FLUKE®

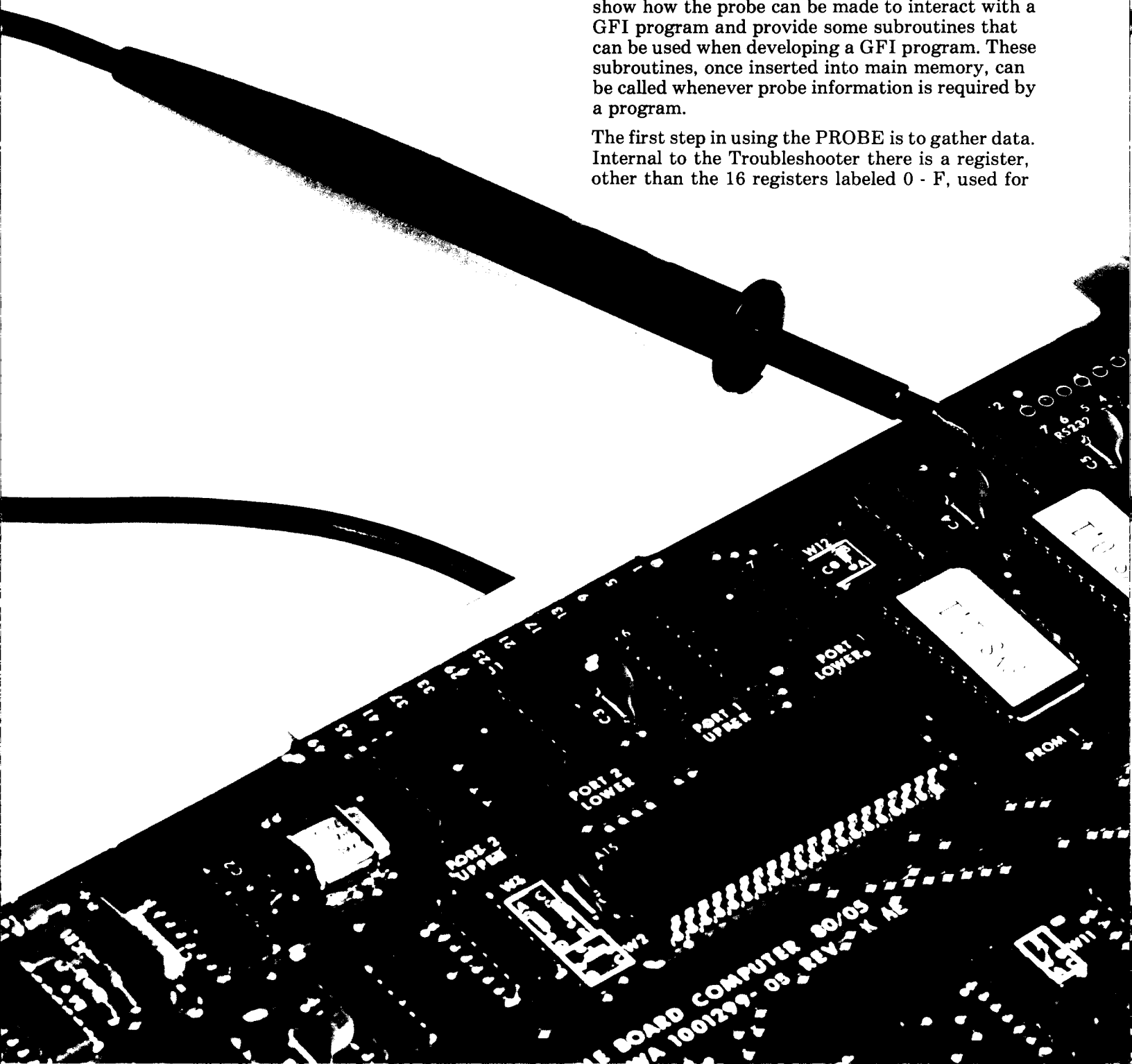
Technical Data

Application Information B0163

9010A Micro-System Troubleshooter: The Probe and the Program

The 9000-series Micro-System Troubleshooter's probe is used extensively when designing a Guided Fault Isolation (GFI) program. This bulletin will show how the probe can be made to interact with a GFI program and provide some subroutines that can be used when developing a GFI program. These subroutines, once inserted into main memory, can be called whenever probe information is required by a program.

The first step in using the PROBE is to gather data. Internal to the Troubleshooter there is a register, other than the 16 registers labeled 0 - F, used for

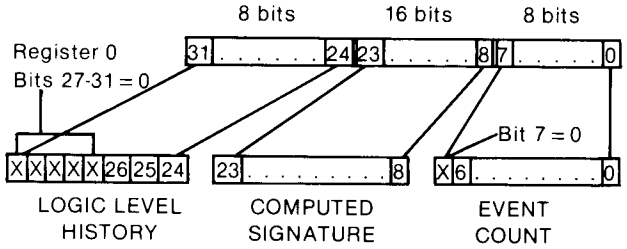


accumulating PROBE data. This bulletin will refer to this register as the probe register. Whether in the immediate or programming mode, the procedure for using this register is always the same. The three basic steps are:

READ PROBE	To clear the probe register
STIMULUS	Run stimulus by the probe. This could take the form of any Troubleshooter function i.e. RAMP, WALK, TOGGLE, etc.
READ PROBE	To terminate the gathering of data by dumping the data from the probe register into the display and register 0. In the executing mode, only register 0 is affected and not the display.

The only way that you can get probe data into a program is through Register 0. The following table shows how the probe data is formatted inside Register 0.

The probe response data is stored in bits 0-31 of Register 0 as follows:



Bit 24 = 1 if logic high detected
Bit 25 = 1 if logic tristate (invalid) detected
Bit 26 = 1 if logic low detected

NOTES:
1. Accumulated probe data is placed in Register 0 when the Read Probe step is executed.
2. Signatures may range from 0000 to FFFF.
3. Event Counts may range from 0 to 127. When a count of 127 is reached, the counter begins counting again at 0.

Figure 1

Once the data is in Register 0, the three pieces of data, LEVEL HISTORY, SIGNATURE, and COUNT have to be separated into their individual parts to be of any use.

To show this, using the Troubleshooter, assume that a READ PROBE operation has caused the value of 596EC55 to be entered into register 0.

Register 0 looks like this:

0	5	9	6	E	C	5	5
0000	0101	1001	0110	1110	1100	0101	0101

Figure 2

Using the AND function of the Troubleshooter as a mask, unwanted data from Register 0 can be masked out leaving only the desired information as a result. If, for instance, only the count is required, then all bits EXCEPT 0 thru 6 need to be masked out. To do this you can AND Register 0 with a number that has bits 0 - 6 set to one and all other bits set to zero.

0	5	9	6	E	C	5	5
0000	0101	1001	0110	1110	1100	0101	0101
ANDed with:							
0000	0000	0000	0000	0000	0000	0111	1111
Resulting in:							
0	0	0	0	0	0	5	5
0000	0000	0000	0000	0000	0000	0101	0101

Figure 3

The Troubleshooter will only accept the AND value in hexadecimal form, so 1111111 converted to hex is 7F.

To demonstrate this operation, perform the following keystrokes.

First set register 0 equal to 596EC55:

KEYSTROKE	DISPLAY
REG	REG_
0	REG0 = _
596EC55	REG0 = 596EC55_
ENTER	REG0 = 596EC55

Now to perform the MASK operation without changing register 0:

KEYSTROKE	DISPLAY
REG	REG_
1	REG1 = _
REG	REG1 = REG_
0	REG1 = 596EC55 _
AND	REG1 = 596EC55 AND _
7F	REG1 = 596EC55 AND 7F_
ENTER	REG1 = 55

This leaves the count in Register 1 and register 0 is unchanged.

To get the signature information, you can mask out unwanted bits by ANDing with FFFF00.

KEYSTROKE	DISPLAY
REG	REG__
1	REG1 = __
REG	REG1 = REG__
0	REG1 = 596EC55 __
AND	REG1 = 596EC55 AND __
FFFF00	REG1 = 596EC55 AND FFFF00__
ENTER	REG1 = 96EC00

That leaves the signature followed by eight bits set to zero. To remove these trailing zeros, the shift function should be used. After performing eight shift right operations, the signature will be left as a result.

The keystrokes for this operation are:

REG	REG__
1	REG1 = __

Press SHIFT RIGHT key eight times.

ENTER	REG1 = 96EC

Bits 24-26 are used for the logic level history in Register 0.

To separate level history, AND register 0 with 7000000. Then shift register 1, right 24 times.

This completes the procedure for separating the probe data into its three parts. The next step is to put this procedure into a program.

If this procedure is to be a subroutine which will be called by another program, then global registers will have to be used to transfer the data back to the calling program. The global registers are 8 - F. So this program uses register 8 for the COUNT, register 9 for the SIGNATURE and register A for the LEVEL.

First write the program so it only performs the separating operation and not the read probe and stimulus operations.

KEYSTROKE	DISPLAY
PROGRAM	PROGRAM __
1	PROGRAM 1__
ENTER	PROGRAM 1 CREATED
REG 8	REG8 = __
REG 0	REG8 = REG0 __
AND	REG8 = REG0 AND __
7F	REG8 = REG0 AND 7F__
ENTER	REG8 = REG0 AND 7F
REG 9	REG9 = __
REG 0	REG9 = REG0 __
AND	REG9 = REG0 AND __
FFFF00	REG9 = REG0 AND FFFF00__
ENTER	REG9 = REG0 AND FFFF00
REG 9	REG9 = __
SHIFT RIGHT	REG9 = REG9 SHR __

Press the SHIFT RIGHT key seven more times:

ENTER	SHR SHR SHR SHR SHR SHR SHR SHR
REG A	REGA = __

KEYSTROKE	DISPLAY
REG 0	REGA = REG0 __
AND	REGA = REG0 AND __
7000000	REGA = REG0 AND 7000000__
ENTER	REGA = REG0 AND 7000000
REG A	REGA = __
SHIFT RIGHT	REGA = REGA SHR __

Press the SHIFT RIGHT key 23 more times:

ENTER	SHR SHR SHR SHR SHR SHR SHR SHR

For test purposes add a display statement that displays the LEVEL HISTORY and SIGNATURE in hexadecimal, and the COUNT in decimal. The display step should look something like this:

ENTER	DPY-LVL = $A SIG = $9 CNT = @8

Also add a step at the very beginning setting register 0 to a test value.

Press the PRIOR key until display =

KEYSTROKE	DISPLAY
PRIOR	START OF PROGRAM 1

Then enter:

REG 0	REG0 = __
596EC55	REG0 = 596EC55__
ENTER	REG0 = 596EC55

Now close the program:

PROGRAM	PROGRAM 1 CLOSED-XXXX BYTES LEFT

Now execute the program:

EXEC 1 ENTER	EXECUTE PROGRAM 1__

Almost immediately, the display should read:

LVL = 5 SIG = 96EC CNT = 85

If the display is different, then check your program. Often there is a mistake in the number of Shift Right commands. Remember that when the SHR steps are displayed, every time "MORE" is pressed, eight characters are added to the display - less if it is the end of the program line.

Once this program is working properly with the test value, you should remove the first step which sets REG 0 to 596EC55 and the last step for displaying the values. Now you could add the three basic steps for probe operation.

Read Probe - Stimulus - Read Probe

As mentioned in the beginning of the bulletin, the probe operation should be used as a subroutine, for it is performed many times in different places throughout a GFI program. Putting these three steps into this program would make it very inflexible.

Having the stimulus as part of the separator routine would mean that the program could only be used for that one type of stimulus at that one address. One solution would be to put the Read Probe - Stimulus - Read Probe in the main program and then jump to the separator subroutine. Unfortunately this would not work because Register 0 is not a global register. When a program branches to a called program, Register 0 is stored and then set to 0 before entering the called program. Entering Register 0's value into a global Register (8-F) before going to the routine would work, but is not necessary.

Putting the first Read Probe and Stimulus in the main program, and then going to the separator routine where the second Read Probe is located, would give the flexibility required. This removes the requirement of using a global register for transferring data to the called program. The Probe Register, that accumulates probe data before dumping it to Register 0, is used for this transfer.

Instead of adding all three basic steps, just add a READ PROBE step at the beginning of the program. The separator routine should now look like this:

```
READ PROBE
REG8 = REG0 AND 7F
REG9 = REG0 AND FFFF00
REG9 = REG9 SHR SHR SHR SHR (8 TIMES)
REGA = REG0 AND 7000000
REGA = REGA SHR SHR SHR SHR (24 TIMES)
```

With the separator routine in this form, it only needs to be remembered that Reg 8 will equal the COUNT, Reg 9 will equal the SIGNATURE and Reg A will equal the LEVEL HISTORY. When returning to the main program from the separator routine, just test the proper register for the proper value.

An example of how to use this routine from another program is:

```
    START OF PROGRAM XX
    .
    .
    .
    READ PROBE
    RAMP @ FFFF
    EXECUTE PROGRAM 1
    IF REG9 = 96EC GOTO 1
    DPY-REPLACE U36
    .
    .
    .
1:  LABEL 1
    DPY-U36 TEST GOOD
    .
    .
    .
    END OF PROGRAM XX
```

If the GFI program cannot afford to use three registers for this subroutine, a slight modification will reduce the requirement to one register. For this example, assume Reg 8 is available. The separator routine can be told which one of the three pieces of Probe data is required by the main program.

Before executing the separator routine, the main program enters a code into a global register that will be used by the separator routine to identify which piece of probe data is required by the main program. The separator routine will peform the proper action and return to the main program with the proper value.

One of our Troubleshooter users suggested the following program.

The first significant variation, from our first separator routine, is that the second READ PROBE will be in the main program. This change was made because as long as we are going to use a global register (8) to send a code to the separator routine, we might as well use it to carry the probe data as well. There were some other considerations for this change that will become clear later. The first problem is to get the code into register 8 along with the PROBE DATA from register 0.

Refer to figure 1 and notice that bits 7, and 27 thru 31 are not used for PROBE DATA. The separator routine will use bits 28 and 29 to determine wich piece of PROBE DATA the main program needs. If bit 28 is set to a 1, the separator routine will return register 8 to the main program with the signature in it. If bit 29 is set, register 8 will contain LEVEL HISTORY, and if neither bit 28 or 29 is set to a 1, register 8 will contain the COUNT.

After the second READ PROBE, the main program will put the contents of register 0 into register 8. If COUNT is desired, nothing more needs to be done, just execute the separator routine. If SIGNATURE is required, then the main program will have to set bit 28 of register 8. This is done by ORing register 8 with a value that has bit 28 set to a 1.

```
  0    5    9    6    E    C    5    5
0000 0101 1001 0110 1110 1100 0101 0101
```

ORed with

```
0001 0000 0000 0000 0000 0000 0000 0000
```

Resulting in

```
  1    5    9    6    E    C    5    5
0001 0101 1001 0110 1110 1100 0101 0101
```

This OR value must be entered into the Troubleshooter in Hexadecimal form. 10000000 is the proper Hex value to set bit 28. Bit 29 would be set with a hex value of 20000000.

Now with the proper bit set, the separator program will test these two bits and branch to the proper section of the separator routine. The new subroutine should look like this:

```
    START OF PROGRAM 1
    IF REG8 AND 10000000 >0 GOTO 1
    IF REG8 AND 20000000 >0 GOTO 2
    REG8 = REG8 AND 7F
    GOTO F
1:  LABEL 1
    REG8 = REG8 SHR (8 TIMES) AND FFFF
    GOTO F
2:  LABEL 2
    REG8 = REG8 SHR (24 TIMES) AND 7
F:  LABEL F
    END OF PROGRAM 1
```

To use this subroutine, our main program would look like this:

```
    START OF PROGRAM XX
    .
    .
    .
    READ PROBE
    RAMP @ FFFF
    READ PROBE
    REG8 = REG0 OR 10000000
    EXECUTE PROGRAM 1
    IF REG8 = 96EC GOTO 1
    DPY-REPLACE U36
    .
    .
    .
1:  LABEL 1
    DPY-U36 TEST GOOD
    .
    .
    .
    END OF PROGRAM XX
```

Since Register 8 had bit 28 set, when the main program called the subroutine, the signature was in register 8 after returning from the separator routine.

One feature of this separator routine allows you to get all three pieces of PROBE DATA from one STIMULUS. This is due to the fact that register 0 is restored to its original value after returning from the separator routine. This is another reason why the second READ PROBE is in the main program. Just put register 0 into register 8 along with the proper code and you can get each piece of PROBE DATA by re-executing the separator program for each piece of data.

No matter which of the two routines you use, they both should work in any type of GFI program you design.