board's data lines from the bus; it also removes the load supplied by the processor, and many switching power supplies will operate only with a full load. To compensate for this, the board needs another set of switches and some pullup resistors. With the switches, it is possible to select software designed to exercise only one board at a time.

Another way to isolate nodes is to operate the kernel in the free-run mode outside the instrument. To do this without an extender board, the serviceman may use jumpers to connect the supply, ground, and clock signal to the kernel. The designer can facilitate this procedure by providing easy access for the jumpers, such as test terminals.

The designer must provide the facility to open the feedback paths. If feedback is not broken when the signature-analysis program is sending a stimulus pattern to a device, then an error will propagate around the loop and all subsequent signatures will be wrong. It is a good idea to select an obvious feedback path to open, like the data bus of the kernel, where the added switches will interrupt the path.

Initialization of the boards under test is extremely important in signature analysis. Without initialization, consistent signatures cannot be guaranteed from one example of the product to another, or even in the same instrument from power-up to power-up. Some circuits initialize themselves with their own reset pulses after passing through one complete cycle. Otherwise, the designer must insure that the service manual documents an initialization procedure for the serviceman to follow.

Similarly, there must be provision for disabling asynchronous devices like monostables and interrupts. Also, multilayer boards should be avoided where there are bus lines going to several devices. It is difficult to use a current tracer with a multilayer board to discover which device is holding a bus line low or high.

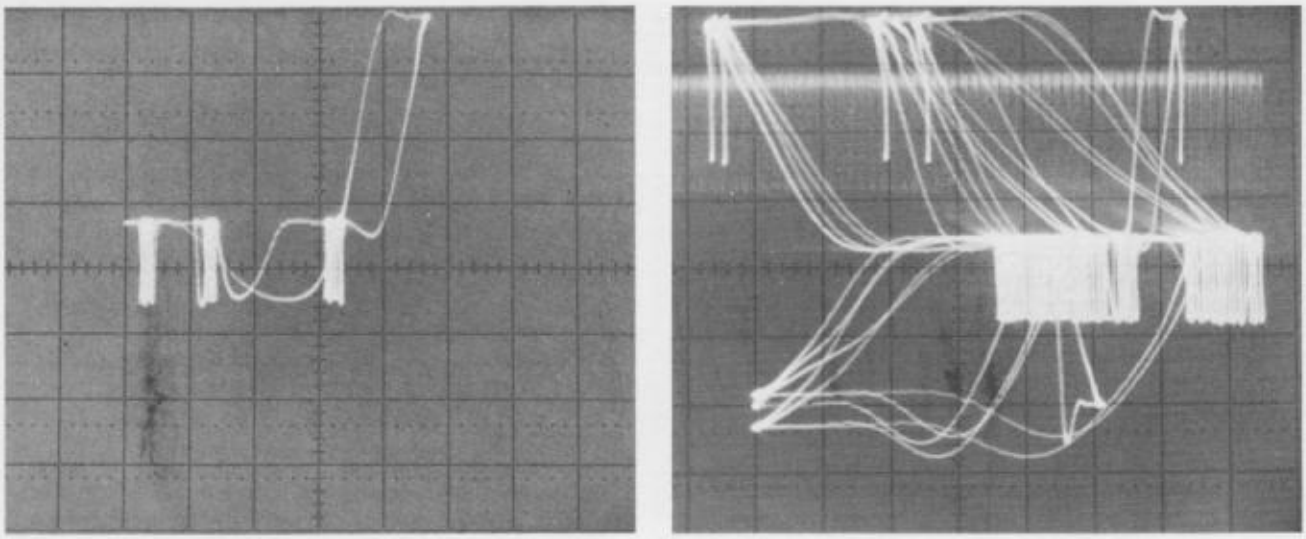
Spotting defective ROMs

The designer can facilitate isolation of a ROM failure by insuring that the address decoding circuitry is used to generate the measurement window for each chip. The chips shown added to the extender board in Fig. 5 perform this task. Then the start/stop signal for each ROM aligns with its first/last address, and the signatures appearing on the data bus come from only one device at a time (assuming that the output enables are working).

It also is important to provide for disconnection of the data bus when taking signatures on ROMs. In free-run analysis, other devices are being addressed and are putting out random data that results in erroneous signatures appearing on the bus lines.

Three-state devices can present a particular problem in obtaining stable, repeatable signatures. Nonrepeating signatures can occur if the start/stop window brackets the device's transition from high-impedance output to enable output when pullup resistors are absent.

To avoid this problem, pullup resistors should be on the board with the three-state device, not on some other board. Otherwise the extender board will need pullups to function when the board is isolated from the rest of the product. An alternative is to use address decoding that



8. Go/no-go pattern. The first map (a) indicates a properly operating system after the power-up diagnostic test. The second map (b) shows the same test on a similar system with the kernel malfunctioning. The serviceman checks the pattern, not individual points.

ensures the start/stop window only brackets the cycle time in which the three-state outputs are enabled.

Finally, it is up to the designer to provide the careful documentation of the signatures that will go into the service manual. Fortunately, there are several ways to insure that the signatures being read will be consistent from product to product.

The first step is to check that the signatures are stable and repeatable on several product samples.

The most stringent test features reducing the clock frequency by 50%. This procedure should leave the signatures unchanged—otherwise there is a potential timing problem, probably due to excessive bus settling time and usually caused by the lack of pullup resistors on the bus lines. To reduce the clock frequency, all that is necessary is to pull the kernel out of the unit and to use an external pulse generator as a clock instead of jumpering the clock from the unit under test.

Moreover, it is important always to document the characteristic 1s signature for each new setup so that the serviceman can determine if the start/stop signals supplied to the signature analyzer are correct. This characteristic signature is obtained by placing the data probe on a transistor-transistor-logic high level that remains high during the start/stop window.

Designing a microprocessor-based product to work with signature analyzers is a giant step towards serviceability. However, the designer can do more to help out field service, and one important consideration to take into account is the inclusion of diagnostic subroutines.

Built-in diagnostics

Product-initiated self tests and user-callable tests are the two basic kinds of built-in diagnostics. Self-initiated routines are automatically exercised by the product every time some preset condition is met. A user-callable routine is exercised only when the serviceman selects it.

Figure 6 is an example of the flow chart of a self-initiated test. A convenient implementation point for such diagnostics is power-on. When the processor fetches its power-on address in ROM, a pointer causes program execution to begin with a check of RAM memory. In this routine, a checkerboard 1 and 0 pattern is written into each RAM and then read back. If this pattern is not identical to the one sent, the processor initiates display of an error code. In the HP 5370A universal time-interval counter, for example, a display readout of *Err 6.1* means that RAM 1 failed the check, while *Err 6.2* means that RAM 2 failed, and so on.

If all the RAMs pass the test, a ROM checksum test is next. The contents at each ROM location are added up until the final checksum is compared with the correct one—in this case, the sum in the first location of each ROM. An incorrect checksum will cause an error message on the display. In the 5370, *Err 7.1* means ROM 1 failed the checksum test, and so on. (Of course, this routine works only if that portion of ROM containing the diagnostic program is good.)

If all ROMs pass their checksum tests, the self-diagnostic program advances to a display check. All segments of the light-emitting-diode display and all front-panel lamps are lit briefly. These are, of course, just a few examples of the many kinds of checks that can be performed at power-on.

User-callable diagnostics

There are two basic ways for a designer to implement user-callable diagnostics. Some routines can be initiated by pressing the appropriate front-panel keys. Others can be selected by setting internal servicing switches to the appropriate codes. There are many choices as to which diagnostic routines to include, with the choice depending on the type of product.

One generally useful diagnostic routine is the algorithm-tracing program. The product goes through its usual operating algorithm but also displays mnemonic codes at key points.

For example, in the HP 5342A microwave frequency counter, striking SET, SET, 0 on the keyboard initiates a diagnostic program display in four points in the algo-

Design for serviceability check list

Taking into account the following field-service considerations at the initial design stage will realize several advantages. Service calls are shortened, redesigns are eliminated, extra equipment is not needed, and the manufacturer and customer see their costs reduced.

1. Are inputs and outputs protected from normal abuse?
2. Are light-emitting diodes used to advantage inside the instrument to indicate proper operation? For instance, a lighted LED could indicate a locked phase-locked loop, clock is present, power-supply voltage OK, etc.
3. Are components located far enough from integrated circuits to allow test clips to be placed on the ICs?
4. Can feedback be easily disabled for troubleshooting purposes? A good example would be a jumper wire that can be removed to break feedback between several ICs on a board. Sometimes feedback can be broken by pulling a board in the feedback loop.
5. Are display LEDs easy to replace?
6. Are there power terminals for logic-probe/pulser/current-tracer operation?
7. Are interconnecting cables long enough to allow operation of boards when placed on extenders?
8. Are boards independent? Avoid, if possible, a device on one board and its pullup resistor on another. If this is necessary for proper line termination, provide on the board a switchable pullup resistor for stand-alone testing.
9. Is it possible to manually force a node to a particular state for troubleshooting purposes?
10. Can the instrument be operated with any of its circuit boards removed, i.e. does the power supply require a certain load?
11. Are boards keyed so that they cannot be inserted incorrectly?
12. Have all three types of troubleshooting documentation techniques been considered? These are:

- Symptom cause: list symptoms and possible causes. This technique can be helpful when used with microprocessors that can display error codes when a specific failure occurs.
- Troubleshooting tree: test and branch based on results of test. A good tree requires more development time than any other approach since its developer must convince himself that the person following a fault through the tree will not get lost in a wrong branch.
- Growing the kernel: the instrument is exercised in a series of operating modes arranged in increasing levels of complexity so that each successive operating mode exercises a larger percentage of the instrument. By observing the first operating mode in the sequence that fails, it is possible to quickly limit the problem to those assemblies that are used in the failed operating mode but are not used in the previous modes that all passed.

rithm's operation. At the beginning of the sweep portion, *SP* is displayed, indicating that the instrument is sweeping its internal synthesizer for the desired intermediate frequency signal. When the signal is detected, sweeping stops and a *2* is displayed. Then the intermediate frequency is centered in the passband, the display shows *3*. To indicate determination of the harmonic number (the harmonic of the synthesizer that is mixing with the unknown to produce the intermediate frequency), the display shows *Hd*.

Yet another useful user-callable diagnostic that applies to many products is a keyboard/display check. Such a test gives the serviceman confidence in using the keyboard and display in other diagnostic checks.

To initiate this test, the serviceman hits SET, SET, 8 on the keyboard of the 5342A. The next key pressed fills the display with a unique character associated with that key. For example, pressing the 1 key results in a display reading of all 1s, and so on.

A go/no-go test

A third testing technique that can simplify servicing of microprocessor-based products is mapping, which helps the serviceman decide where to begin troubleshooting. In the 5370A time-interval counter, for example, the first check is of the kernel and associated buses, for they must be working in order to troubleshoot any of the counting circuits, the display, and the analog sections.

Mapping can quickly check the operation of the microprocessor and its communication with other boards in the unit without spending time to take signatures. However, it is a go/no-go evaluation, so when it indicates a bad kernel, signature analysis or some other technique must be used to isolate the failed component.

Essentially, mapping is a picture of the address bus, and the processor generates the picture as it performs its routines. In the 5370A, rather than connect a logic-state analyzer to all 16 lines of the address bus, two digital-to-analog converters on a special service board plug into the bus to supply analog levels to the X and Y inputs of an oscilloscope (Fig. 7). Each dot of the address matrix displayed by the scope represents an individual address. As the processor performs its programmed routines, these dots are connected together to form a map.

The 5370A is designed so that power-on with no input signal results in the map of Fig. 8a. The map appears after the microprocessor executes its power-up diagnostics of RAM, ROM, etc. It indicates that the kernel is working properly, that the data and address buses are clear and functional, and that the counter is waiting for an input signal to start the instrument.

It is important to realize that the serviceman is not looking for each dot of the map; he is interested just in the pattern. There are only 12 maps for the 5370A that he needs to be able to recognize. Documented in the service manual, they include such routines as: measurement in progress, start signal but not stop, free-run, read/write test patterns, trigger-level routine, reset, and display-rate hold.

If the scope shows any other pattern (Fig. 8b), then the processor has entered an undefined routine or is stuck in a portion of the algorithm. If the cause is suspected to be a board holding a line on the address bus low or high, boards can be removed one at a time until the proper map appears on the scope. Otherwise, it's on to signature analysis. □

Free-running signature analysis simplifies troubleshooting

Although the advent of μPs has made designing easier, it has also created a troubleshooting nightmare. Fortunately, a solution exists.

Andrew Stefanski, Hewlett-Packard Co

Failures that cause the complete breakdown of a μP-based product also generate extremely difficult troubleshooting tasks. But of all such challenges the location of a faulty component in the μP's bus structure is particularly troublesome. A failure there causes all components to behave abnormally because the bus forms a complex feedback loop with the microprocessor and memory. A fault anywhere on the bus propagates throughout this loop, resulting in erroneous inputs to each component and, as a consequence, erroneous outputs from each one. Logical in-circuit testing of such a μP-bus system gone wild is practically impossible.

But you can overcome feedback-loop-caused difficulties by eliminating the loop itself. A simple, easily constructed test fixture breaks the loop, enabling the processor to operate in a free-running mode—thus forcing it to behave predictably and allowing you to troubleshoot in an orderly fashion.

In addition to breaking the μP free from its data bus, this test fixture provides a no-operation (NOP) opcode to the bus so that the processor sees a NOP instruction regardless of the contents of the address being fetched. This instruction increments the program counter and causes a fetch of the next instruction (another NOP). Utilizing this technique forces the processor to address the entire memory-address space despite failures in the bus, address decoder or ROM. You can then verify μP operation step by step, checking the address lines with a signature-analyzer probe and comparing the

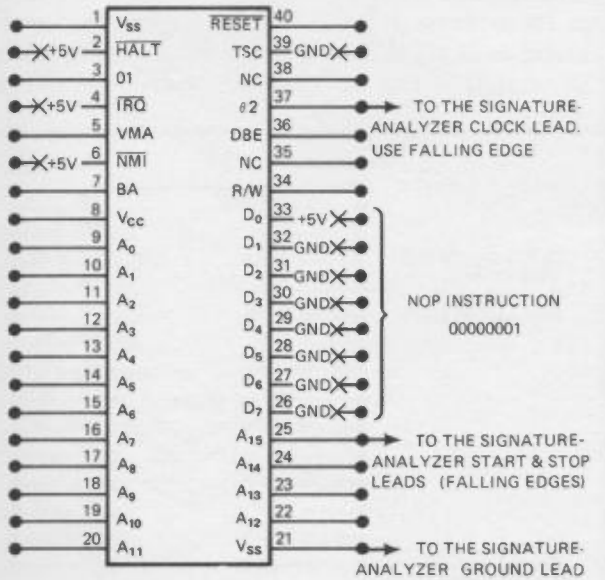


Fig 1—This 6800 free-run test fixture *provides a NOP instruction (0000001) to the data bus to step the microprocessor through its entire address space.*

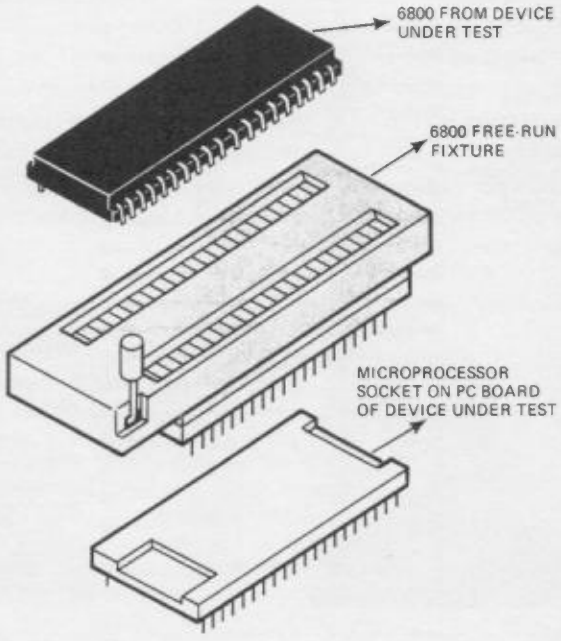


Fig 2—The test fixture *is inserted between the chip under test and its socket on the pc board.*

signatures displayed with correct signatures. Thus, you can utilize the signature analyzer to probe any portion of the circuit where correct inputs and outputs are known.

Preparation for this troubleshooting procedure involves inserting the free-run test fixture into a known good circuit and noting the signatures of all bus lines and nodes of the circuit's address decoders. The address-bus signatures for all μPs discussed in this article are the same (the notation—a modified hexadecimal scheme—is chosen for visual clarity):

A_0	UUUU	A_8	HC89
A_1	5555	A_9	2H70
A_2	CCCC	A_{10}	HPP0
A_3	7F7F	A_{11}	1293
A_4	5H21	A_{12}	HAP7
A_5	0AFA	A_{13}	3C96
A_6	UPFH	A_{14}	3827
A_7	52F8	A_{15}	755U.

In fact, this list is valid for *all* μPs with 16 address bits, so you can use it, in conjunction with the appropriate test fixture, to check μC boards fresh from the prototype shop—even if those boards have never been previously characterized. Misconnected traces, shorts and opens all appear quickly and easily.

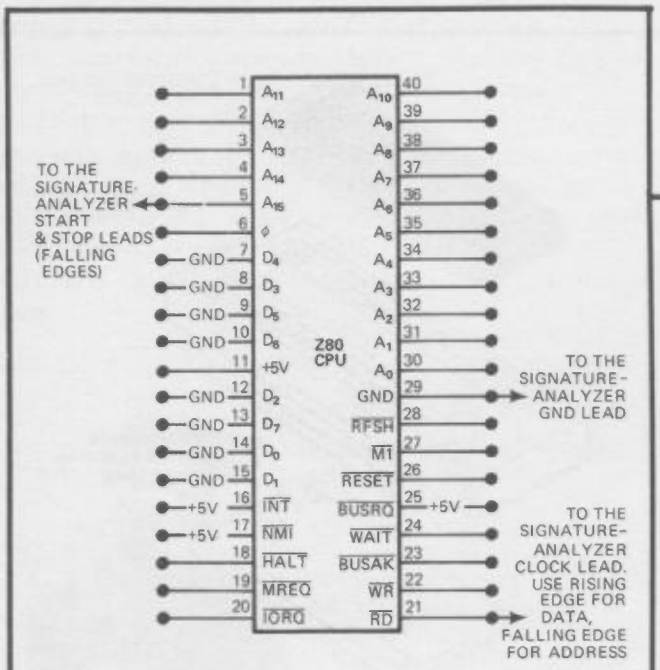


Fig 3—The fixture for the Z80 *resembles that for the 6800; you adjust the pinouts to suit the tested processor.*

Designing the fixture

Obviously, the key to implementing this approach is the design of the test fixture itself: It must provide a method of disconnecting the μP from its data bus and applying the NOP instruction. The exact fixture design required depends on the type of microprocessor used. The theory, however, applies universally.

The test fixture for the 6800 is particularly straightforward. **Fig 1** shows the wiring for a 40-pin socket that is inserted between the μP chip and its own socket (**Fig 2**). The data bus (pins 26 through 33) is tied to ground with the exception of D_0 (pin 33), which is connected to +5V; this approach provides a 00000001 NOP instruction. The ϕ_2 clock (pin 37) connects to the signature-analyzer clock lead—use the signal's falling edge for your trigger. V_{SS} (pin 21) provides signature-analyzer ground. Disconnecting the interrupt lines (pins 2, 4 and 6) and tying them to +5V prevents interference from interrupts during the test.

A fixture for the Z80 (**Fig 3**) utilizes similar techniques; the wiring simply changes to suit that processor's unique pinouts and input requirements for generating a NOP instruction. The 8080, however, requires a somewhat different approach because its status signals are multiplexed with the data; straightforward interruption of the data lines would thus completely prevent its operation. Therefore, the 8080 test-fixture approach (**Fig 4**) involves breaking the data lines on the "outside" side of an 8228/38 system controller chip—the side where the status and data lines are demultiplexed.

Presenting still another problem, the 8085 multiplexes its data bus with the lower eight bits of its address bus. 8085-based systems incorporating conventional memory devices perform the necessary demultiplexing on board, but to accom-

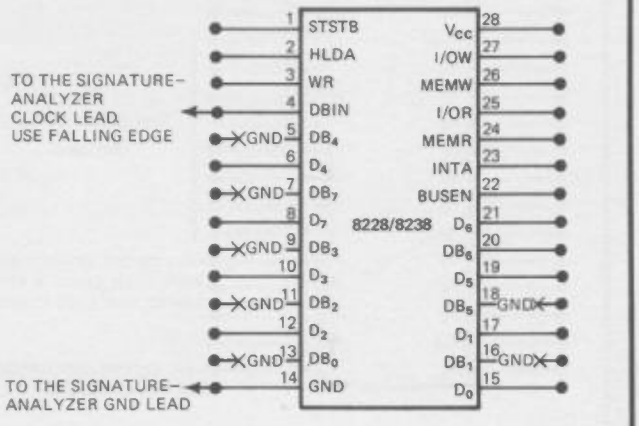


Fig 4—The 8080 fixture, *unlike those for other μPs, goes between the 8228/38 system controller chip and its socket.*

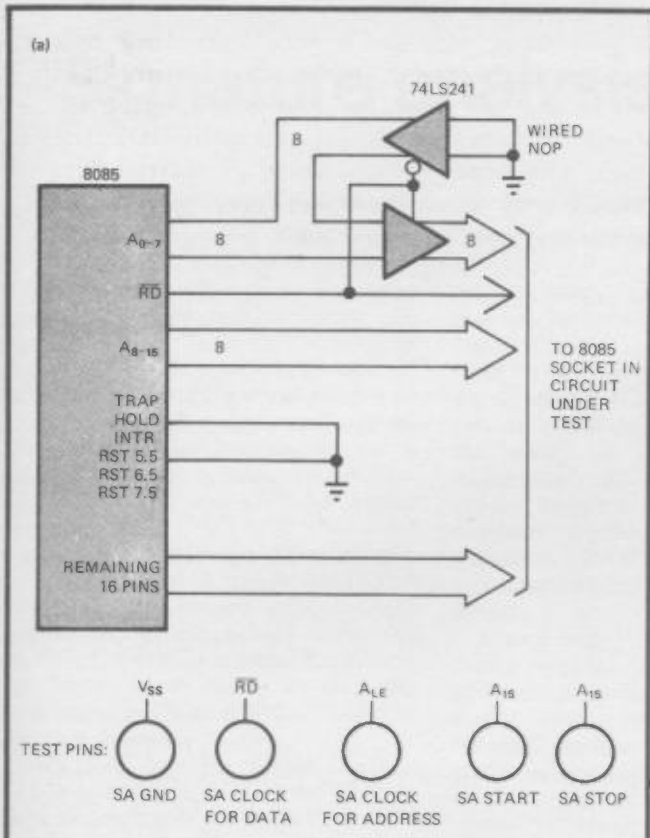


Fig 5—The 8085 *requires a more complex fixture than other processors* ***(a)****. Use a ribbon cable to connect a test board to this fixture and its support circuitry* ***(b)****.*

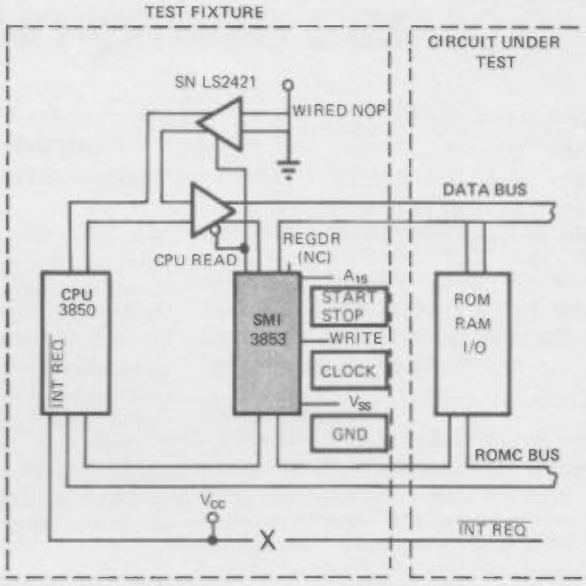


Fig 6—An F8 fixture *might require addition of a 3853 static-memory interface to generate Start and Stop signals for the signature analyzer.*

modate all possible systems, the 8085 free-run fixture uses a bidirectional buffer. This buffer preserves bus integrity when the μP sends an address out; it also breaks the bus and provides the NOP code when data is read in. **Fig 5** shows a schematic of the free-run circuit for the 8085 and its implementation.

Finally, the F8 presents still another problem, because some F8-based systems don't have a visible address bus. Thus, if the tested board doesn't contain a static-memory interface chip (3853), you must incorporate one in the test fixture. Interfacing the 3850 CPU to the data bus also requires buffers so the CPU can use the bus to reset program counters when the system is powered up. The procedure for this μP remains relatively straightforward, however—one test board interfaced through a ribbon cable does the trick.

EDN

Author's biography

Andrew Stefanski is a senior member of the IEEE and project leader at Hewlett-Packard's Santa Clara, CA Instrument Div. He holds a PhD EE from the University of Pennsylvania and an MSEE from Warsaw Polytechnic, Poland.

Signature analysis simplifies service

Repairing retail electronics can be a knotty problem; Hugin found a good solution for its electronic cash registers.

At one time all a cash register repairman needed was a well-stocked toolbox and minimal training. Today, with the rapid acceptance of electronic cash registers (ECR), he still needs that tool kit, but he also needs several complicated, expensive electronic test instruments and a solid background in basic electronics. That means he commands a higher salary, too.

Most ECR dealers are mom and pop operations that can't justify this capital outlay or pay increased salaries. Their only alternative for repairing electronics is a board swapping system. Repairman plug known-good boards in until the ECR works and send the "bad" board to a factory depot for repairs.

With five days for postal handling each way and five days in the depot it takes a minimum of two weeks to get the board back. In addition to the actual repair charges, it costs an average of $50 for each board processed this way. Furthermore, the dealer has to stock spares for **every** board type, including new versions, as manufacturers improve and up their product lines.

The "electronic" tool kit

Cash register dealers are an independent breed; they want to make repairs themselves and they want to make them on-site. This type of market pressure convinced Hugin Kassaregister AB of Stockhold, Sweden to look for a low-cost, minimal-training electronic equivalent of the tool kit. It thinks it has found the answer in signature analysis. Says Jack E. Kleinert, US supervisor of service and education for Hugin Cash Registers, Inc., "All ECR's on the market today use digital logic and Hewlett-Packard's signature analyzer (the HP5004A) is designed to easily troubleshoot digital circuitry. By providing signatures of a good machine in our service manual, a technician needs only the ability to compare two 4-digit 'signatures' and some basic power supply knowledge." He also needs a company supplied test PROM, digital voltmeter (DVM), signature analyzer and accurate pictorial and schematic layouts.

Armed with these items a technician uses a Heathkit-type step-by-step troubleshooting procedure until he finds the faulty component, indicated by a "wrong" signature. With fault isolation at the component level the dealer can eliminate his spare board inventory; a complete repair center could now consist of soldering and unsoldering equipment, components, DVM and a signature analyzer. But even more important, Hugin's records show that with signature analysis a bad component can be found in less than 15 minutes, excluding set-up time.

***Semiskilled workers** can easily repair printed circuit boards by following pictorial diagrams and comparing two 4-digit signatures.*

The bottom line

The dealer must still justify the $1000 cost of a signature analyzer. He won't save on service calls, because the number remains the same no matter how the register is repaired. Shop time also breaks even since signature analysis repair takes about the same shop time as the paper work that accompanies board exchanges.

The big savings, however, come from eliminating the $50 shipping and handling fees to a repair depot. It's also cheaper to stock components than full boards since many components are duplicated on boards. But even ignoring that, a dealer would have to see 20 failures a year to justify the cost of the equipment in one year. With an industry average of one failure per ECR each year, an installed base of just 20 units could easily justify the cost.

The one seeming drawback is the need for a test PROM and "known-good" signature list. Accordingly Hugin, the world's second largest cash register manufacturer by installed base, will design signature analysis into all future ECR's and supply these items. Kleinert thinks other manufacturers will have to follow suit to stay competitive. He also sees some digitally-oriented entrepreneurs adapting signature analysis to those ECR's not expressly designed for it and marketing test PROM's and signature catalogs for the more popular models—***R. Grossman***

For more information, call your local HP sales office or (East) (301) 948-6370 • Midwest (312) 255-9800 • South (404) 955-1500 • West (213) 877-1282. Or write: Hewlett-Packard, 1501 Page Mill Road, Palo Alto, California 94304. In Europe, Hewlett-Packard S.A., 7, rue du Bois-du-Lan, P.O. Box CH-1217 Meyrin 2 Geneva, Switzerland. In Japan, Yokogawa-Hewlett-Packard Ltd., 29-21, Takaido-Higashi 1-chome, Suginami-ku, Tokyo 168.

02-5952-7542

MAY 1979

PRINTED IN USA