Implementing Signature Analysis for Production Testing

with the HP 3060A Board Test System

Application Note 222-1

HEWLETT hp PACKARD

Table of Contents

I. Introduction

The advantages of production testing printed circuit boards with capabilities for functional and component level testing have been realized in recent years as board test systems and techniques have been refined. The testing of low complexity digital circuitry is well served on the HP 3060A Board Test System by a digital testing technique known as static pattern testing. As the digital circuitry's complexity increases (with feedback, multilevel logic, and increased IC count), the programming for this test capability becomes challenging and the time required to thoroughly test a board can sometimes make the cost effectiveness of this technique questionable. Other digital testing techniques, such as transition counting, have been ineffective in consistently and accurately locating defects. This is partially caused by the inability to test a circuit in a manner which detects timing related errors.

What would be most desirable is to apply all possible data to all possible inputs as a truth table type signal set while the circuit runs at its full operating speed. The set need not be organized in an orderly fashion as one usually constructs a truth table on paper; the objective is to apply as many of the possible inputs as is feasible with order of application being unimportant. For all but the simplest digital circuits, checking the outputs and intermediate circuit states for such an input set would be impossible by conventional means. With signature analysis, however, a multitude of intermediate and output conditions can be quickly and accurately analyzed. There are two keys to this ability: 1) high speed circuit exercising and 2) data comparison.

When using signature analysis, many input combinations are applied to the unit under test (or are generated by the unit under test) at the circuit's full operating speed, up to 10 MHz. Then responses, at one node, to all of these inputs are compressed into a four-character hexadecimal signature. The four-character hexadecimal display has a total of 12^{16} (16^4) possible states, which in conjunction with the 16 bit input shift register (which also has 2^{16} possible states) and associated feedback provides a unique "compressed" representation of a data stream.

For testing prototype digital devices and for field service of digital devices, the HP 5004 portable Signature Analyzer is a valuable tool. For production testing of digital printed circuit boards, the signature analysis option to the 3060A Board Test System offers simple, quick, and accurate testing of complex digital printed circuit boards.

II. Operation and applications of signature analysis

One of the main advantages of signature analysis over other digital testing techniques is its ability to detect timing related errors. In order to have this ability, the data stream to be checked must be input to the analyzer in a synchronous manner and since we wish to perform this test at the circuit's full operating speed, we make a connection from the system clock to the signature analyzer. A clock phase or control signal for which the testpoint data is stable is chosen for this purpose. The choice of the clock connection point is simplified by the ability to select whether data is accepted on the rising or falling edge of the clock signal. For asynchronous devices, the signature analyzer's clock input will be connected to the clock of the circuit which generates the input signal set (more on this under "Testing ROMs and Combinational Circuits").

While accurate signatures can be obtained by exercising a circuit with a single finite length data stream, it is more desirable to let the circuit be exercised in a continuous loop. We may then read a signature for any one of these complete cycles. In order to perform this function, it is necessary to be able to tell the analyzer what data of the infinite cyclic stream to read for the determination of each signature. This is accomplished by start and stop connections made between the unit under test and the signature analyzer. These two connections set the "time window" during which the analyzer reads data. The choice of these connection points is simplified by allowing the start and/or stop to occur after a high to low or low to high transition of these signals.

The last connection between the unit under test and the analyzer is the analyzer's data input which is programmed to different circuit nodes to check their characteristic signatures. The determination of whether a signature is good or bad is then made by comparing the signatures of the board being tested to the signatures of a known good board (with these known good signatures being stored within the system as demonstrated in Appendix E). Bad signatures may then be traced to the source of a fault by programming the analyzer's data input to various circuit nodes.

The analyzer's data input is fed to a 16 bit shift register with feedbacks. It is very similar to a register used in some pseudorandom code generators. The reason for this stems from information theory which says that the shift register contains a maximum amount of information when all of its possible states occur with equal probability. When this condition is met, the shift register takes on one of 2^{16} possible values with a probability of being in this state of $1/2^{16}$. For input streams ranging from about 16 bits to beyond 2^{16} bits this condition is approached and measured signatures can be compared to correct signatures with the excellent assurance that if the data is correct the measured signature will match the correct signature.* For the reader not previously familiar with the operation of a signature analyzer an illustrative example of its operation is given in Appendix A.

The features of signature analysis provide a flexibility which allows it to be easily applied to many digital devices from combinational circuits to individual ROMs to microprocessor-based boards. As an integral part of the 3060A Board Test System, signature analysis offers simple, accurate and flexible production testing of digital printed circuit boards.

*For a further look at signature analysis accuracy, see "Electronics Magazine", March 3, 1977; Vol. 50; No. 5; p 91.

III. General SA testing techniques

Signature Analysis testing relys on the principle of "exercising" or wiggling circuit nodes - that is forcing a node to change states. The manner by which this is accomplished is relatively unimportant. For example, a counter can generate a truth table type input signal set which is a good exercise for combinational circuits and ROMs. Or if a part of the circuit is designed as a self-test for the final product, this makes a good exercising routine. For exercising microprocessor-based boards, the address bus can easily be made to continuously cycle through its entire address field, exercising a good portion of the board's circuitry. In addition, simple test programs written into ROM may be used to more thoroughly test MPU-based boards.

Once a means of exercising the board is implemented, signatures can be taken at circuit nodes and a bad signature can be traced to its source analogous to the way one traces a bad waveform to its source in an analog circuit. This analogy also applies in relation to testing circuits which employ feedback in that the feedback should be disconnected if one wishes to fully isolate the source of an error. While not all digital circuits contain feedback, sequential circuits, as well as MPU-based boards (with their data busses and interrupts), do have feedback to be disconnected for the purpose of diagnosing faults to the component level.

As with analog signal tracing, several fault isolation techniques exist for digital signature tracing. Expanding the kernel is one technique which is analogous to tracing a test signal from its point of application toward the output. As the testpoint is moved away from the input, there will be a point where the signatures will change from good to bad. The faulty component is thus determined to be one of the components connected to the point where the first bad signature is encountered.

Half-splitting is another technique for fault isolation in which the board test programmer chooses a point in the circuit where failure ahead of or behind this test point is approximately equally likely. In this way, the bad half is determined and then split in half again and again until the fault is isolated.

Expanding the kernel is especially useful for field testing since starting at the input end of a circuit where the probability of component failure is low provides the user with high assurance that the test equipment is set up properly. Half-splitting, on the other hand, is a more efficient technique and therefore recommended for board test programming.

One important difference between analog signal tracing and digital signature tracing should be noted. When tracing waveforms in an analog circuit, one gets clues as to the cause of a fault from waveshape, such as a clipped sinusoid or 60 cycle hum superimposed on the test waveform. When tracing signatures, however, subtle differences between signatures are not useful in the same way that differences in waveshape are useful for analog analysis. If one bit of a long data stream is in error, it is very likely that all four characters of the resulting signature will be different from the characters of the correct signature (this can be seen by examining the examples given in Appendices A & B). The particular characters of a bad signature are, therefore, of little use except for conveying to the tester whether they agree with the correct signature or not and thus whether the signal at the node under test is correct or in error. The SA data input may then be programmed to different circuit nodes with each measured signature telling the user where the fault is in relation to the node being tested.

There are a few special cases where the signature tells the user more than whether the signal at the node under test is good or bad. One case is 0 0 0 0 signature which either indicates a stuck logical 0 or a signal that is low prior to transition of each clock pulse. Another case occurs when a bad signature for one node matches a bad signature for a different node on the board. In this case, there is a good chance that these two points are shorted together, although this condition might be diagnosed by the 3060A's shorts test.

One last signature which conveys information other than whether the signal at the test node is good or not is the V_{cc} signature. When the analyzer's data input is connected to V_{cc}, a continuous string of ones are input to the analyzer's shift register and the resulting signature will take on different values dependent upon the length of the time window (see Appendix B for an illustration of this). Since improper setup of the SA inputs and triggering edges would most likely cause the time window to change, this signature gives information related to the correctness of the start, stop, and clock connection points and triggering edges.

In addition SA tests can be implemented such that the time window is one length for proper circuit operation and a different length (even one clock cycle longer is a sufficient difference) for improper circuit operation. A V_{cc} signature would then differentiate between a good and a faulty circuit.

IV. Testing ROMs and combinational circuits with SA

Combinational circuits and individual ROMs (i.e., ROMs which at this stage of production have not been interfaced with other circuitry) contain no feedback paths.* Because there is no feedback, these circuits can be fault isolated to the component level without the need to design any special signature analysis test capabilities into each printed circuit board. To exercise these circuits, there are several options. If there is circuitry on the board which was designed to be a self-test for the final product, this may be used as an exercise for SA production testing.

There is another exercising possibility which exists if the board to be tested interfaces with another board in the final product which contains a self-test routine. In this case, the board containing the test routine can be mounted on the 3060A's test fixture and connected such that it can exercise each unit under test.

Even if these self-test routines are only intended to provide a go/no go indication for the final product when they are used in conjunction with SA techniques, faults can be isolated to the component level.

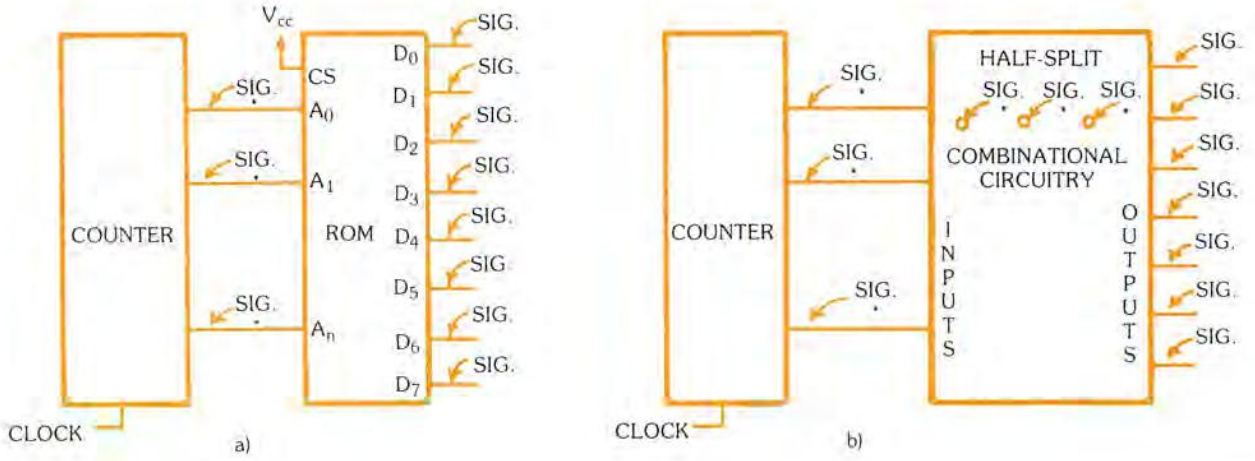
If self-test capabilities were not designed into the product, a counter built into the 3060A test fixture clocked from one of the 3060A's clocks can generate a signal suitable to exhaustively exercise these devices. The number of counter bits required will be equal to the number of inputs for combinational circuits and the number of address bits for ROMs.

The counter input can be applied to a ROM's address inputs and signatures can be taken at the ROM's data output as shown in Figure 1a. If one of these signatures is in error, it is a good practice to have the test program then take signatures on the address inputs of the ROM to determine whether the counter is working and if the ROM inputs are good.

All of these signatures can be taken with the analyzer's start and stop inputs connected to the MSB of the address input to the ROM.

In a similar manner, a counter built into a 3060A fixture can be applied to the inputs of a combinational circuit as shown in Figure 1b. The clock signal may be generated by the 3060A and the analyzer's start and stop connections are made to the MSB of the counter. Signatures may then be taken at the circuit outputs. If one of the output signatures is in error, the board test program can be written to isolate the fault by half-splitting.

*Combinational circuitry which has feedback is classified as sequential circuitry since its outputs are dependent on past inputs.



*If any output signatures are in error, check these points to isolate a fault to the component level.

Figure 1 - Connection of a counter to a) a ROM and b) a combinational circuit, to provide an exercise for signature analysis

V. Testing sequential circuits with SA

Designing with SA in Mind

Sequential circuits are devices for which the output is dependent on past inputs, as well as present inputs. Implementations usually employ feedback which should be disconnected for SA testing if more information is desired than a go/no go indication. If feedback is not disconnected, faults could propagate from their source through the circuit, and then through feedback they could be introduced at the inputs. When this occurs, fault isolation is no longer possible since signatures ahead of, as well as behind, the faulty component would be in error.

When a board is in the early design stages, provision should be made for the disconnection of feedback during testing. This may be implemented with switches, jumpers, or a tristate buffer inserted into the feedback path(s) as shown in Figure 2. Switches and jumpers have the advantage of being positive disconnecions and the disadvantage of requiring extra circuit handling. A tristate buffer could be employed such that an input driven from a 3060A driver can put the buffer into its high impedance state. The disadvantage of this scheme is that if the buffer should fail, signals could be present at the feedback inputs. Thus, if this method of feedback disconnection is employed, it is recommended that an early part of the test program check the buffer to see that none of its outputs are stuck high or low (this could be done with the 3060A's static test capabilities). Once feedback paths have been disabled, a sequential circuit is basically a combinational circuit and can be tested as such.

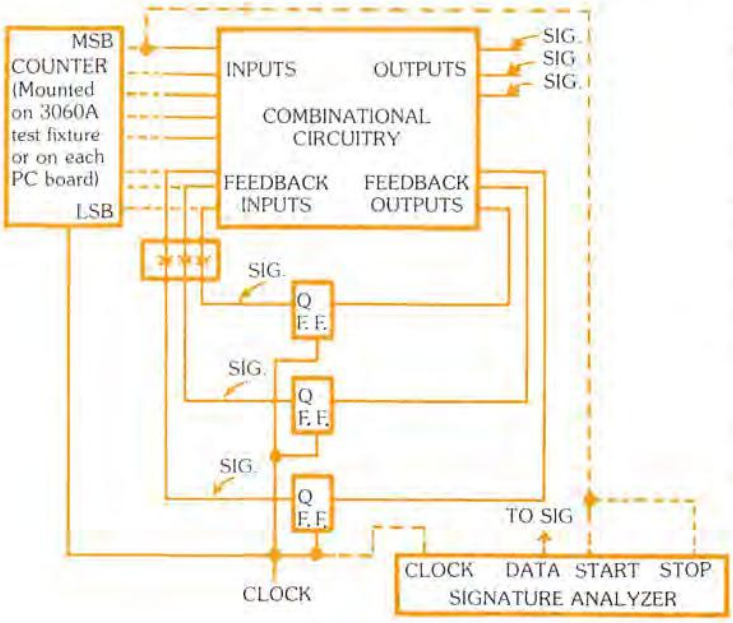


Figure 2 - Method for testing synchronous sequential circuits. Feedback disconnections are shown by X. Connections which are made for the purposes of SA testing are shown with ----lines.

If a synchronous sequential circuit is being tested, the circuit's own clock can be used to clock a counter built into the 3060A test fixture which can then be connected to the circuit inputs (including the now disconnected feedback inputs).

If an asynchronous sequential circuit is being tested, the counter can be built into the test fixture and clocked from a 3060A clock. In both cases, the signature analyzer's start and stop inputs may be programmed to the most significant bit of the counter and signatures can be taken at the circuit outputs. If a bad signature is found, half-splitting may be used to isolate the fault.

Retrofitting

When given a sequential circuit which was designed without signature analysis testing in mind there are several testing options. The most desirable option is to add a method for disconnecting any feedback path(s) and then test the board as described above. If this is not a feasible option, it might be possible to mount the feedback flipflops of a synchronous sequential circuit in a socket so that they can be removed during testing. One disadvantage to this method is that a component must be removed from the board and replaced again after testing. This requires increased testing time and increases chance of damage to the removable component.

In some cases, feedback signals might be logical 0 for a specific number of clock cycles. If this is the case, a feature of the 3060A's signature analyzer called stop on count may be used as a software means of feedback disconnect. Stop on count is a feature of the 3060A's signature analyzer in which the stop signal that ends the time window, occurs a preselected number of clock cycles after the start signal. This could therefore be used to end the time window just before any signals are fed back.

If none of the above choices is possible, signatures can still be taken to determine whether the board is good or faulty, but fault isolation capabilities have been sacrificed.

VI. Testing microprocessor-based boards with SA

Disconnecting feedback paths

When designing a microprocessor-based board which will be tested with signature analysis, it is desirable to design a method of feedback disconnect into the circuit to aid in fault isolation. There are ways in which a MPU-based board can be tested (and even partially fault isolated) without feedback disconnect provisions; but since there are other drawbacks to these methods, consideration of them is deferred to a discussion of retrofitting.

Interrupt lines and the data bus input to a microprocessor are feedback paths for which one of several disconnect methods should be chosen. These methods vary in terms of board space required, reliability of operation, and simplicity of use. Choosing the most suitable method is, therefore, dependent on the particular application and its associated restrictions. Generally, most SA test capabilities can be built into the 3060A's fixture (counters, a PROM with SA test programs, etc.). As more testing related implementation is put onto the test fixture (as opposed to being employed on each individual printed circuit board), ease of field testing is reduced.

A DIP jumper or switches are the most positive means of disconnecting the data bus input to the processor. An alternative which can be used if a tristate buffer has been added to the data bus for other reasons (such as a heavy bus load) is to use this buffer to disable the data bus feedback path. By adding an and gate to the buffer's disable input, the buffer can be driven to its high impedance state with a 3060A driver. Then, when being field tested, the buffer may be disabled with a jumper wire or with a switch.

If there is no buffer for other reasons one could be added specifically for the purpose of feedback disablement. In either case, there is a chance that the buffer output could be stuck or not obey the disable instruction. It is, therefore, recommended that the 3060A test program check the data bus to see that the buffer outputs are not stuck high or low.

Interrupt lines are most easily disconnected by an interrupt masking instruction. Or, if desired, they could be disconnected directly with a jumper or switch connected to the processor's interrupt input.

Exercising the board

Once these feedback paths have been disconnected, we want to consider how to exercise the circuit. The simplest way to do this is to force the MPU to free-run.

In relation to microprocessors, this involves getting the processor to continuously cycle through its entire address field.* This can be accomplished by applying an instruction to the processor's now disconnected data input which causes the processor's program counter to increment. The processor will then check its data bus where it will again see the "free-run" instruction and again the program counter will increment, continuing the cycle. In this way the processor's address bus scans its entire address field, thus applying a truth table type signal set to the address bus.

The choice of what instruction will be applied to the processor to get it to free-run is not critical - basically any instruction which causes the program counter to increment and the processor to thus look for the next instruction will suffice. Some instructions, however, are easier to implement than others. The application of a free-run instruction can be accomplished directly with drivers on the 3060A so that no hardware needs to be added to the circuit. For increased field testability, however, a few parts may be added which can be used to force the free-run condition in the field. As shown in Figure 4 "pull up" resistors are permanently connected from each of the data bus lines to the V_{cc} supply. Then diodes are connected from the appropriate data lines through a "free-run switch" to ground. These connections are made on the side of the data bus disconnect which is toward the processor. Then when the data bus is opened and the switch grounding the diodes is closed, the free-run instruction is applied to the MPU's data input. Implementations of this for several different microprocessors are given in Appendix C.

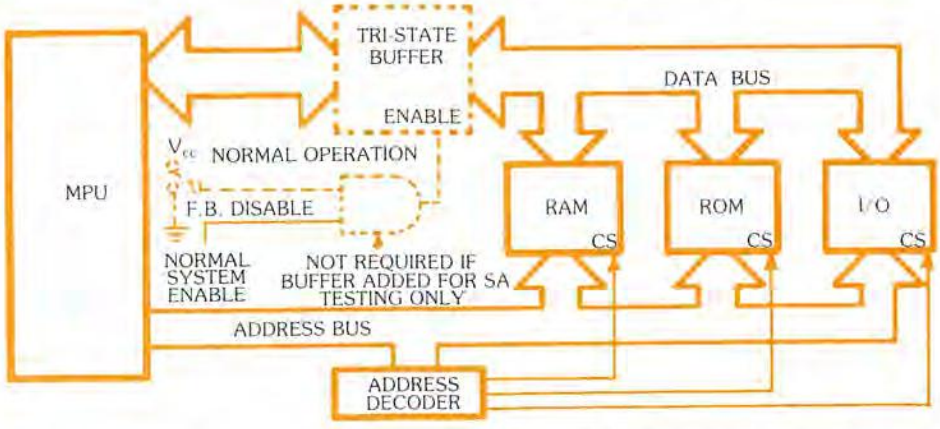


Figure 3 - Generalized microprocessor based circuit which employs a buffer in the data bus as a means of feedback disablement.

*For a more general definition of free-running see HP Application Note 222, p6.

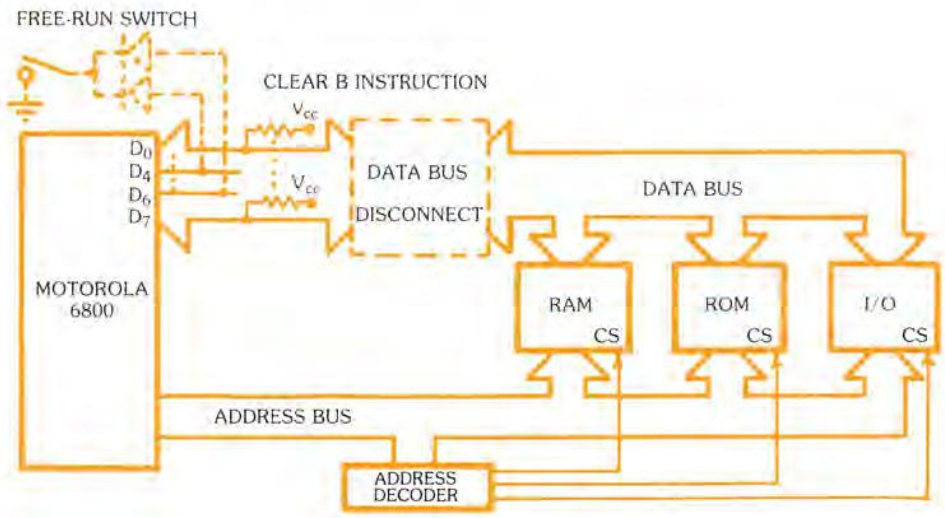


Figure 4 - Method for free-running a microprocessor which can be used for both field and production testing.

Output ports are exercised by free-running. Signatures can therefore be taken at these ports and on their peripheral circuitry. Input ports, however, are not exercised by free-running. One method for exercising these ports is to tie outputs back to inputs with 3060A relays.

The major parts yet untested are the RAMs. These may be tested by now reconnecting the data bus to the processor. Then a simple test program written into a small portion of ROM (or in a PROM on the test fixture) can write data into RAM and read it out again. Checking the data bus signatures for the read cycle of this exercise verifies the operation of the RAM.

There are many RAM test programs designed to verify different aspects of a RAM's performance. Here are a few examples.

1. Store the contents of ROM into RAM. This provides a somewhat random-check on the addressing ability of the RAM and the operation of its storage locations.

2. Store a checkerboard pattern of 1's and 0's into RAM and then store in the complement of this pattern. This checks every RAM storage location to see that they each have the ability to enter a logic 0 and 1 state.

3. Walk a 1 through every RAM location. This checks to be sure that no storage locations are stuck and, in addition, it is a thorough test for shorts between locations.

4. Store each RAM address in that addresses memory location. This test assures the ability of the RAM to properly address every storage location.

Example software for each of these tests is given in Appendix D.

Once the processor is free-running, the board's operation can be simply checked with the techniques of half-splitting or expanding the kernel. Expanding the kernel will be used here for illustrative purposes although half-splitting might be preferred to increase testing efficiency. The processor's addressing field may be checked by programming the signature analyzer's start and stop inputs to the address bus's MSB while checking signatures on all of the address lines. With these same start, stop, and clock connections, signatures may be taken at the address decoder outputs to check its operation.

The start and stop connections may then be changed to shorten the time window so that the analyzer only inputs data to its shift register during a portion of the address field corresponding to a single device. This may be implemented by connecting the start and stop inputs to the chip select of the individual device to be tested. If this is done and the analyzer's data input is programmed to the data bus (on the side of the feedback, disconnect away from the processor), the contents of each ROM is verified.

Another method for "windowing in" on a bad ROM is to make use of a technique called stop on count. As mentioned earlier, stop on count is a feature of the 3060A's signature analyzer in which the stop signal that ends the time window, occurs a preselected number of clock cycles after the start signal.

As before, the analyzer's start input is connected to the MSB of the address bus. The end of the time window may then be varied, using stop on count, such that the analyzer's data input accepts data for various portions of the address field. Since RAMs could contain unpredictable data, it is important to either not include RAMs within the time window or disable the RAMs (possibly by the addition of some logic to the address decoder such that a 3060A driver could disable them) or fill the RAMs with known patterns.

Retrofitting boards for SA

When retrofitting a microprocessor-based board for signature analysis testability there are several considerations to be made. Methods of feedback disablement and circuit exercising should be chosen.

Devices which were not designed with signature analysis in mind are often designed modularly such that the microprocessor and a few individual components are located on a separate board from ROM and RAM. In this case, the data bus input to the processor has already been disconnected. The processor board can then be tested by free-running the processor (this can be forced from drivers on the 3060A) while taking the signatures of the address bus, and other devices on this board.

RAMs located on boards separate from the processor can be tested by mounting the processor board on the test fixture. A PROM containing the SA test program could also be included on the test fixture. RAM boards are thus tested under the control of the microprocessor with which it will interface in the final product.

ROM boards could also be tested under processor control by mounting a processor board on the test fixture, or they could be tested as described in the section on testing ROMs by mounting a counter on the test fixture.

If the board to be retrofit was not modularly designed there are several other options.

To disconnect the data bus input to the processor one could add either a DIP jumper, switches, or a tristate buffer to the bus. The tristate buffer has the advantage that it can be driven to its high impedance state from a 3060A driver. This driver can be controlled by the test program, thus saving production testing time over other disconnect means.

If there is not board space to add one of these disconnect methods, a few single wire jumpers can be added to the address decoder such that when these jumpers are pulled, all bus-connected devices are put into their high impedance state. This method limits free-run testability to the processor's address counter alone, since the address decoder and ROMs are now disabled.

Once the processor's operation is verified by this method, other components may be tested (and to a certain extent fault isolated) by software signature analysis routines written into memory which are executed after reconnecting the address decoder jumpers. This is one instance where a fault may be partially isolated without the processor's data bus input being disabled. If a checksum is performed on each ROM in such a way that an unused address bit toggles at the start and again when an incorrect checksum is found, a variable length time window may be created for the signature analyzer. If the analyzer's data input is then programmed to V_{cc}, the signature will be different depending on the length of the time window and thus depending on which ROM has failed. (See Table 1)

For the greatest field testability, SA test programs should be written into a ROM which is already on the board. Often the space at the end of a ROM already on the board is enough for a thorough test program. If this is not the case, a separate ROM might be added to the board.

If there is not enough space for this, there are several other options. For example, a PROM (which contains an SA test routine) could be included on the 3060A fixture.

Another option which could be cost effective for some users is to store an SA test stimulus program in a logic pattern generator, such as the HP 8170A.

The 8170A is a programmable word generator in which an SA test routine could be stored and then executed by connecting its outputs to scanner relays on the 3060A. Once the program has been entered into the 8170A it can be used to directly drive a disabled data bus. In this way a microprocessor-based board may be exercised at speeds up to 2 MHz, providing the 8170A is located close enough to the scanner such that cable capacity is not excessive.

Table 1

Motorola 6800 Code for performing a checksum on two ROMs ("ROM1" and "ROM2"). Address line A15 is toggled at the start. A variable length time window is created by again toggling A15 when an incorrect checksum is found, or when the test is completed. The V_{cc} signature will take on one of only three possible values depending on whether 1) ROM1 is bad, 2) ROM2 is bad, or 3) both ROMs are good.

MEM. ADDRESS LOCATION	LABEL	MNEMONIC	LOCATION OR #	EXPLANATION
3800	CSR1	FCB	$D2	Check sum of ROM1
3801	CSR2	FCB	$8C	Check sum of ROM2
	STSP	EQU	$8000	Label used to toggle A15 for SA start and stop
	BESA	STA A	STSP	
		LDX	#$3801	Load index register with the starting address of ROM1
		CLR A		Clear accumulator for checksum
	ROM1	ADD A	0,X	Add contents of ROM to Accumulator A.
		INX		Increment index register to check next location
		CPX	#$4000	Compare the index register with the end of ROM1.
		BNE	ROM1	Branch to "ROM1" if all ROM1 locations have not been checked
		STA A	$00	Output checksum onto data bus
		CMP A	CSR1	Compare accumulator A with ROM1 checksum
		BNE	BESA	Branch to "BESA" to end time window if not equal
	ROM2	LDX	#$4000	} Same as above except ROM1's are changed to ROM2's
		CLR A		
		ADD A	0,X	
		INX		
		CPX	#$4200	
		BNE	ROM2	
		STA A	$00	
		CMP A	CSR2	
		BNE	BESA	
		NOP		Extend time window one step
		BRA	BESA	If both checksums are correct then end time window

VII. Summary

The versatility of signature analysis allows it to quickly and accurately isolate digital faults to the component level. As an option to the 3060A Board Test System, SA is particularly flexible in that it is especially easy to implement. SA testing is further simplified with the aid of 3060A capabilities and features such as programmable drivers, stop on count, and the ability to mount components or PC boards on the test fixture.

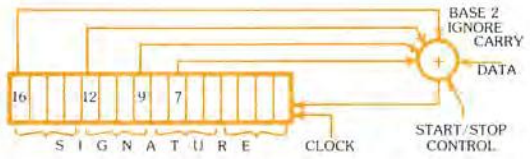
An average of incremental design and development costs for designing SA testability into five HP products was about one percent while incremental material costs were less than one percent. The additional time required to program signature analysis fault isolation tests on the 3060A is also a very small fraction of the total product development.

When one considers the fault isolation capabilities and extremely consistent characterization of good and faulty devices, these costs are recovered many times over for normal use of the 3060A.

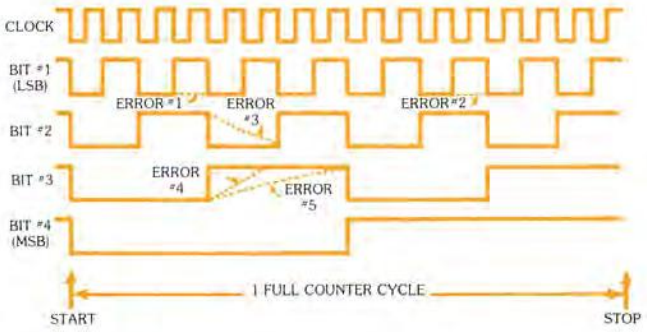
VIII Appendices

Appendix A
How the Analyzers Shift Register Works

Shown below is a model of a signature analyzer.



For illustrative purposes let us take a four-bit counter and look at some signatures associated with its four outputs. Then, let us see what these signatures would be if possible errors (shown ----lines) are introduced.



If the analyzer's clock input is connected to the clock signal shown, and if the analyzer's start and stop connections are made to the MSB of the counter, the time window (i.e., the portion of the data stream used for determination of each individual signature) will consist of one full counter cycle. This is true providing that the following triggering edges are selected: ⌟ for clock, and ⌝ for start and stop.

Several signatures we might check are:

a) Bit 1 with no errors
b) Bit 1 with error #1 introduced
c) Bit 1 with error #2 introduced
d) Bit 2 with no errors
e) Bit 2 with error #3 introduced
f) Bit 3 with no errors
g) Bit 3 with error #4 introduced
h) Bit 3 with error #5 introduced

For instructive purposes we will derive the signatures for signals a and b. This derivation is shown below with data input "a" shown first while the changes error #1 introduces (to produce signal b) are shown shaded.

CLOCK CYCLES AFTER "START"	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	DATA INPUT
1																0	0
2															0	1	1
3														0	1	0	0
4													0	1	0	1*	1*
5												0	1	0	1*	0	0
6											0	1	0	1*	0	1	1
7										0	1	0	1*	0	1	0	0
8									0	1	0	1*	0	1	0	1	1
9								0	1	0	1*	0	1	0	1	1	0
10							0	1	0	1*	0	1	0	1	1	1	1
11						0	1	0	1*	0	1	0	1	1	1	0'	0
12					0	1	0	1*	0	1	0	1	1	1	0'	1	1
13				0	1	0	1*	0	1	0	1	1	1	0'	1	0'	0
14			0	1	0	1*	0	1	0	1	1	1	0'	1	0'	0	1
15		0	1	0	1*	0	1	0	1	1	1	0'	1	0'	0	0	0
16	0	1	0	1*	0	1	0	1	1	1	0'	1	0'	0	1*	1	1
RESULTING SIGNATURE			5*				5				H*				1*		

In a similar manner, signatures for input signals a through h can be determined. Shown below are these input data streams and the resulting signatures. Changes caused by the errors shown in the counter timing diagram are again shown shaded.

CLOCK CYCLES AFTER "START"	SIGNAL PRESENTED TO DATA INPUT OF ANALYZER							
	a Bit 1 No Error	b Bit 1 Error #1	c Bit 1 Error #2	d Bit 2 No Error	e Bit 2 Error #3	f Bit 3 No Error	g Bit 3 Error #4	h Bit 3 Error #5
1	0	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0
3	0	0	0	1	1	0	0	0
4	1	1*	1	1	1	0	0	0
5	0	0	0	0	0'	1	1*	1*
6	1	1	1	0	0	1	1	1*
7	0	0	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	0
11	0	0	0	1	1	0	0	0
12	1	1	1*	1	1	0	0	0
13	0	0	0	0	0	1	1	1
14	1	1	1	0	0	1	1	1
15	0	0	0	1	1	1	1	1
16	1	1	1	1	1	1	1	1
RESULTING SIGNATURE	55H1	45U8	55F1	334U	3C5C	0U16	0702	0308

Note that minor errors in data cause major differences in the resulting signatures. Note also (from a, b, and c) that the same type of error occurring at different times results in very different signatures. In addition, from signals f, g, and h it is apparent that the same error occurring with greater severity (h) causes a different signature from the signature which results with the same type of error of lesser severity (g), and that they both are different from the correct signature (f).

Appendix B

Shifting a Continuous String of Ones into the Analyzer

Clocking a continuous string of 0's into the analyzer's shift register will always result in a 0 0 0 0 signature. Clocking a continuous string of 1's into the analyzer (by connecting the data probe to a V_{cc}), however, does not always result in the same signature. Feedback from within the analyzer's shift register (as shown in Appendix A) assures this "randomizing" of non-zero input data so that a maximum amount of information is conveyed in a signature. Because of this fact, connecting the analyzer's data input to logic 1 will check the time window and, thus, the start, stop, and clock connection points and triggering edges. If one of these is connected incorrectly, the signature will most likely be different than the signature which would result if the proper connections were made.

Shown below are the signatures which result from shifting one through fifty 1's into the analyzer. Note that all 50 signatures are different and that one extra or one fewer clock pulses than intended causes a signature different from the desired signature. In fact, any one signature will not repeat until 65,536 clock cycles later. Note also that in this case, the pattern present at the analyzer's data input is far from random (i.e., constant logic 1) yet soon after 16 clock cycles, the signatures appear random. In practice, much greater than 16 bits will be shifted into the analyzer such that the state which the shift register is in when the "stop" signal occurs will be occupied with a probability of approximately $1/2^{16}$.

# of 1's Clocked In	Resulting Signature	# of 1's Clocked In	Resulting Signature
1	0001	26	FFP5
2	0003	27	99FA
3	0007	28	3395
4	000U	29	672A
5	001U	30	FP54
6	003U	31	9FA8
7	007U	32	3951
8	00UP	33	72A2
9	01UF	34	P545
10	03U9	35	FA8A
11	07U3	36	9515
12	0UP7	37	2A2C
13	1UFP	38	5456
14	3U9F	39	A8AF
15	7U39	40	5159
16	UP73	41	A2C3
17	UFP6	42	4566
18	U9FF	43	8AFH
19	U399	44	159A
20	P733	45	2C34
21	FP67	46	5669
22	9FFP	47	AFH2
23	399F	48	59A4
24	7339	49	C349
25	P672	50	6692

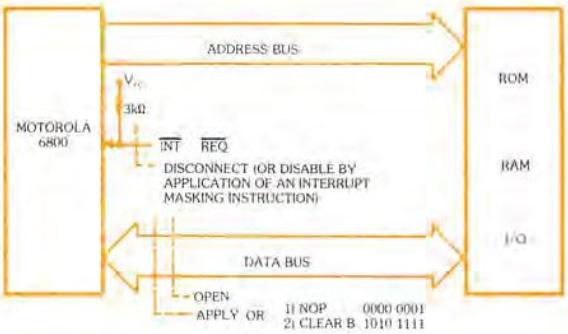
Appendix C

How to Free Run Several Different Microprocessors

Free-Running The Motorola 6800

The free-run instruction may be applied by using pull-up resistors and diodes (as shown in Figure 4), or by use of 3060A drivers.

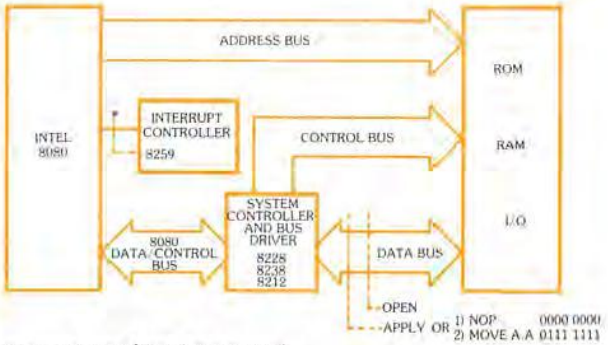
START		STOP		CLOCK		
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	TAKE SIGNATURES ON:
A15	↓	A15	↓	0_2	↑	Address Bus
A15	↓	A15	↓	0_2	↑	Any wiggled node which is stable during ↑ of 0_2.
ROM C.E.	↑	ROM C.E.	↓	0_2	↑	Data Bus (To isolate individual ROM)



Free-Running The Intel 8080

The free-run instruction may be applied by using pull-up resistors and diodes or by use of 3060A drivers.

START		STOP		CLOCK		
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	TAKE SIGNATURES ON:
A15	↓	A15	↓	DEBIN	↓	Address Bus
A15	↓	A15	↓	STATUS STROBE	↓	8080 Control Lines
A15	↓	A15	↓	DEBIN	↓	Any wiggled node which is stable during ↑ of DEBIN
ROM C.S.	↑	ROM C.S.	↓	DEBIN	↓	Data Bus (To isolate individual ROM)
A15	↓	A15	↓	DEBIN	↓	8028 Control Bus

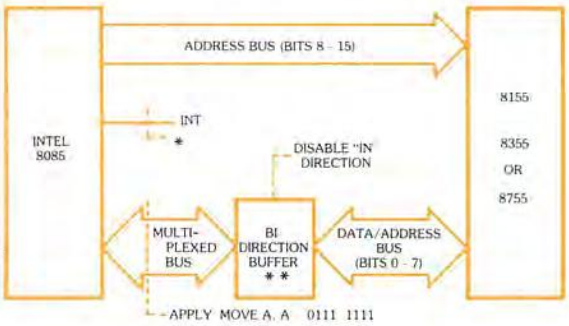


*Disconnect (or disable by application of an interrupt masking instruction).

Free-Running The Intel 8085 With Special Memory I/O Chips

The free-run instruction should be applied with the use of 10k pull-up resistors except for a fast diode pull-down from AD_{7} to RD.

START		STOP		CLOCK		
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	TAKE SIGNATURES ON:
A15	⅂	A15	⅂	ALE	⌐	Address Bus
A15	⅂	A15	⅂	ALE	⌐	Any wiggled node which is stable during ⌐ of ALE
ROM C.E.	⌐	ROM C.E.	⅂	RD	⌐	Data/Address Bus



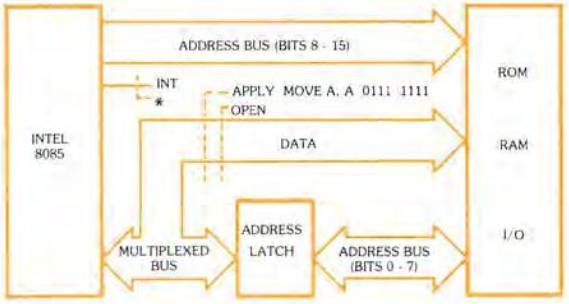
***Disconnect (or disable by application of an interrupt masking instruction).**

****T.I. 74LS245, or T.I. 74L5 243, or National 81LS95; or plug-in jumper for normal operation and "out" only buffer for SA testing.**

Free-Running The Intel 8085 With Conventional Memory And I/O Chips

The free-run instruction should be applied with the use of 10k pull-up resistors except for a fast diode pull-down from AD_{7} to RD.

START		STOP		CLOCK		
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	TAKE SIGNATURES ON:
A15	⅂	A15	⅂	RD	⌐	Address lines (outboard of latch)
A15	⅂	A15	⅂	RD	⌐	Any wiggled node which is stable during ⌐ of RD
A15	⅂	A15	⅂	ALE	⌐	Address lines (inboard of latch)
ROM C.E.	⌐	ROM C.E.	⅂	RD	⌐	Data lines (To isolate individual ROM)

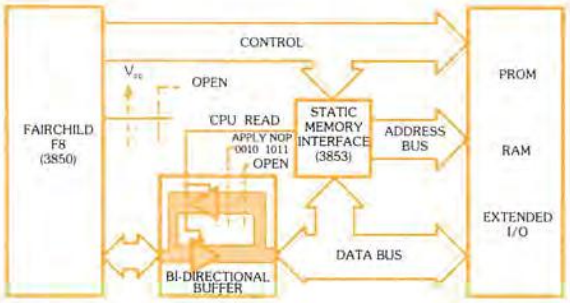


***Disconnect (or disable by application of an interrupt masking instruction).**

Free-Running The Fairchild F8 With External Static Memory

The free-run instruction may be applied by using pull-up resistors and diodes, or by use of 3060A drivers.

START		STOP		CLOCK		
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	TAKE SIGNATURES ON:
A15	⌐	A15	⌐	WRITE	⅂	Address Bus
A15	⌐	A15	⌐	WRITE	⅂	Any wiggled node which is stable during ⅂ of WRITE
A15	⌐	Vary STOP on count until fault isolated		WRITE	⅂	Data Bus (To isolate individual faults)



Appendix D
Example RAM Test Programs

Shown below are four examples of Motorola 6800 code which are each designed to check a particular aspect of a RAM's operation.

In each instance the program could be shortened both in terms of ROM space and running time; they have been written for conceptual clarity.

Each of the programs contain a CMP B instruction. The purpose of this instruction is to put the stored test data onto the system data bus where the signature analyzer can check it.

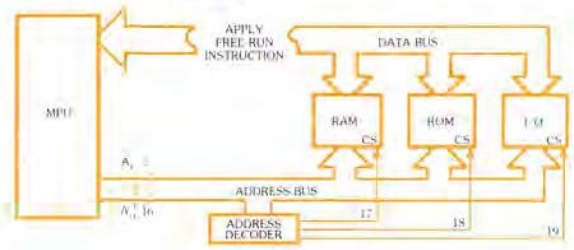
There are several other ways in which the stored test data could be checked. One method is to perform a checksum on the RAM. The software for this is just like the ROM test given in Table 1 where a V_{CC} signature checks a variable length time window for a correct checksum. Another method for checking the test data stored in RAM is to have the processor do a comparison of what is stored against what it recalls from the same location. Again, a variable length time window can provide a fault indication for the signature analyzer.

For each of these examples it is assumed that the RAM to be tested occupies the address locations $0000 to $0100. In addition, test #1 assumes that there is ROM at locations $4000 to $4100.

LABEL	MNEMONIC	LOCATION OR #	EXPLANATION
			STORE ROM IN RAM TEST
	SEI		Set interrupt mask to avoid destruction of stack data used in this test.
	CLX		Clear index register for use as a loop counter and address pointer.
	LDS	$3FFF	Load stack with $4000 - 1 (the first ROM address minus one).
TST 1	PUL A		Stack pointer increments and contents of ROM pointed to by stack are pulled into accumulator A.
	STA A	0,X	Store the contents of accumulator A in the RAM location addressed by the index register.
	CMP B	0,X	Output stored test data to data bus for checking with signature analyzer.
	INX		Increment index register to check next location.
	CMPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 1	Branch to "TST 1" if all RAM locations are not yet checked.
			CHECKERBOARD AND INVERSE CHECKERBOARD TEST
	CLX		Clear index register for use as a loop counter and address pointer.
	LDA A	#$55	Load accumulator A with the binary pattern 0101 0101.
	LDB B	#$AA	Load accumulator B with the binary pattern 1010 1010.
TST 2	STA A	0,X	Store the binary pattern 0101 0101 in the RAM location addressed by the index register.
	STA B	1,X	Store the binary pattern 1010 1010 in the RAM location addressed by the index register plus one.
	CMP B	0,X	Output the stored 0101 0101 pattern to the data bus for checking with signature analyzer.
	CMP B	1,X	Output the stored 1010 1010 pattern to the data bus for checking with signature analyzer.
	INX		Increment the index register.
	INX		Increment again since two patterns were stored.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 2	Branch to "TST 2" if all RAM locations are not yet checked.
	COM B		Complement the 1010 1010 pattern in accumulator B to obtain 0101 0101.
	COM A		Complement the 0101 0101 pattern in accumulator A to obtain 1010 1010.
	BMI	TST 2	If COM instructions have been executed once the N flag will be set and test will continue by branching to "TST 2". The second time through N will be 0 and branch will not occur.
			WALKING 1 TEST
	CLC		Clear carry flag for use.
	CLX		Clear index register for use as a loop counter and address pointer.
TST 3	LDA A	#$01	Load accumulator A with the binary pattern 0000 0001.
AGAIN	STA A	0,X	Store this pattern in the RAM location pointed to by the index register.
	CMP B	0,X	Output stored pattern to the data bus for checking with signature analyzer.
	ASL A		Arithmetic shift left to walk the one across the word.
	BCC	AGAIN	If carry flag is set, the one has walked across the whole word. If not, branch to "AGAIN".
	INX		Increment index register.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 3	Branch to "TST 3" if all RAM locations are not yet checked.
			ADDRESS TEST
	CLX		Clear index register for use as a loop counter and address pointer.
	CLA A		Clear accumulator A for use as a counter.
TST 4	STA A	0,X	Store the contents of accumulator A in the RAM location addressed by the index register.
	CMP B	0,X	Output stored address to the data bus for checking with signature analyzer.
	INX		Increment the index register.
	INC A		Increment accumulator A.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 4	Branch to "TST 4" if all RAM locations are not yet checked.

Appendix E
An Example Board Test Program

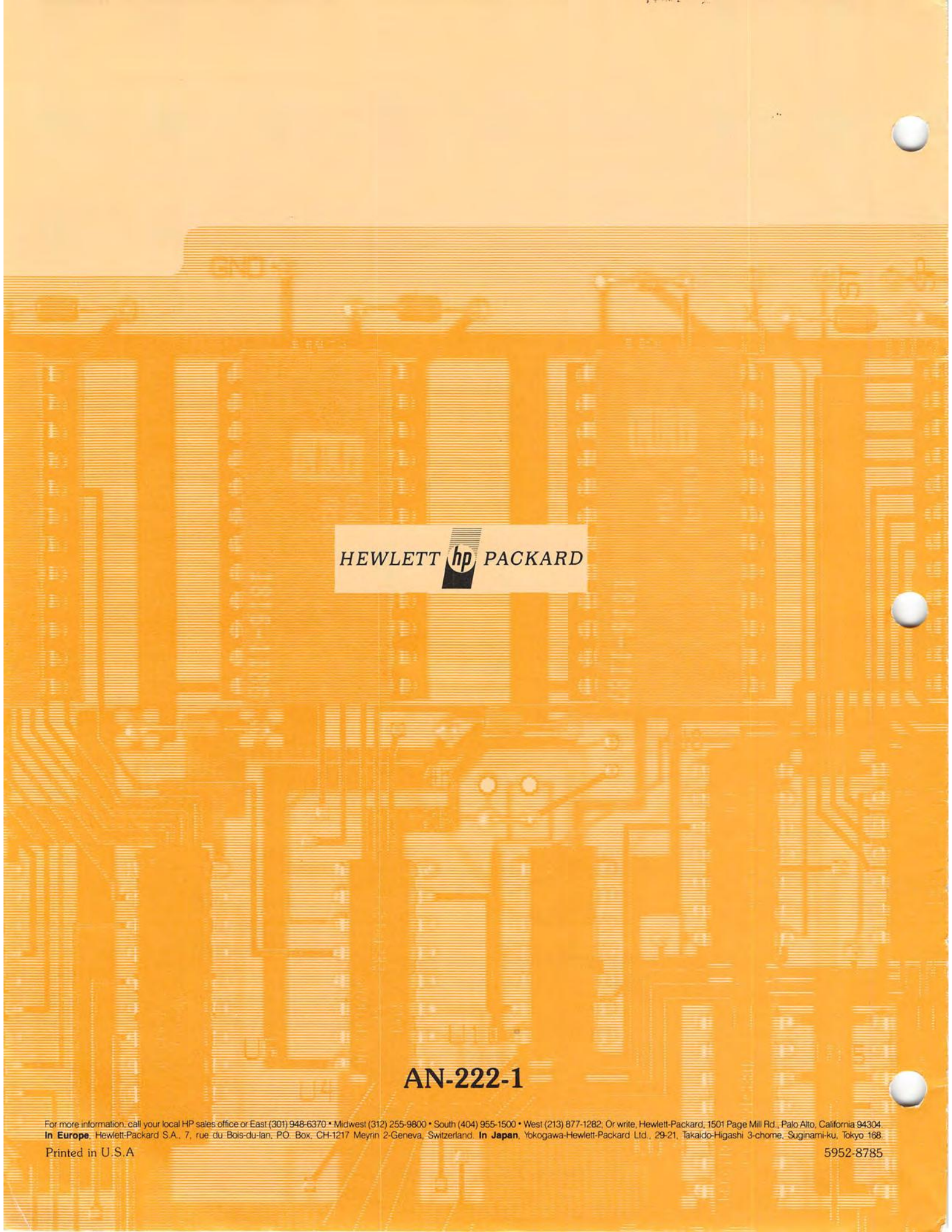
Shown below is a microprocessor-based circuit which is assumed to be free-running.



The following is a board test program which stores the correct signatures of nodes 1-19 of a known good board. The second part of the program then compares the signatures of units to be tested with the stored good signatures.

There are three basic BTL commands used in this SA test.

1) saset (insert in these parentheses parameters specifying CLOCK START and STOP inputs and edges) 2) mcon (insert relay number in these parentheses) 3) sig (insert in these parentheses the the parameters including the expected signature)	- Used to set up the inputs for SA - Connects the SA DATA input to the node where the signature is to be taken. - Causes the signature to be taken an input to the calculator.
PART 1 0: dim A$ [19,4] 1: asgn "SIGS", 1, 0, Z: if Z: open "SIGS", 1 2: rcv ref 1, .8, 2, 2 3: saset 1,"F",1,"F",1,"F" 4: for I = 1 to 19 5: (I + 11)/100 + .0044 →N 6: mcon N 7: sig "GOOD SIGNATURE", "*****", A$[I,1] 8: next I 9: sprt 1, A$	- define A$, an array in which the 19 four-character signatures will be entered. - opens data file "SIGS" on diskette for storage of signatures. "SIGS" is assigned to be file #1 - defines high and low reference thresholds at .8V and 2.0V. - CLOCK input #1, START input #1 and STOP input #1 are selected with falling triggering edges. (SA set-up) - sets up loop to take 19 signatures. - converts node number to multiplex relay number - assumes nodes 1-19 are wired to column 44 row 12-31, respectively, of the test fixture. - closes the selected multiplex relay to connect the 3060A SA circuitry to the node under test - takes the signature and stores it in row I of array A$ - end of loop. - stores (serial prints) the known good signatures on diskette in file 1 (SIGS).
PART 2 0: dim B$[19,4], C$[10] 1: rcv ref 1, .8, 2, 2 2: asgn "SIGS" 3: sread 1, B$ 4: saset 1,"F",1,"F",1,"F" 5: for I = 1 to 19; fxd 0 6: "Node #" →C$; str (I) →C$[7] 7: (I →11)/100 + 0044→N 8: mcon N 9: sig C$, B$ [I], D 10: next I	- dimensions arrays B$ and C$. - defines high and low reference thresholds at .8V and 2.0V. - transfers (serial reads) good signature on diskette to the array B$ in the calculator. - CLOCK input #1, START input #1, and STOP input #1 are selected with falling triggering edges. - sets up loop to take 19 signatures. - stores a label including the node number. This label will be used by statement 8 to designate the number of a failing node. - converts node number to multiplex relay number - assumes same fixture wiring as stated (PART 1, statement 5). - closes the selected multiplex relay to connect 3060A SA circuitry to the node under test. - takes the signature at the node under test and compares it to the correct signature stored in B$. If they are different, the system will print an error message indicating a bad signature was found on "Node #I". The bad signature is also printed out. A number indicating whether the test passed is stored in D which can be checked for troubleshooting decisions in a fault isolation algorithm. - end the loop.



AN-222-1

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 Rd., 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 3-chome, Suginami-ku, Tokyo 168.

Printed in U.S.A

5952-8785