FLUKE
TROUBLESHOOTER
A COLLECTION OF ARTICLES FOR MICRO-SYSTEM TROUBLESHOOTER USERS 1985

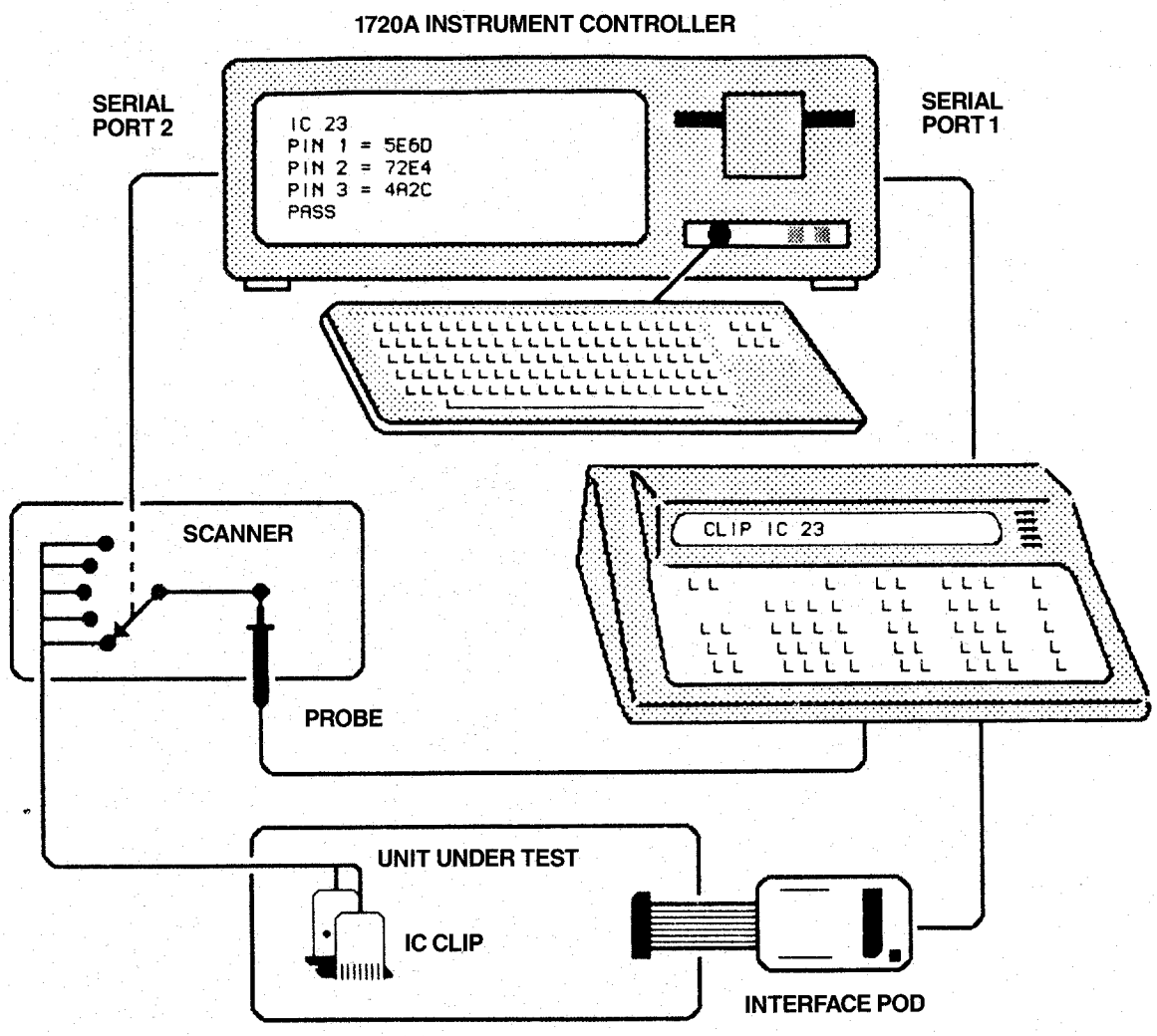


Figure 1.

A scanner for multipoint probing

by Ed Ferguson

Part 2

Part 1 of this article appeared on cover of 1984 TROUBLESHOOTER Annual.

In Volume 2, Issue 1 of the TROUBLESHOOTER (reprinted in the 1984 ANNUAL edition), I wrote an article that described a scanner circuit to multiplex up to 24 inputs to the 9010A's probe. The scanner was controlled from the 9010A's RS-232 interface to select a given input from an IC clip or board edge connector, and routed the high, low, or tristate levels from the selected input to the 9010A probe. A sample 9010A program showed a method to automatically probe and compare the signature at each pin of an IC.

In the second part of this article I have expanded upon this system to show that with the addition of a personal computer you can now "learn" as well as test a UUT, one IC at a time. By using a good UUT, the computer can store the data gathered from each IC and record this data for later comparison with faulty UUTs.

Any computer with two serial ports and a floppy disk can be used. Serial port 1, connected to the 9010A, selects the stimulus program and reads the resulting probe data. Serial port 2 connects to the scanner and just selects the probe's input. The floppy disk stores the probe data gathered at each pin of each IC. The system shown in fig. 1. uses a Fluke 1720A Instrument Controller with a menu driven program written in BASIC. The software:

(continued next page)

Scanner...

(continued from cover)

1. Learns the signature, count, or level history at each pin of an IC.
2. Tests a given IC.
3. Displays a list of failed IC's and their pin numbers.

Basic Program

The computer's program contains a disk array to store the data for each IC. The array contains the IC number, the number of pins, the 9010A stimulus program number and the probe data for each pin.

When "learning" a good UUT, the computer will prompt the operator to clip on to an IC and enter its number. The program then looks in the array for the IC pin count and the stimulus program number. It begins a loop at line 500 which ranges from 1 to the total number of pins. Lines 510-530 convert the IC pin number to the hex data required by the scanner. The pin numbers are shifted at line 520, as required, so a single 16 pin IC clip will work with all ICs having 16 pins or less.

Next, the program triggers the 9010A to execute the proper stimulus program. The 9010A reads the probe and sends the probe data to the computer. The computer then strips the signature from the probe data and stores the IC number, pin number, and signature in the array. This process is repeated until the pins on all ICs are learned. The program takes several probe readings at each pin to insure that the signatures are repeatable. If they are not, a count or level may be stored instead.

The troubleshooting philosophy is to first test the UUT kernal (BUS, ROM, RAM) using the 9010A alone. To test the UUT beyond the kernal, the "learn" steps are repeated except that the actual signatures are now compared to those stored on disk, and any discrepancies are displayed. You may wish to design the program so that the operator enters the IC number to test, or allow the program to guide the operator by adding the necessary logic.

9010A Program Description

The following description is for the 9010A. If using a 9020A, it will be remotely programmed by the computer simplifying the interface used. The 9010A main program (Program 0) inputs the desired stimulus program number (1-3) from the computer, executes it and can be expanded, if needed, to handle more stimulus programs.

9010A Main Program (Program 0)

```
0: LABEL 0                  ! loop until a valid stimulus program
   AUX/1                    ! number (1-3) is received from
   IF REG 1 = 1 GOTO 1      ! computer port 1.
   IF REG 1 = 2 GOTO 2
   IF REG 1 = 3 GOTO 3
   GOTO 0                   ! did not receive a valid program number
1: LABEL 1
   EXECUTE PROGRAM 1        ! stimulus program for IC 1
   GOTO 0
2: LABEL 2
   EXECUTE PROGRAM 2        ! stimulus program for IC 2
   GOTO 0
3: LABEL 3
   EXECUTE PROGRAM 3        ! stimulus program for IC 3
   GOTO 0
```

A stimulus program:

- exercises the circuit under test;
- sends the probe data to the computer; and
- must be written on the 9010A for each circuit under test before learning the UUT.

Program 1 is a sample of a stimulus program.

9010A Stimulus Program For IC1 (Program 1)

```
SYNC DATA          ! syncronize probe
READ PROBE         ! clear probe data
RAMP @ 2001        ! stimulus for IC under test
READ PROBE         ! read probe data
DPY-$0             ! display probe data on 9010A
AUX-$0             ! send probe data to computer port 1
```

Space does not permit a full listing of BASIC program, however the routine below shows the steps to close the scanner switch, select the 9010A stimulus program, and input the probe data.

```
10   ! PN    is number of IC pins
20   ! SW    is scanner switch 1-24
30   ! ST    is 9010A stimulus program 1-3
40   ! PD$   is 9010A probe data
•
•
500  FOR L = 1 TO PN                  ! begin loop
510  LET SW = L
520  IF L > PN/2 THEN LET SW = L + 2*(8-PN/2)
530  PRINT #3, CHR$ (31 + SW)         ! close scanner switch (port 2)
540  PRINT #2, CHR$ (ST);             ! select 9010A stimulus program
                                      ! (port 1)
550  INPUT #1, PD$                    ! input probe data from 9010A
                                      ! (port 1)
•
•
```

Compiler gets programs working sooner

by Lee Molho

Marcus Information Systems, Santa Monica, California

The 9010A Language Compiler lets you write test programs that are easy to read. Abundant evidence shows that programs which are easy to read are easier to understand, easier to debug, and easier to change later on, compared with hard-to-read programs. Here's an example:

"PROBELEVEL" is a subroutine that is going to take care of details of reading logic levels with the 9010A probe. All a calling program will need to do is tell PROBELEVEL the level it expects to find, EXECUTE PROGRAM PROBELEVEL, and look for a "true" or "false" response. Because the 9010A Language Compiler lets us name registers, let's call the level that is wanted "WANTLEVEL", the true-or-false response "VALIDITY", and assign registers to them. Since those registers are used to pass values between programs, we must use two of the global registers by writing declarations in the main program like, "ASSIGN REG8 TO WANTLEVEL" and "ASSIGN REG9 TO VALIDITY."

What we've just done is to specify our program by defining its input and output. Now, take a look at the listing of a working version of PROBELEVEL.

At the beginning, a comment tells what the program does. Meaningful names are chosen for labels and registers. Step by step, comments explain what is going on and what we think the program is doing. We put those in when the program is fresh in our mind; later, they will explain the program when a problem shows up.

Even though it is short, PROBELEVEL contains a multilevel branching structure. If your program didn't seem to work quite right, which would you rather debug—the Language Compiler version, or the 9010A keyboard Program 9.

Program Probelevel

```
! ENTER WITH "PROBE U#—#" IN DISPLAY, WANTLEVEL = 0, 1, or 99 (99 = FLOAT).
! RETURNS 0 ( = FALSE) OR 1 ( = TRUE) IN VALIDITY.
! TESTS FOR AND EXPECTS LEVELS, NOT PULSES.
! WANTLEVEL AND VALIDITY ARE GLOBAL REGISTERS DECLARED IN MAIN PROGRAM.
DECLARATIONS                              ! LOCAL ONLY
    ASSIGN REG0 TO PROBEVAL
    ASSIGN REG1 TO ANSWER
! END DECLARATIONS; START OF PROGRAM

    SYNC FREE—RUN                         ! INITIALIZE PROBE MODE.
    DPY – +, CONT WHEN READY              ! CONCATENATES WITH U#—#.
    STOP
BEGINPROBE:
    READ PROBE                            ! DO TWICE. SEE 9010
                                          ! PROGRAMMING
    READ PROBE                            ! MANUAL P. 5-13.
    VALIDITY = 0                          ! SET VALIDITY FALSE, PROVE
                                          ! TRUE.
    IF PROBEVAL = 4000000 GOTO GOTLOW     ! THESE VALUES REQUIRE
                                          ! THAT THE
    IF PROBEVAL = 1000000 GOTO GOTHIGH    ! PROBED NODE BE A
                                          ! STABLE LEVEL
    IF PROBEVAL = 2000000 GOTO GOTFLOAT   ! MAYBE NOT READY IF NOT
                                          ! STABLE:
    DPY—PROBE LEVEL UNSTEADY. RETRY?1     ! ENTER = RETRY,
                                          ! CLEAR = EXIT.
    IF ANSWER = 1 GOTO  BEGINPROBE        ! ELSE,
    GOTO  RETURNFALSE                     ! IF HE'S READY, IT'S
                                          ! BAD.
GOTLOW:
    IF WANTLEVEL = 0 GOTO RETURNTRUE      ! ELSE,
    GOTO  RETURNFALSE
GOTHIGH:
    IF WANTLEVEL = 1 GOTO RETURNTRUE      ! ELSE,
    GOTO  RETURNFALSE
GOTFLOAT:
    IF WANTLEVEL = 99 GOTO RETURNTRUE     ! ELSE,
    GOTO  RETURNFALSE
RETURNTRUE:
    VALIDITY = 1
RETURNFALSE:                              ! IT'S FALSE ALREADY.
```

Program 9

```
SYNC FREE-RUN
DPY – +, CONT WHEN READY
STOP
LABEL 0
READ PROBE
READ PROBE
REGD = 0
IF REG0 = 4000000 GOTO 1
IF REG0 = 1000000 GOTO 2
IF REG0 = 2000000 GOTO 3
DPY-PROBE LEVEL UNSTABLE. RETRY?1
IF REG1 = 1 GOTO 0
GOTO 4
LABEL 1
IF REGC = 0 GOTO 5
GOTO 4
LABEL 2
IF REGC = 1 GOTO 5
GOTO 4
LABEL 3
IF REGC = 99 GOTO 5
GOTO 4
LABEL 5
REGD = 1
LABEL 4
```

Signetics 8X300 pod adapter

by Kirk E. Schuetz

Kentron International , Topeka, Kansas 66619

Since Fluke does not offer an interface pod for a Signetics 8X300 microprocessor (μP) based system, it was necessary to build an adapter to allow an existing interface pod to communicate with a unit under test (UUT). The Z80 pod was chosen for its availability and for its 5 MHz maximum clock rate, its unmultiplexed address and data lines, and its single 5 volt supply. This article describes an interface pod adapter that allows the 9000 Series Micro-System Troubleshooter to test and troubleshoot a printed circuit board controlled by a 8X300 μP.

A 50-pin header is needed to plug into the 8X300 socket. If a 50-pin header is unavailable, a suitable 50-pin configuration can be constructed by cutting up dual in-line package headers and glueing them to a fiberglass vector board.

IC	Type	VCC	GND
U1	74LS04	14	7
U2	74LS74	14	7
U3	74LS244	20	10
RN1	899-1-R180	—	14

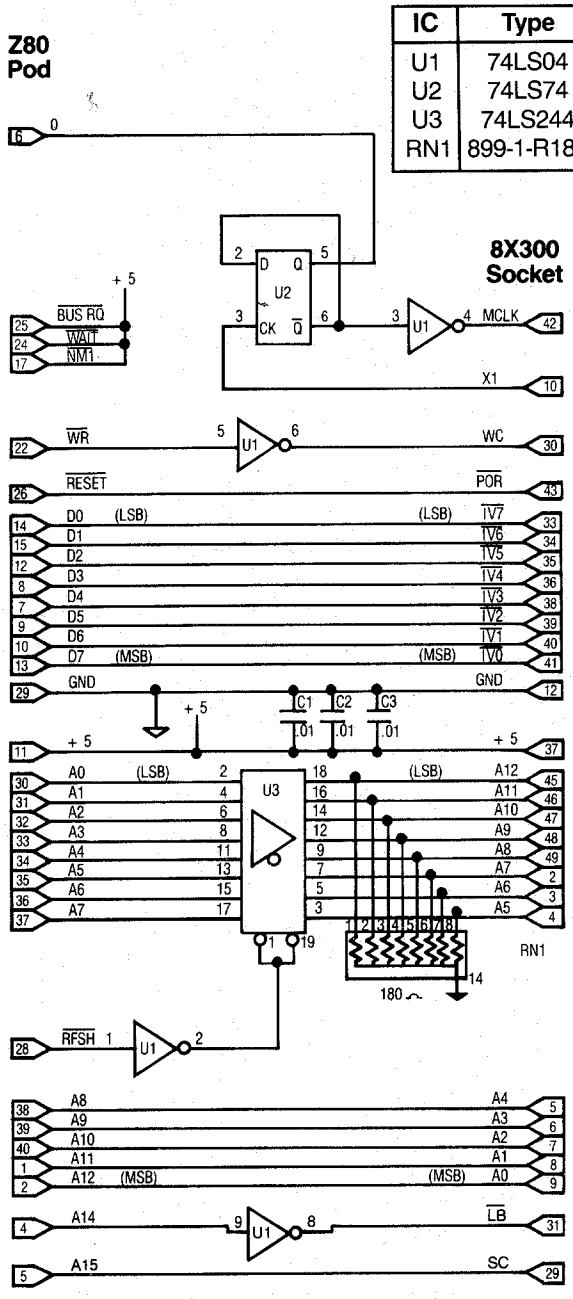


Figure 1. Signetics 8X300 to Z80 Pod Adapter

The 8X300 typically uses clock rates over 5 MHz which makes it necessary to divide the UUT clock to below 5 MHz. As the schematic (Figure 1) indicates, a D-type flip-flop, U2, is used to divide the UUT clock (X1) in half. This provides a satisfactory clock rate to run the pod and a proper rate for the MCLK signal normally coming out of the 8X300.

The 8X300 has three busses: the instruction bus (I0-I15), the address bus (A0-A12), and the interface vector bus ($\overline{IV0}$-$\overline{IV7}$). The instruction bus does not feed back to the Z80 pod, so these lines have to be probed for signatures. The address and interface vector lines for the 8X300 are defined differently than the address and data lines of the Z80. They are connected as follows:

	Address		Data	
	Z80	8X300	Z80	8X300
LSB	A0	A12	D0	$\overline{IV7}$
MSB	A15	A0	D7	$\overline{IV0}$

The Z80 $\overline{WR}$ output is inverted to provide the write command (WC) output, and the Z80 A15 output is used for the select command (SC) output. The Z80 A14 output is inverted to provide the left bank (LB) output. The unused Z80 A13 output, if needed, could be inverted to provide the right bank (RB) output.

The automatic refresh function of the Z80 puts unwanted signals on address lines A0-A7. This causes unstable UUT signatures. This problem is corrected by the tri-state buffer, U3. The buffer outputs are disabled during $\overline{RFSH}$, causing the buffer outputs to go to the high impedance state. The effects of the high impedance outputs are minimized by the 180 ohm pull-down resistors of RN1.

The adapter worked well to test and troubleshoot all or most of the circuits on the UUT. This is just another example of the versatility of the Fluke 9000 Micro-System Troubleshooter.

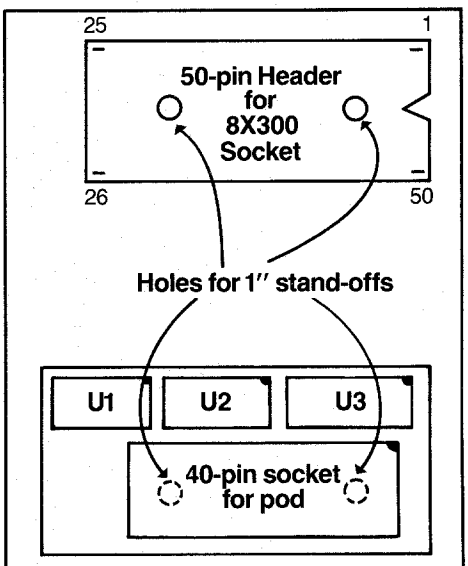


Figure 2.

Taking measurements with the Async option in a programming mode

The new Asynchronous Signature Probe Option from Fluke comes complete with all the software to operate the option in the immediate mode. But the operator has the added capability to customize their own routines using the programs provided with the Async option as subroutines for a larger customer designed testing package.

The cassette tape provided with the Async option contains the Merge tape program (Program 0 Side B) that allows the operator to selectively pull programs from a master tape and merge them with new programs under development. Using this program the operator can select those programs from the Async option software that will best fit their needs for testing.

Several programs are supplied on the cassette included with the Asynchronous Signature Probe Option. Below is a listing of all the programs; an explanation of each program can be found in section three of the operators manual.

Program Number	Program Name
Side A	
0	Initialize
1	Interactive Operation
2	Service Gate Keys
3	Service Setup Key
4	Display Gate
5	Start Setup
6	Stop Setup
7	Clock Setup
8	Enable Setup
9	Stop Count Setup
11	Event Setup
12	Setup Hardware
13	Arm Gate
14	Get Signature
15	Get Events
16	Get Waveform
17	Read Data and Status
18	Send Op Code
19	Display Waveform
20	Append Signature

Programs 0-9 and program 11 are user interface programs used by the option when it is operating in the interactive mode. Their primary function is to load the initialization register (Reg 8) with the proper setup values for the polarity of the control signals, source of the control signals, count limit for the stop counter, and mode of the events counter.

Programs 12-20 are library programs used by the option, independently of one another and the user interface programs. These programs will become a part of the troubleshooting sequence when gathering signatures, event counts or waveforms.

1. Initialize Register 8

The first step in incorporating the Asynchronous Signature Probe Option into a guided troubleshooting routine is to load the initialization register. If the programmer knows which setup values he wants they can be loaded into the initialization register by using the bit assignment as documented in Table 3-3 of the operators manual. Bits 3 thru 29 of register 8 are used for initializing the setup commands.

Another way to determine the desired value in the initialization register is to Execute Program 0 of the operating programs provided with the Async Option. Then enter the Set-up mode and manually set the desired values for START, STOP, CLOCK, etc. After all the set-up values have been entered, stop Program 0 and look at the contents of Register 8. It will contain the intialization value that represents the setup values you just selected.

2. Execute 12

The operator then sets up the hardware of the module, based on the value in the initialization register, by executing Program 12.

3. Execute 13

Next, Program 13 is executed, which arms the gate, preparing the module for actually taking signatures. This program resets all the registers within the module and then arms it to receive control signals (START, STOP, and CLOCK).

The first three steps, initializing Reg 8, Executing Program 12, and Executing Program 13 will be standard for any programmming done with the Asynchronous Signature Probe Option. The steps that follow will depend on the type of stimulus used and the desired parameters that are to be measured.

4. Stimulate Circuit

The next step is to provide some stimulus to the circuit. This can be in the form of a subroutine generated from the 9000A mainframe such as the WALK or RAMP function. Or the operator can write a program to provide the stimulus to the circuit under test. If the program uses an operator generated stimulus then the program can proceed to Step 5 because we know the stimulus has been completed.

However, if a UUT generated stimulus, such as the refresh signals in a DMA circuit, is used then the program must wait until the specified measurement window (as determined by the set ups) has occurred. This can be accomplished by using Program 17 which gets Data and Status information. This program loads status bits into register B that indicates when certain events have occurred. Bits 4, 5, and 6 indicated the occurrence of a CLOCK, START, and STOP respectfully. A loop should be written into the program following Program 13 that monitors the status, looking for the CLOCK, START, and STOP as determined by the setup commands. Following the execution of Program 17, when register B contains a 70 (bits 4, 5, and 6 set high) the stimulus is complete.

5. Execute 14

Following the stimulus you would execute the program that retrieves the particular piece of data you are interested in. To obtain the signature, Execute program 14, which places the signature data in register B. Your testing program can now compare the measured signature with the actual expected value or display the measured value to the operator for analysis.

6. Execute 15

To look at the event count, execute program 15, which stores the event count in register A. One stipulation is that program 14 must be executed before program 15 is run. Signature data must be shifted out of the register before the event count is placed in a position to be retrieved.

7. Execute 16

Program 16 places waveform information into registers A and B. Register A will have a bit set high whenever a high is measured and Register B will set a bit high when ever a low is measured. If a common bit is low in both registers, that indicates a tristate condition at that moment in time. This program must be run after programs 14 and 15 to insure that signature and event count data has been shifted out of the register. This program also overrides the data placed in registers A and B in the previous programs so that when you execute 16, you no longer have access to signature and event count information.

(continued next page)

Async measurements . . .

(continued)

8. Execute 19

With the proper levels stored in registers A and B, you would then execute program 19 to display the waveform to the operator.

Another example of using the waveform capture would be to see what level was present at a particular instant in time.

Each bit in register A or B represents a 20 nanosecond window in the data path. If you wish to see what level was present 180 nanoseconds prior to your reference or STOP signal you would look at bit 9 in the register. Suppose the correct or expected value was a logic high occurring 180 nanoseconds prior to your STOP signal. To test for its presence you would execute the following step after executing program 16: If Reg A and 100 > 0 then a high occurred 180 nanoseconds before the STOP signal.

The following is an example of a typical application of the Async Option in a Guided Fault Isolation type program testing DMA circuitry.

This program can be easily modified to fit your particular needs, whether you want to measure signatures, counts, or waveform information. Operator induced stimulus, such as programmed subroutines can be inserted prior to Label 1 and all the programming steps between Label 1 and Label 2 could then be eliminated.

The power of the Fluke 9000 Series Micro-System Troubleshooter has always been in its flexibility. The new Asynchronous Signature Probe Option, 9000A-006, adds to that flexibility and makes the Troubleshooter the most powerful testing tool available. For more information or a demonstration call your local Fluke sales office.

```
.
.
.
.
.
Execute Program 0                    Obtain the initialization values for reg. 8.

PROGRAM 90

Dpy-"Place Probe on U17 pin 22"      Prompts the operator to go to a specific test
                                     point.
Stop                                 Wait for operator to place probe and then
                                     press Continue
Reg 8 = 528                          Set initialization values

Exec 12                              Set up hardware according to values in ini-
                                     tialization register
Exec 13                              Arm Gate

Label 1

Exec 17                              Read Data and Status

If Reg B AND 70 = 70 goto 2          Check to see if CLOCK, START, and STOP
                                     have occured
GOTO 1                               If not return to Label 1 and check again

Label 2                              Stimulus has occurred at this point

Exec 14                              Get signature

Reg 1 = AF6C                         Set expected signature for test point U17 pin
                                     22
If Reg B = Reg 1 GOTO 4              Signature is good, go to next test point at
                                     Label 4
Goto 3                               Signature is bad, go to next test point at
                                     Label 3
.
.
.
```

Using the 9010A with a Tektronix 7D02

by E. Hegar

Martin Marietta Aerospace,
Denver, Colorado

This article discusses the increased diagnostic power that is available when the 9010A is used in conjunction with the Tektronic 7D02 logic analyzer. These routines were worked through on a Z80 based test tool.

Usually during debugging, little confidence exists concerning the accuracy of the design, the fabrication, the software and the documentation of the tool. The Tektronix 7D02 analyzer is extremely useful in tracing the flow of the program software, but it is limited when very fast and meaningful observations are needed, or when setting breakpoints. On the other hand, the 9010A has no timing measurement capability.

Since the 9010A is connected in parallel with the 7D02 and the 9010A substitutes for the microprocessor in the UUT, an obvious improvement would be to connect the logic analyzer to the UUT via the processor socket. Instead of installing a processor in the logic analyzer pod, replace it with the 9010A pod. A quick examination of the schematics for the 9000A-Z80 pod and the Tek Z80 Pod for the 7D02 show that this will cause no damage to either item.

Once you have made this interconnection, you will be amazed at the increase in the system's power.

Now the 9010A will allow:

- immediate downloading of object code into UUT RAM,
- full front panel control of the UUT resources,
- rapid examination of the contents of memory,
- start address specification,
- many other control and observations functions.

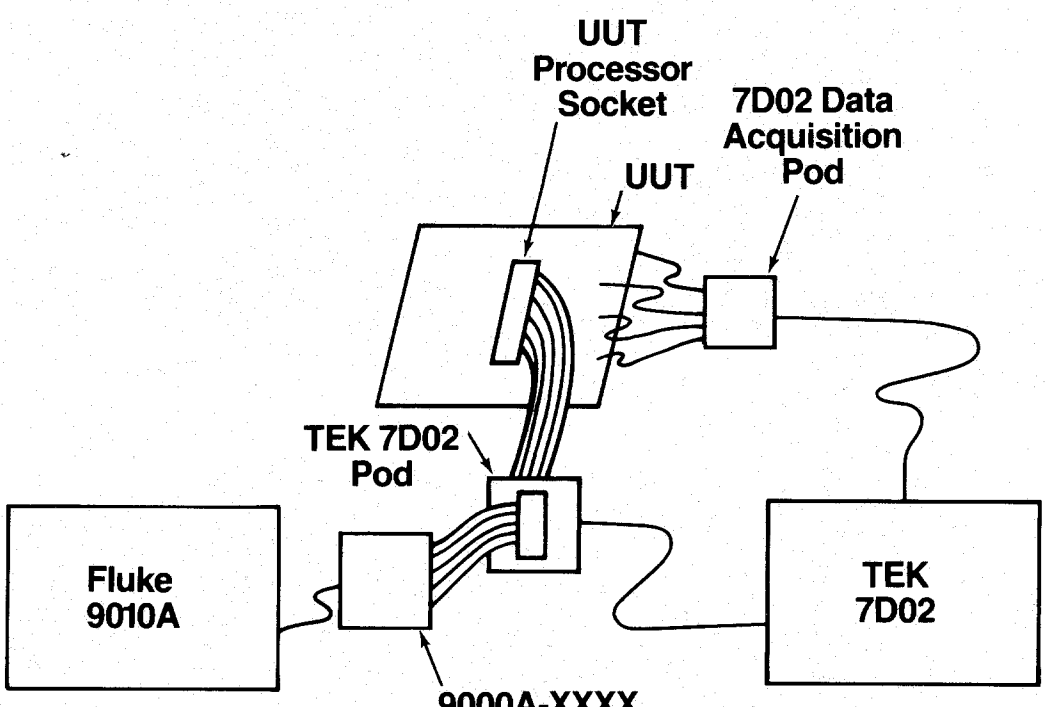
The disassembly capabilities of the 7D02 will allow:

- examination of address and data bus contents,
- precise timing measurements,
- glitch capture.

In addition, the logic analyzer is now able to look at resources not directly connected to the processor bus or isolated by intermediate resources. Since the analyzer allows masking of interrupt cycles, you

(continued next page)

Using the 9010A...
(continued)



Capabilities
- Resource Manipulation
- "Start Execution" point
- Object Code Alteration
- Custom Diagnostic S/W Entry

Capabilities
- Mnemonic Disassembly
- Triggering
- Timing Diagram
- Glitch Capture

can now examine only those states you are interested in, without having to wade through several hundred bus cycles of information.

Inclusion of a RAM flag check in the UUT ROM is another technique that has been developed for debugging the Z80 testing tool. This technique could also be applied in similar situations. Depending on the status of this flag, the UUT would either execute out of UUT ROM or it would branch to the address following the flag in UUT RAM. This allows the placing of object code other than the normal ROM code into RAM, and executing it in lieu of the ROM code. The flag, of course, could be set via the 9010A, which is also used to download the RAM object code.

The most obvious use of this technique is that it allows an engineer to put manually assembled routines in RAM, and in turn, will allow you to exercise small portions of UUT circuitry without blowing a new PROM. These small routines can be used to branch around code in the ROM that is discovered to be defective or to create alternate initialization states for simulated reset conditions.

John Fluke Mfg. Co., Inc.
P.O. Box C9090, Everett, WA 98206
Tel. 206-347-6100

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

Printed in U.S.A. P0014A-13U8604/SE EN