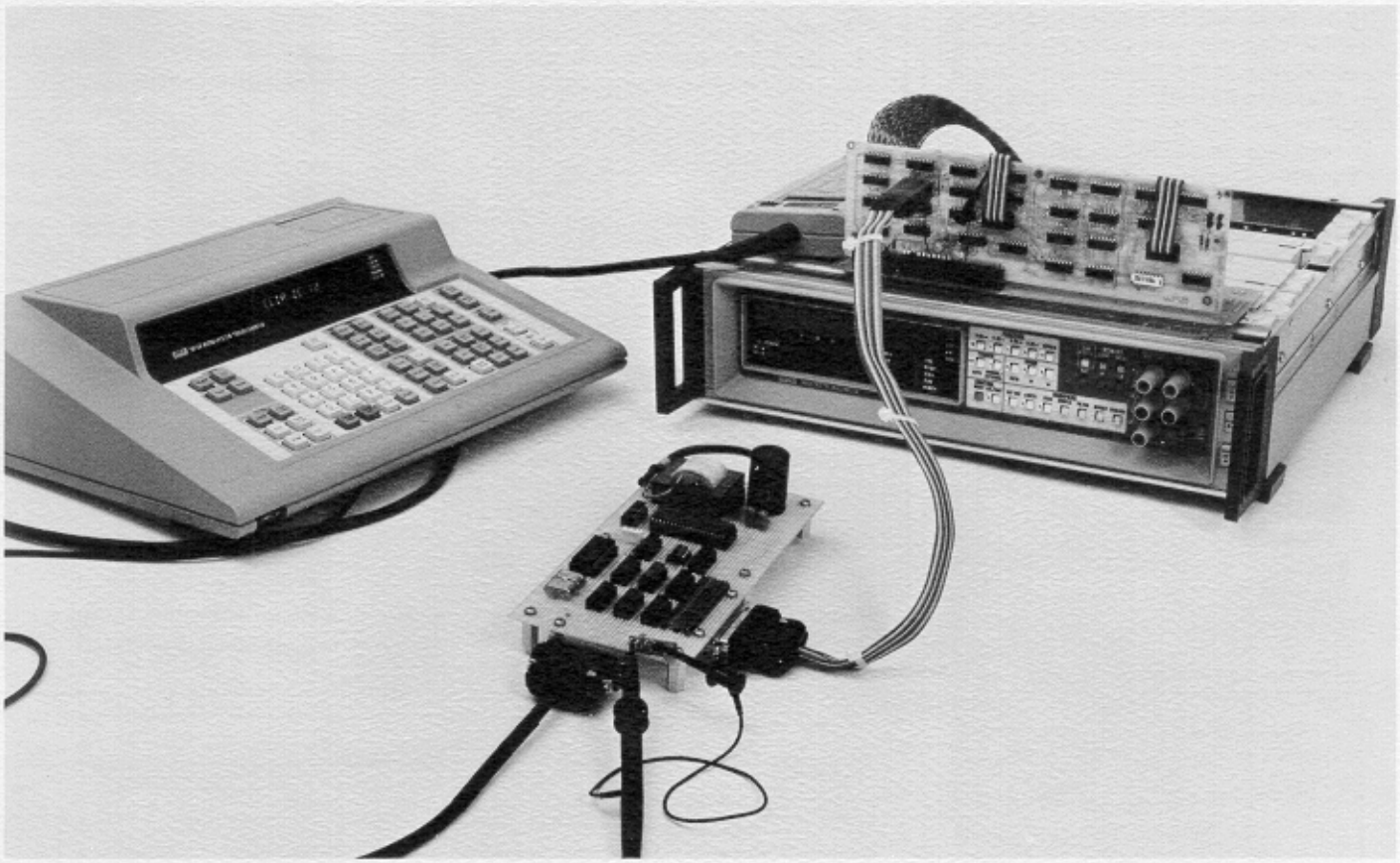
FLUKE
TROUBLESHOOTER
A COLLECTION OF ARTICLES FOR MICRO-SYSTEM TROUBLESHOOTER USERS 1984



A scanner for multipoint probing

by Ed Ferguson

I use the 9010A with Guided Fault Isolation programs to troubleshoot products with extensive amounts of I/O circuitry that include A/D converters, DACs, and parallel output ports. Once the microprocessor kernel is tested, I rely on the 9010A probe to test the I/O.

Frequently a large number of pins must be probed to test an IC. It is time-consuming to locate the pin and hold the probe for the duration of the test. Also, mistakes occur when probing the wrong pin or the probe slips off a pin in the middle of a test.

A solution to these problems is to use an IC clip in conjunction with a switching scheme to route one pin at a time, under program control, to the 9010A probe. Now the operator can place an IC clip over a device and gather the readings on the pins. When finished, the clip is moved to the next device. A suitable connector can be substituted for the IC clip to test parallel ports or board edge connectors.

The scanner described here is an electronic switch that will multiplex up to 24 inputs to the 9010A's probe. The scanner acts under command from the 9010A's optional RS-232 interface to select a given input.

As shown in figure 1, the scanner consists of a serial interface, analog multiplexers, and voltage dividers. RS-232 hex data of 20 thru 37 correspond to channels 1 thru 24. U1, U2, and U3 form the 4800 baud serial link with the 9010A. The lower five bits from the UART U3 are latched into U5 and U6. The NOR gates connected to bits 5 and 6 prevent data below hex 20 (all machine commands) from latching. Latched bits 3 and 4 select 1 of the 3 multiplexers (U9-U11) via the 1 of 10 decoder U7. Latched bits 0 thru 2 select 1 of 8 inputs on the enabled multiplexer. The multiplexers are CMOS analog switches that are essentially 120 ohm resistors when closed. The multiplexers are operating at 12 volts for minimum distortion of the input signals. U8 provides the TTL 5-volt to CMOS 12-volt interface. The high impedance voltage dividers on each channel bias the input to

(see "Scanner" inside)

Scanner...

(continued from cover)

a tristate level when the input is open, and prevent open channels from "following" active pins. The optional LED's provide a binary indication of the selected channel (0-23).

Construction of the scanner is straightforward. Note that a regulated 5-volt and 12-volt power supply are required. Include a .22 μF capacitor from VCC to ground on each IC. The 24 input lines are connected to a DB-25 connector to allow various types of test connectors to be attached. The test cable is an 18-inch long pair of ribbon cables with alternate lines tied to ground. The UUT end of the cable is connected to an IC clip or a board edge connector as required for your application. Unused channels may be left open.

There are two restrictions when using the scanner. First, the inputs are not protected and therefore must not exceed 5 volts. Second the 9010A probe pulser may be routed to a selected channel as the multiplexers are bi-directional. However, the pulser may only be used to drive open inputs since the overdrive capability is lost.

The 9010A program that controls the scanner performs the following steps:

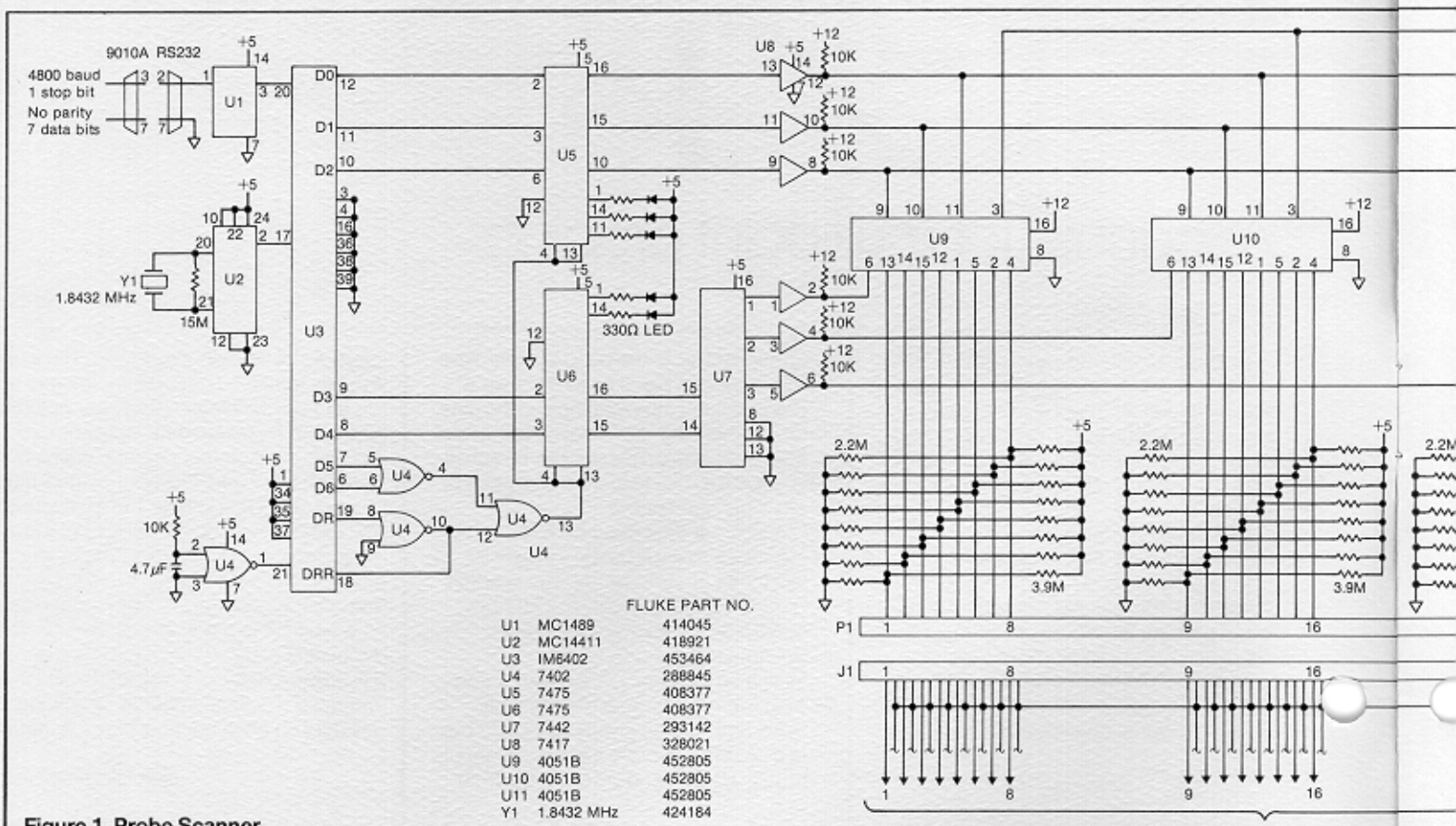
- Instruct the operator to clip the IC.
- Switch a pin to the probe via the scanner.
- Supply a suitable stimulus to exercise the circuit node.
- Measure the probe data.
- Compare the actual probe data to the expected data.
- Repeat for the remaining pins.
- Move the clip to the next device.

A 16-pin IC can be tested in 15 seconds using this approach. Refer to fig. 2 for a sample program.

An alternate approach is to connect both the 9010A and the scanner to a personal computer (PC) equipped with two RS-232 ports. Now the PC can control the 9010A and the scanner, as well as store large amounts of data. This increased capability allows us to "learn" as well as test the IC. This subject will be discussed in a following issue.

```
PROGRAM 1
DPY-CLIP IC 12 - PRESS CONT #
STOP
SYNC ADDRESS
REG8 = C
REG9 = 1
REGA = BB34
EXECUTE PROGRAM 10
REG9 = 2
REGA = 96EC
EXECUTE PROGRAM 10
REG9 = 3
REGA = 725B
EXECUTE PROGRAM 10
*
*
*
PROGRAM 10
REG1 = REG 9 OR 20 DEC
AUX-% 1
READ PROBE
RAMP @ FFFF
READ PROBE
REG2 = REG0 SHR SHR SHR
       SHR SHR SHR SHR SHR
       and FFFF
DPY-IC@8-@9 SIG $A = $2
IF REGA = REG2 GOTO 1
DPY-+ FAIL #
STOP
LABEL 1
```

Figure 2. Sample Scanner Program



		FLUKE PART NO.
U1	MC1489	414045
U2	MC14411	418921
U3	IM6402	453464
U4	7402	288845
U5	7475	408377
U6	7475	408377
U7	7442	293142
U8	7417	328021
U9	4051B	452805
U10	4051B	452805
U11	4051B	452805
Y1	1.8432 MHz	424184

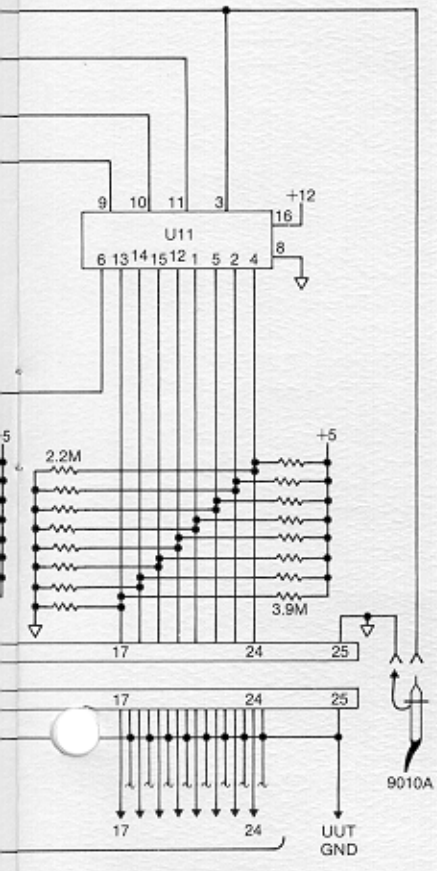
Figure 1. Probe Scanner

```
!test program - IC 12

!wait for "CONT"
!sync probe to address
!IC 12
!pin 1
!expected signature, pin 1
!scanner program
!pin 2
!expected signature, pin 2
!scanner program
!pin 3
!expected signature, pin 3
!scanner program


!continue program for remaining pins
!scanner program
!convert IC pin # to scanner channel
!send channel # out RS-232
!clear probe
!stimulus for IC
!read probe data


!probe signature
!IC #, expected & actual signature
!signature compare; pass
!signatures do not compare; fail
!wait for "CONT"
!return to test program
```



Testing a CRTC using the 9000 Series Troubleshooters

by Paul Montgomery
Technicare Corp., Solon, Ohio

This article describes one method of testing the video output generated by a CRT controller chip and its related support circuitry. Although this application deals specifically with an Intel 8279 CRTC and a Motorola 6800 microprocessor with 16K of dynamic RAM (refreshed by the DMA requests of the CRTC), the ideas are adaptable to any CRTC/microprocessor combination.

Since the composite video output to a CRT is serial, it makes sense to monitor this signal using the 9010 probe. However, the components of the composite video must be analyzed while they are still at TTL levels, before they are mixed together, level-shifted, and scaled for output to the CRT. The horizontal and vertical retrace signals are slow enough for the probe to handle, but the video data has a pulse-width of only 43 ns. per dot—much less than the 75 ns. required by the 9010 probe for accurate level detection. In order to correct for this incompatibility, the dot clock (from which all video timing is derived) is replaced with a 2MHz oscillator. This slows down the entire video system so that reliable probe readings can be obtained, while maintaining a video rate high enough to ensure proper refresh cycling of the memories.

Three utility routines are contained within a test ROM required for the CRT controller test* One routine is used to initialize all peripheral devices on the microprocessor bus to ensure that no two devices will contend for the bus.

Another routine generates random ASCII characters and loads them into the appropriate section of the display memory along with "end of the line/stop DMA" and "end of frame/stop DMA" control characters. This routine creates a page of random text that is 60 characters per line by 24 lines for a video display defined as 80 characters per line by 32 lines. The third routine synchronizes the CRT controller and the interface circuitry (figure 1) to enable the 9010 probe for only one complete screen of video information.

The interface circuitry required for this test is basically a signal gate added ahead of the 9010 probe. A standard X1 scope is used for all probing operations. The probe feeds directly into a CMOS analog switch and although it is uncompensated, probes with shunt capacitance of more than 100 pF. work well with this arrangement. The interface provides two modes of probe operation: one to simulate normal operation of the 9010 probe through the scope probe; the other to utilize the gating circuitry necessary for the probing of one full frame of video data. The circuit consisting of Z1a, Z1b, and Z2 detects the level at the input to the scope probe and presents the same level (high, tri-state, or low) either to the 9010 probe in normal mode, or to Z3a in video mode.

The 9010 program is written to continuously monitor the logic level at the probe and forestall execution of the video probing routine until a stable logic level is detected. In the video mode Z5, R7, and Z3b are used to trigger the video probing sequence by presenting that fixed logic level to the 9010 probe. Z3c is a buffer for the signal used to enable the video gate. This signal is supplied by a latch that can be written from the MPu bus (in this particular case a 6821 PIA). Once the video gate is enabled, Z7 will detect two successive vertical retrace pulses and will allow only the video information between them to reach the 9010 probe. Random video data can be generated during the horizontal and vertical retrace periods and whenever video-suppression is active. Z4 will inhibit video data from reaching the 9010 probe at these times, while R7 presents, instead a logic high level.

With the test ROM suitably interfaced to the PCB under test (through an adapter or perhaps an unused ROM socket intended for later expansion of the UUT), the CRT controller test can begin. The CRTC is initialized and programed for the appropriate screen size and timing. A non-blinking cursor is positioned at a fixed location on the screen, and interrupt requests are enabled. The display is started in order to generate horizontal and vertical retrace signals. Since the interface circuitry uses the horizontal and vertical retrace signals, it is advised to verify their operation at this point using the normal probe mode.

Next, the video memory (which should have been verified previous to this point) is loaded with random ASCII characters and any required control characters. This task is accomplished quickly and easily by a routine in the test ROM (executed by RUN UUT @_____) that generates random numbers, ANDs them with HEX data "7F," ORs them with HEX data "20," and

(continued)

**Editors Note: The initialization of the peripherals can be accomplished with troubleshooter write statements. A display fill program can be downloaded to RAM and executed with RUN UUT if the RAM supports opcode fetches.*

Testing...

(continued)

loads them into memory. The 9010 program now displays a message instructing the operator to select video probing mode and guiding them in placing the probe at the proper node. While waiting for the operator's response, the 9010 executes a loop from which escape is possible only when the 9010 probe detects a valid logic level. When the VIDEO TRIGGER button on Z5 is depressed, the tri-state level present on the 9010 probe changes to a valid (high) level and the video probe routine commences.

The ensuing 9010 program then executes a RUN UUT @_____ statement, running the third CRTC utility program from the test ROM. This ROM program has three basic functions. First it must determine at what point in time the video gate can be enabled. The video gate must be enabled after the DMA requests for the current frame of data have ceased, but before the vertical retrace pulse occurs at the end of the frame of video. This condition must be met so that the page register, which is incremented by DMA requests, can be loaded to the top of the desired video page before DMA requests begin on the next video frame. To accomplish this, the program resets the interrupt request flag in the CRTC and continuously monitors this bit in the CRTC until it is again set (by the CRTC) during the output of the last line of the video frame. Secondly it must enable the video gate by directing the MPU to set the video enable bit in the latch that drives Z3c. Thirdly it must load the page register to the one page of video memory (this particular PCB has 8 pages of software controlled scrollable video memory) containing the data that was generated for this test.

With the video gate enabled, Z7 is clocked by the next vertical retrace pulse, and the next frame of video data is presented to the 9010 probe. While the probe is gathering video data, the 9010 main instrument is executing a timeout loop. The vertical retrace pulse at the end of the measured frame clocks Z7a to disable Z3a. A constant high level is then presented to the 9010 probe by R7 and Z3b until the main instrument regains control of program execution and resets the video enable bit.

Due to the interaction of the interfacing hardware and the controlling software, the probing sequence cannot be retriggered during the acquisition of a frame of video, thereby guaranteeing that only one frame of video will be monitored. The total count information extracted from the probe reading is highly stable. It represents the exact number of dots that would be written to a CRT monitor. The probe reading also provides signature data, but depending upon the accuracy of the CRTC timing, this information may not be reliable.

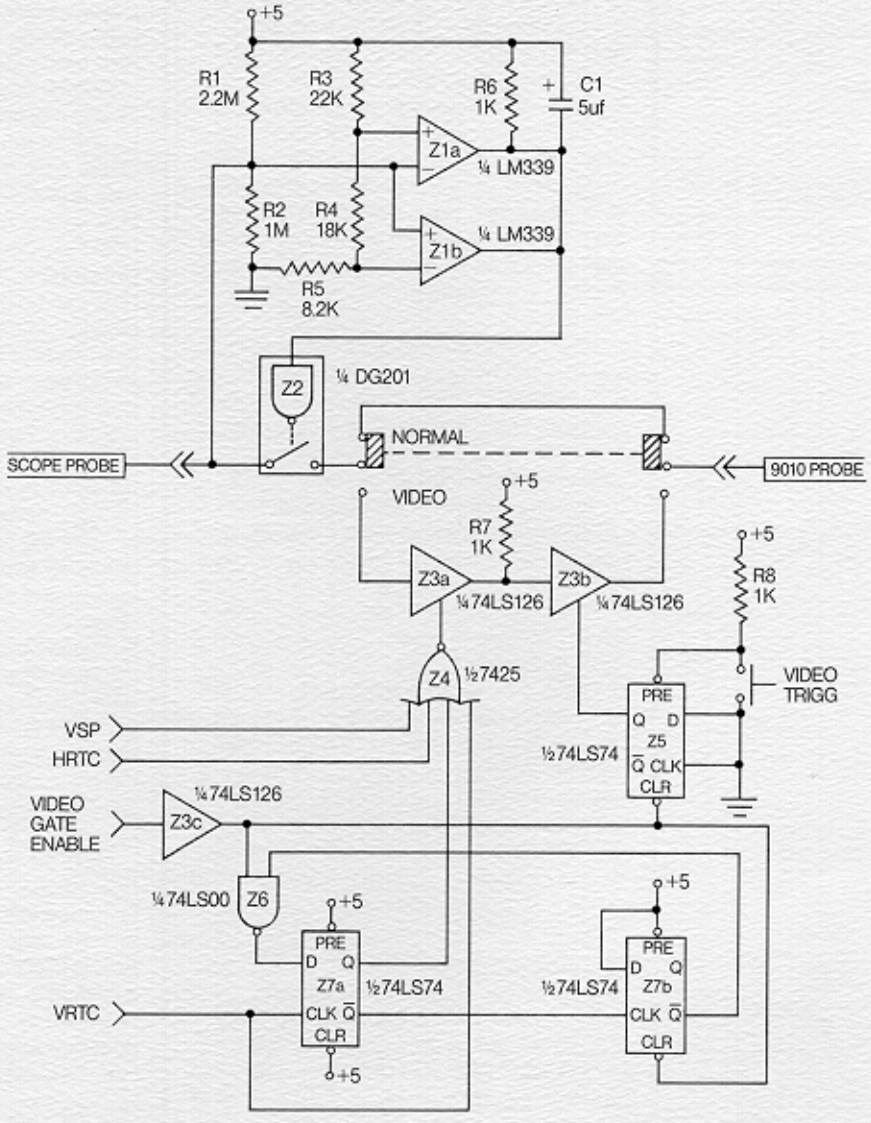


Figure 1. Interface Circuitry 9010 Video Probe

Solving noise and timing problems

Have you ever plugged your 9000-Series Troubleshooter into an unfamiliar Unit Under Test (UUT) and had difficulty in getting it to run? Maybe BUS TEST passes, but RAM SHORT fails. When you try a WRITE followed by a READ, you get erratic results. You've used the Troubleshooter enough to have confidence in it. It's bailed you out of some tough troubleshooting situations before, but now you're stuck. What's wrong?

When problems getting a UUT to work well with the Troubleshooter occur, they're usually caused by system noise and/or marginal timing.

System Noise

System noise is always present. It's not new; it's always been there. However, it becomes more and more of a problem as system speeds increase, because the sharp-edged fast data pulses generate plenty of noise. This noise is dealt with by system designers enough to get the UUT to operate under expected operating conditions. But when the Troubleshooter Interface Pod is plugged into a UUT, two things happen.

First, the UUT is opened up to the outside world. The integrity of the UUT's own shielding is violated if the UUT cannot be closed up with the Interface Pod connected to it. This may expose the UUT to noise from outside equipment or from other portions of its own circuitry. This situation doesn't usually cause problems, but if it does, it must be dealt with creatively with each different type of UUT.

The second thing that happens is several inches of cable are added to the system bus. Even if you are able to keep this added cable inside of the UUT's shielded environment, it may cause internal system noise to be picked up on the μP lines. (The new strict FCC requirements deal primarily with noise that is emitted into the environment outside of the UUT. For obvious reasons, the FCC requirements do not restrict noise that is contained within the UUT.) The best solution to this problem is to be sure that your Interface Pod uses a shielded UUT cable (the cable that plugs into the μP socket).

Marginal Timing

If system noise doesn't seem to be causing your problem, then you may be troubled by marginal timing. Marginal timing occurs when the system is designed close to the timing requirements of the various devices in the UUT (usually in order to maximize the performance of the UUT). For example, a μP or Interface Pod sends out address information during a READ and expects the system ROM, RAM, or the addressed peripheral device to return stable data to the bus within a predefined time period. The timing margin is the number of extra nanoseconds (nsec) that are available during this process. Marginal timing problems occur when the timing margin isn't long enough for data to be stable when the μP of Interface Pod expects it to be.

Designing a UUT with marginal timing characteristics is generally recognized as poor design practice, but it does happen, intentionally or unintentionally. (To avoid this situation, see the "What's the difference?" article in the August 1983 issue of the TROUBLESHOOTER for an example of timing margin testing.)

A good way to detect a marginal timing problem is to measure the timing of the system using the Troubleshooter and an oscilloscope. To do this, synchronize the oscilloscope to DATA using the Troubleshooter TRIGGER OUTPUT BNC Connector (on the rear panel). Do a looping READ operation and look for marginal timing problems while comparing your waveforms to the timing waveforms in the manual for your particular Interface Pod.

A Timing Margin Estimator

Evaluating the timing is not an easy task but a tool can be built to make this much easier: the Timing Margin Estimator. What is needed is a plug-socket device that will connect between the Interface Pod cable and the UUT μP socket that inserts a delay circuit (see Figure 1) into one of the data or address lines, e.g., DO). The timing margin can be experimentally determined by adding a variable delay in any portion of the system's address or data path and adjusting the delay until a system error occurs.

NOTE: The circuit of Figure 1 will also work on μPs with multiplexed data/address lines. The delay will affect both an address signal and a data signal, but it will still detect the minimum delay that will cause a problem.

If a 10-turn potentiometer with a calibrated knob is used, a synchronized oscilloscope can calibrate the device according to knob settings. Figure 2 shows a sample calibration curve. Due to different drive currents and voltage levels of different devices, the Timing Margin Estimator should be re-calibrated when it is used on a different UUT (or even a different part of the same UUT).

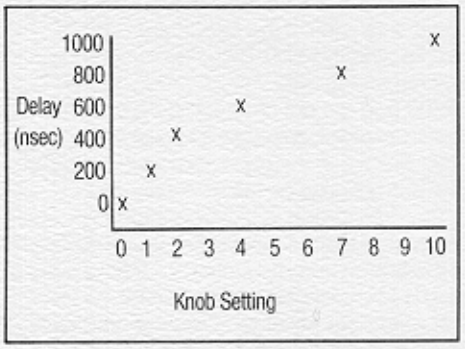


Figure 2. Sample Delay Curve

By adjusting the potentiometer, approximately 10 to 1000 nsec of delay can be introduced into the signal path. Increasing the delay during a looping RAM SHORT is an easy way to determine the timing margin of the RAM. For ROM, the delay must be adjusted while performing a looping ROM test. To speed up the test time, limit the ROM test to about 10 bytes.

The timing margin estimator can be a good tool for evaluating new UUTs and for preparing them for 9000-Series testing. You can also use the Timing Margin Estimator to measure the timing margins on a known-good UUT, and then use those measurements to determine if the timing margins on a suspect UUT are within the proper range.

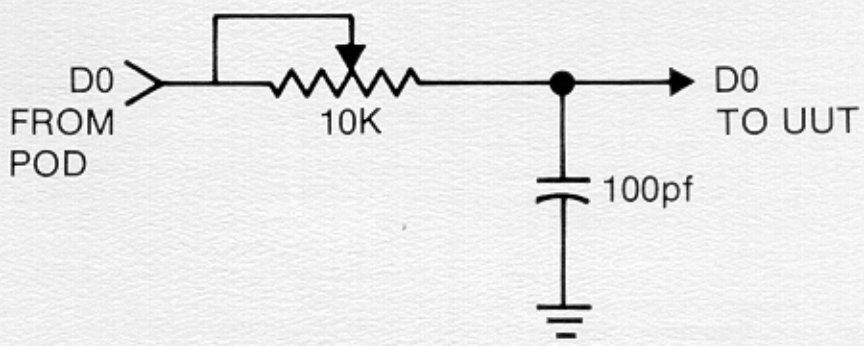


Figure 1. Timing Margin Estimator

A pod adapter for the 68008

by Tom Locke

This article describes a circuit that will adapt a Fluke 68000 pod to 68008 based systems. Once connected, the troubleshooting functions and some of the tests of the 9000 Series Troubleshooters can be used to troubleshoot 68008 microprocessor systems.

The differences between the 68000 and 68008 are very minor and only requires the translation of pins between the pod and the UUT. The 68000 uses a 16 bit data bus while the 68008 uses only 8 data bits. The 8 data lines of the 68008 are connected directly to D0-D7 of the 68000 pod. The remaining data lines, D7-D15, of the 68000 pod are not connected. By attaching to the lower data bits, it allows the operator to use the pods special Interrupt Vector capability on the 68008 system.

The data strobe (DS) of the 68008 is connected to the Lower Data Strobe (LDS) of the 68000 pod because D0-D7 is valid when LDS is low. This arrangement means that valid data will always appear on the lower 8 data bits of all Troubleshooter commands.

The 68000 pod does not physically have an address 0 line, and the 68008 does. For this reason it is necessary to connect A0-A19 of the 68008 conector to A1-A20 respectively on the 68000 pod. With the address lines connected in this way it makes it necessary to enter all addresses into the Troubleshooter shifted one place to the left. This can be accomplished by simply using the SHIFT LEFT key. Listed below are a few examples.

UUT address	Troubleshooter address
000000	00 0000
000001	00 0002
0ABC13	15 7826
0FFFFF	1F FFFE

68008 adapter operation is very easy. Normal word addresses work fine for most reads and writes. The ability to do simple reads and writes on the UUT means that the TROUBLESHOOTING functions will work as well. Remember, for RAMP and WALK, while the upper 8 data bits are moving nothing is happening on the 68008 UUT.

The limitations of the adapter are in the built-in RAM tests and of course RUN UTT. The 68000 pod will expect 16 bit opcodes in the RUN UUT mode and with ony 8 data lines conected, RUN UUT is impossible.

The RAM SHORT and RAM LONG tests both expect to find 16 read/writable data bits at each UUT address. With the upper 8 data bits of the pod not connected, these tests will not operate through this adapter. Even the ROM test computes a signature on all 16 data bits no matter how they may be connected. Using the QUICK RAM and ROM tests in the BYTE mode is the best way to test the UUT's RAM and ROM areas.

The addresses must always be ODD, when the BYTE MODE is specified through the 68000 pods Special Addresses. This is due to the fact that the 68000's LDS which, remember, is connected to DS on the 68008, is only valid on ODD address accesses in the 68000 pod. To compute the proper special address first shift the UUT address left one place, as before, and then increment once to get an odd value. This address should then be combined with the special address for BYTE specifications. The last two steps can be accomplished by setting a register to 900 0001 and ORing it with the modified address. Listed below are a few examples.

UUT address	Byte address
000000	900 0001
000001	900 0003
0ABC13	915 7827
0FFFFF	91F FFFF

Remember that only the UUT addresses are shifted left. The upper bits used for designating Special Addresses are interpreted by the Pod, and shifting is not required.

To specify a Quick Ram test for a UUT address range of 0000-1FFFF (128 K bytes)

WRITE @ 2900 0001=0
WRITE @ 2903 FFFF=21

To specify a Quick Rom test for a UUT address range of 80000-8FFFF (64 K bytes)

WRITE @ 3910 0001=0
WRITE @ 3911 FFFF=21

The construction of the 68008 adapter can be accomplished by simply wiring a 64 pin socket to a 48 pin socket saver using the diagram in figure 1. The only circuit component required for the adapter is a 10 K ohm resister between BGACK and VCC. The Bus Grant ACKnowledge is not implemented by the 68008 so the 68000 pod has to be fooled into believing he has the bus all the time.

Once the wiring is completed the socket and the socket saver should be mechanically connected with standoffs or some other convenient method. The one I have constructed works perfectly for my application and should work on all 68008 applications.

These kits are now available through your local Fluke sales office.

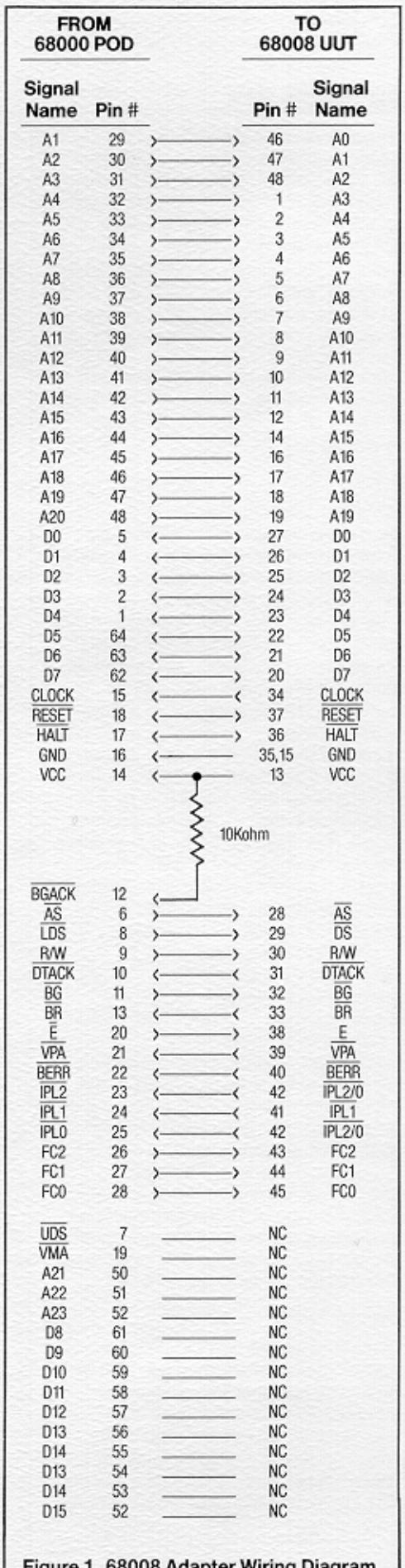


Figure 1. 68008 Adapter Wiring Diagram

Built-in test failure detection

by Jim Clodfelter

The 9010A Micro-System Troubleshooter has several built-in tests that include the Bus test, ROM test, RAM short, RAM long and Auto test. When an error is discovered during one of these tests, the 9010A stops the test and displays an error message indicating the fault type. If these built-in tests reside in a program, the error reporting is still the same: Program execution is halted.

Many customers have asked if there's a way to branch out of the test routine when an error is detected, either to a troubleshooting routine or to indicate the presence of errors without halting program execution. For those customers with the RS-232 interface option, using the set-up commands makes this feat possible.

The following technique was developed by Leo Longo, our Field Systems Engineer in the Tustin office. The set-up command "SET EXERCISE ERRORS? YES" is the command used by the 9010A to halt built-in tests and display the error detected. If the set-up condition is set to "NO", when the error condition is recognized, the test does not stop but the message that would normally be displayed is sent out the RS-232 port on pin 2.

This same action will again be true if the built-in tests are used within a program. Pin 2 is the transmit line for the RS-232 signal and pin 3 is the receive line. What we would like to have is some way for the 9010A to monitor the RS-232 output so that it would know when an error occurred. This can be done by tying the output pin (pin 2) to the input pin (pin 3). Now, whenever an error message is sent out pin 2, a stream of corresponding data would enter pin 3. The activity of data passing from the output to input could be monitored through the status of the RS-232 interface. Figure 1 illustrates the bit assignment for the RS-232 status.

We will monitor the condition of Bit 3 in the status register to check for test errors. If a built-in test fails, the error message will go out pin 2 and re-enter pin 3. This will set Bit 3 to a one (1) since characters were received. If no failure occurred, no input to pin 3, Bit 3 would remain a zero.

To implement this technique, take an RS-232 plug/adapter and tie pins 2 and 3 together, then plug it into the RS-232 port on the back of the 9010A. Set the set-up condition "EXERCISE ERRORS?" to "No". Now design a program that checks the RS-232 status register for a character received condition after each Built-In test. If Bit 3 is high, the test failed because a character was received. The program can then branch to an error-handling routine.

Some special AUX I/F symbols are used in the programming mode to manipulate the RS-232 interface. These can be found in the programming manual or reference card.

Slash (/)—When followed by a hexadecimal digit, places the next byte of data from the RS-232 interface in a register designated by the hexadecimal digit. If data from the RS-232 interface is not present, program execution is suspended until data is present.

Backslash (\)—When followed by a hexadecimal digit, places the status of the RS-232 interface in the lower five bits of the register designated by the hexadecimal digit.

Plus sign (+)—When the last character in an AUX I/F step, prevents a line termination sequence from being sent at the end of the line.

You can use all or parts of this program to branch on individual built-in tests. This routine can be used as a tool by the operator of the 9010A Micro-System Troubleshooter to enhance troubleshooting speed and capabilities, by going directly to a fault isolation program whenever the built-in test finds an error.

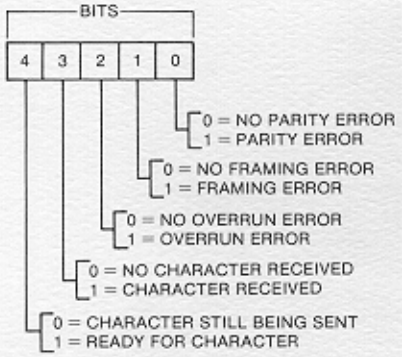


Figure 1. Format of RS-232 Status

```
AUX\1+                    Read the RS-232 status register.
If Reg 1 AND 8 = 0 GOTO 1 We are looking to see if Bit 3 was set high from some
                          previous condition. If it is zero, proceed with your test.
AUX/1+                    Places next byte from RS-232 in register and resets
                          Bit 3.

LABEL 1
Bus Test                  Now perform the desired built-in test.
AUX\1+                    Read the RS-232 status register.
If Reg 1 AND 8 > 0 GOTO A If Bit 3 is high, we detected an error. Jump to a sub-
                          routine that will perform a Guided Fault Isolation (GFI)
                          routine that tests Bus failures.
RAM Short Test            Perform the next built-in test.
AUX\1+                    Read RS-232 status register.
If Reg 1 AND 8 > 0 GOTO B If Bit 3 is high, the RAM short test failed. Label B would
                          be a GFI for the Ram area.
ROM Test                  Perform the next built-in test.
AUX\1+                    Read RS-232 status register.
If Reg 1 AND 8 > 0 GOTO C If Bit 3 is high, the ROM test failed. Label C is a GFI for the
                          ROM area.
I/O Test                  Perform the next built-in test.
AUX\1+                    Read RS-232 status register.
If Reg 1 AND 8 > 0 GOTO D If Bit 3 is high, the I/O test failed. Label D is a GFI for
                          the I/O area.
GOTO F                    Label F would be the end of your GFI routine.
LABEL F                   End of Program
```

Listing 1. Branch on Error Routine

John Fluke Mfg. Co., Inc.
P.O. Box C9090, Everett, WA 98206
(800) 426-0361 (Toll Free) in most of U.S.A.
206-356-5500 from other countries

Fluke (Holland B.V.)
P.O. Box 2269, 5600 CG
Eindhoven, The Netherlands
Tel. (040) 458045, Tlx 51846

Printed in U.S.A. P0009A-01U8501/SE EN