9100 SERIES • SOLUTIONS

9100A SYSTEMS TRAINING – PART II

FLUKE

John Fluke Mfg. Co., Inc.

Customer Support Services
P.O. Box 9090 MS 239D
Everett, Washington 98206
(206) 347 6100 or
(800) 44-FLUKE Ext 73

Rev. November 20, 1991

FLUKE®

Fluke Corporation Course Critique

INSTRUCTIONS
On the first day of this class please fill out Part I and your expectations in Part II. Upon completion of the class, rate those expectations according to how well they were met. Continue down through Part III. There is room in Part IV for your comments and suggestions. Thank you.

PART I

NAME (PRINT AS YOU WOULD LIKE IT SPELLED ON YOUR CERTIFICATE)	
YOUR COMPANY'S NAME	
CLASS TITLE	
CLASS LOCATION	CLASS DATE
INSTRUCTOR'S NAME	

TECHNICAL BACKGROUND

Give a brief listing of experience/education to summarize your level of understanding related to this class.

Form No. B513 2/95

PART II

	EXCELLENT		SATISFACTORY		POOR
1. What are your expectations from this class? ______ ______ ______	*How well were these expectations met?* ⑤	④	③	②	①
2. How do you expect this class to relate to your present job? ______ ______ ______	*Will the skills learned in this class be useful in your job?* ⑤	④	③	②	①
3. What level of expertise do you expect to gain from this class? ______ ______ ______	*How well did this class perform to that level?* ⑤	④	③	②	①

PART III

1. The objectives were clearly identified and fully explained.	⑤	④	③	②	①
2. The instructor explained the technical information clearly.	⑤	④	③	②	①
3. The instructor encouraged questions and student participation.	⑤	④	③	②	①
4. The instructor adjusted the pace according to the needs of the class.	⑤	④	③	②	①
5. The information was presented in logical order.	⑤	④	③	②	①
6. The length of the class was right for the level of instruction.	⑤	④	③	②	①
7. Manuals and visual aids were clear and understandable.	⑤	④	③	②	①
8. The classroom facilities were adequate.	⑤	④	③	②	①

PART IV

Comments and suggestions:

______

______

______

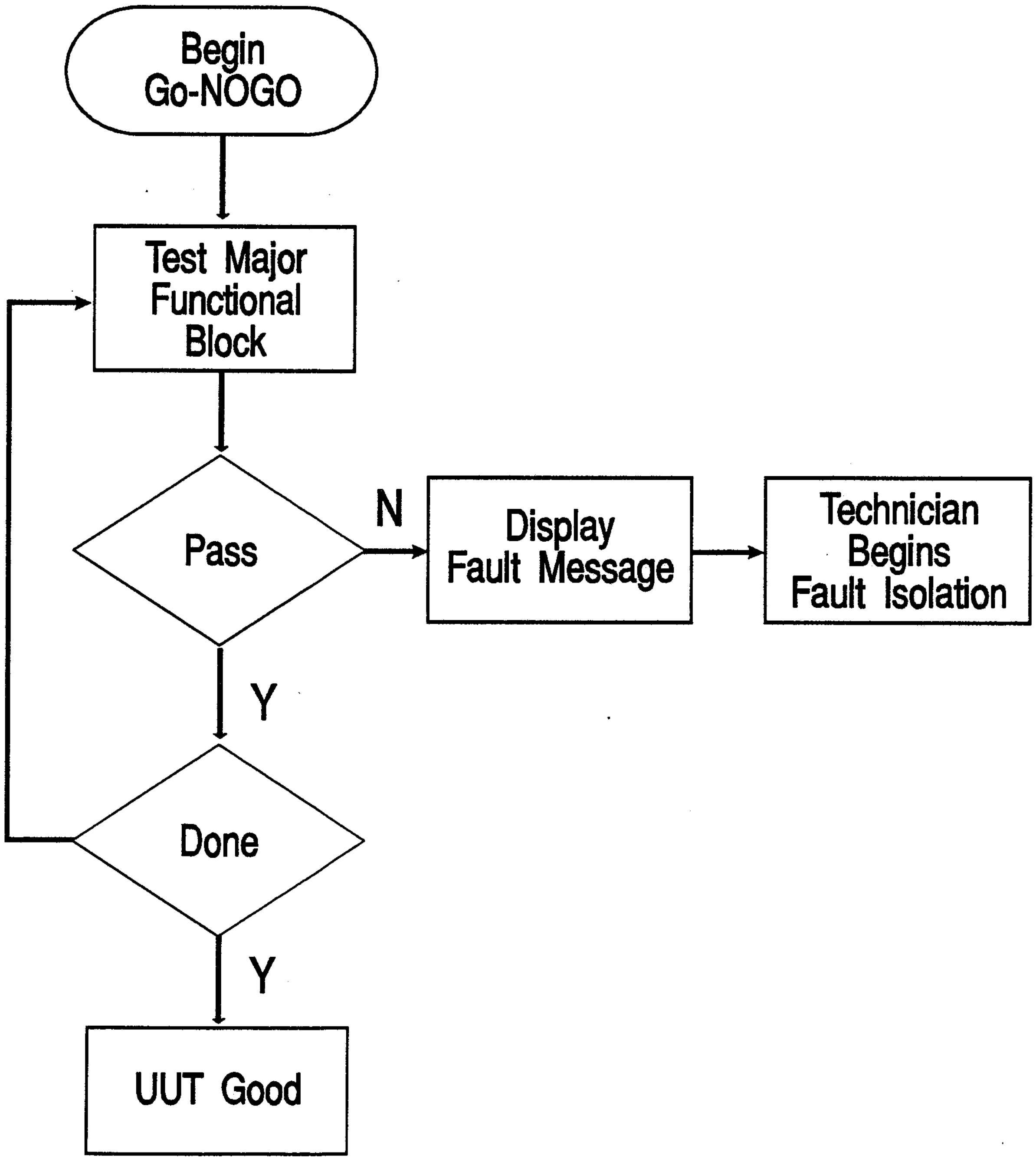
______

Table of Contents

Section 1
9100A Part I Review

Section 2 UFI: Microprocessor Emulation

The exercises you perform in this section illustrate the capabilities and ease of use of the 9100A Series Digital Test Station after software development is complete. At the completion of this course, you will have performed all the necessary steps to recognize faults on the demo UUT and troubleshoot them using the Unguided Fault Isolation (UFI) and immediate mode methods.


Begin Go-NOGO
Test Major Functional Block
Pass
N
Display Fault Message
Technician Begins Fault Isolation
Y
Done
Y
UUT Good

Executing a Functional Test

The software you are about to execute follows the flow illustrated on the previous page. The exercise performs a major functional test on the UUT. If the test passes, the UUT is either declared good, or another major functional block will be tested. If a major functional block fails, the error is displayed. This method is commonly referred to as "go-nogo" testing. The technician can then begin fault isolation.

With no fault switches set on the Demo-Trainer UUT (Switch pack 1, switches 1 through 8 "on", all other switches "off"), press EXEC on the 9100A Applications keypad. The Mainframe displays the following menu:

```
EXEC  UUT   PROGRAM
```

Select the UUT name "CD" using HELP

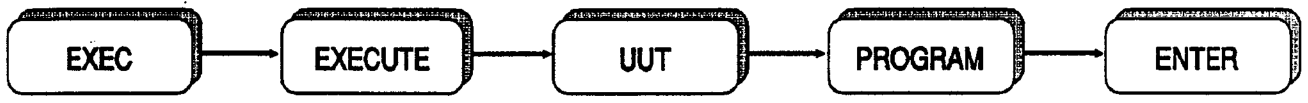
Select the PROGRAM name "GO_NOGO" using HELP.

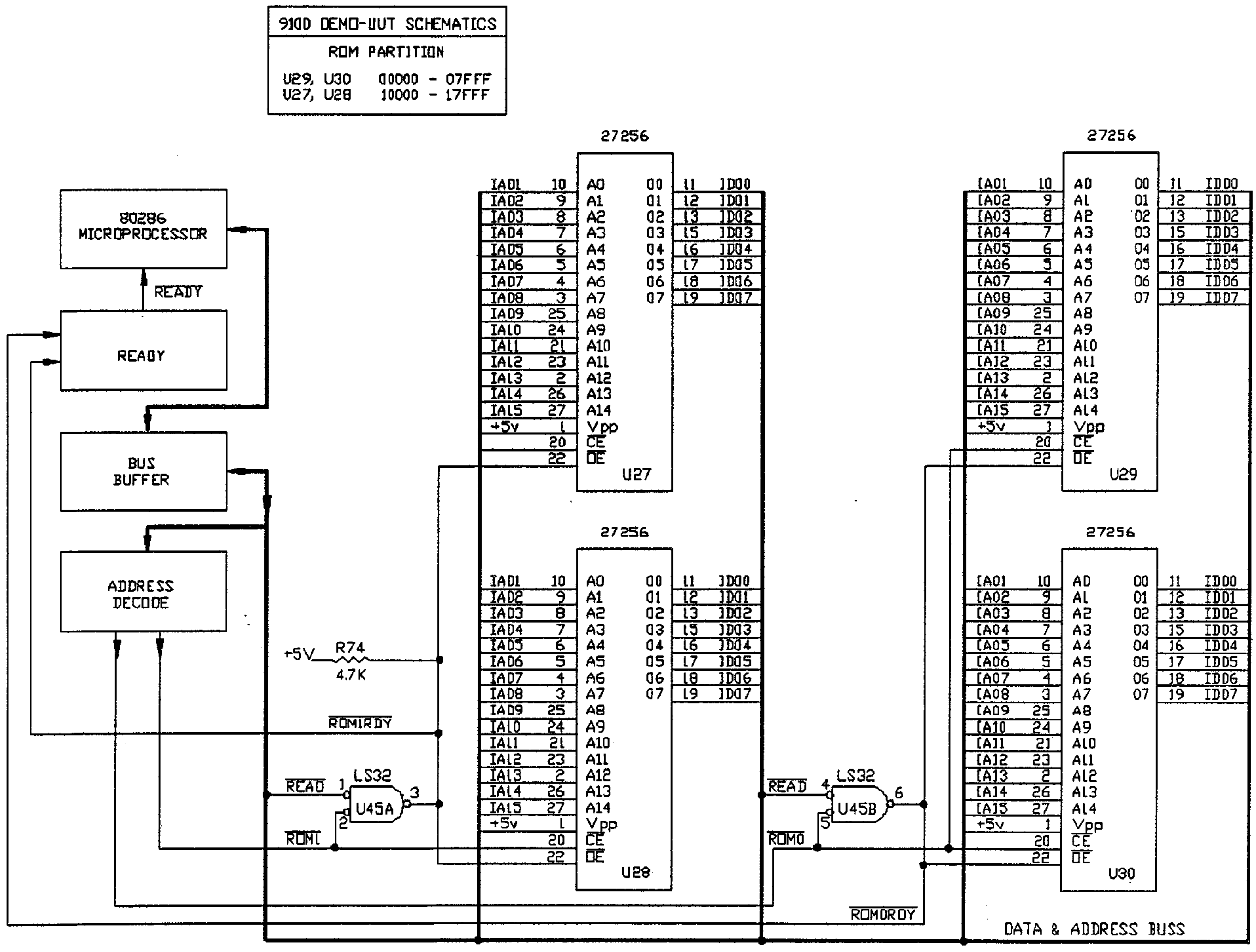
The following should now be displayed on the Mainframe:

```
EXEC UUT CD PROGRAM GO_NOGO
```

Begin the program execution by pressing ENTER YES on the 9100A Applications keypad.

During execution of the program, a go-nogo test will be run on the ROM circuit. U27, U28, U29, and U30 are functionally tested. Since no errors are reported, the ROM circuit is declared "good" by simply ceasing execution (No News is Good News!).


EXEC
EXECUTE
UUT
PROGRAM
ENTER


9100 DEMO-UUT SCHEMATICS
ROM PARTITION
U29, U30 00000 - 07FFF
U27, U28 10000 - 17FFF
80286 MICROPROCESSOR
READY
READY
BUS BUFFER
ADDRESS DECODE
27256
U27
U28
U29
U30
+5V
R74
4.7K
ROMIRDY
READ
ROM1
LS32
U45A
READ
ROM0
LS32
U45B
ROMORDY
DATA & ADDRESS BUSS

Troubleshooting From a Functional Test

To illustrate what happens when a fault is encountered, set fault switch SW1-2 to it's fault position (switch off) then execute the program as shown in the previous exercise.

Now during program execution, a fault is detected while testing the U27 functional block. The following error message will appear on the Mainframe display:

```
    Testing from F0000 to FFFFE
    all ROM data bits stuck high
```

To continue the test, press CONT on the 9100A Applications keypad.

Notice the program resumes the go-nogo test until complete.

From the error message we see that since all data lines are stuck, U27 most likely has a control line problem. This could be caused by several things but lets start at the source of the problem by performing a test of the ROM device itself.

Since all of the UFI programs and responses have been done for us, we'll begin by performing a UFI test of control lines at U27.

Press the GFI key on the Applications keypad. The following should be displayed:

```
        RUN GFI UUT CD REF  PIN
```

Select REF "U27" and PIN "22" (Pin 22 is one of the control lines at U27). The display should look like this:

```
    RUN GFI UUT CD REF U27 PIN 22
```

Press ENTER YES.

What were the results? By analyzing the result, what would be the next logical step?

Find the fault by backtracing.

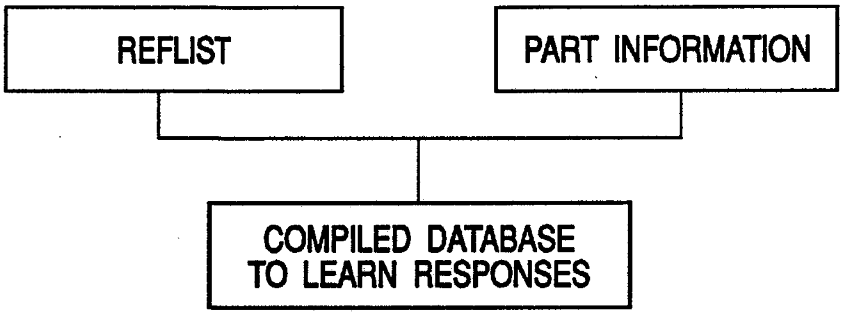


Figure 1

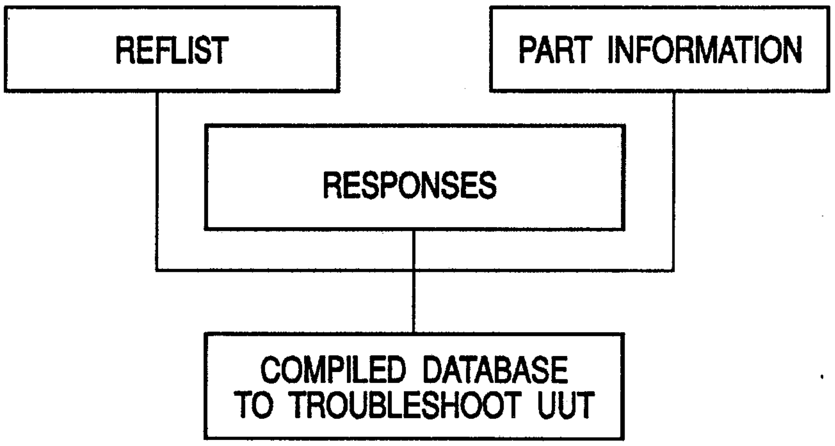


Figure 2

Creating a UFI Database

UFI is designed so that an experienced technician can use the 9100A's GFI pin testing capability and decide from the results were to test next. The UFI operator may use a combination of functional tests, keypad commands, and UFI to troubleshoot a UUT. You already have a number of functional tests designed for the Trainer UUT and you are now familiar with testing from the operators keypad. In this section, you will learn all the steps involved in building a UFI database.

To illustrate the steps required in creating a UFI database, we are going to use the Trainer UUT and together perform all the steps to develop UFI for the data bus out from the microprocessor.

Figure 2-1 shows the structure of the UFI database used to learn responses. It contains the following types of information:

- Part Descriptions
- Reference Designator List (REFLIST)

Figure 2-2 shows the structure of the UFI database used for troubleshooting. It contains the following types of information:

- Part Descriptions
- Reference Designator List (REFLIST)
- Stimulus Program Responses

Creating a UFI Database

1. Enter Part Descriptions

2. Enter Reference Designator List

3. Compile to Learn Responses

4. Develop Stimulus Routine

5. Write Stimulus Program

6. Learn Stimulus Responses

7. Compile to Troubleshoot UUT

8. Perform a Summary

9. Test

Part Description

The part description identifies the package type, number of pins, and the function of each pin (such as input or output). The related input pins can be designated for each output.

Reference List

The Reference Designator List (REFLIST) identifies the part reference (U3), the type of part (74245) and the measurement device (I/O MODULE) used to test that reference.

Compile to Learn Responses

Once the part descriptions and reference designator list are completed, you must perform a compile to learn responses operation to initialize the database.

Response File

The Stimulus Program Response file contains the known good responses of nodes which the corresponding stimulus program stimulates.

Stimulus Programs

In addition to the items in the database, UFI requires a set of TL/1 stimulus programs. The programs are used to generate the responses stored in the database. Each stimulus program will have a corresponding stimulus response file.

Compile to Troubleshoot

Stimulus programs, stimulus responses and reference designators and the UFI database are all stored in the UUT directory. The part descriptions are stored in a part library (PARTLIB) at Userdisk level for use by all UUTs.

On the facing page is a list of steps that must be done to create a UFI database for troubleshooting. The only item that is optional is the Summary. All other items must be performed in the order presented.

Learn Responses

Before you can learn responses for UFI or GFI you must first set up the database with some initial information. In the following exercise, Steps 1, 2, and 3 are necessary to create a database usable by the Learn Responses Utility.

Step 1: Part Descriptions

List the part types and references involved when the microprocessor is putting data out to the bus.

```
    EDIT: /hdr/partlib          TYPE: LIBRARY
```

A directory of part descriptions already in the parts library (PARTLIB) are shown. The types are generic in that a 74245 is used for 74LS245, 74ALS245, 74HCT245, etc. because the pin configuration is the same.

Check the directory for all parts that you have listed.

Are the parts in the library?

To illustrate how to create a part description, let's edit a part called "dummy".

```
    EDIT: dummy                 TYPE: PART
```

How is the pin configuration selected?

How is the function of each pin selected?

QUIT editing the part "dummy" and when asked if you wish to save your changes, select NO.

Where did you return to?

Step 2: Reference Designator File

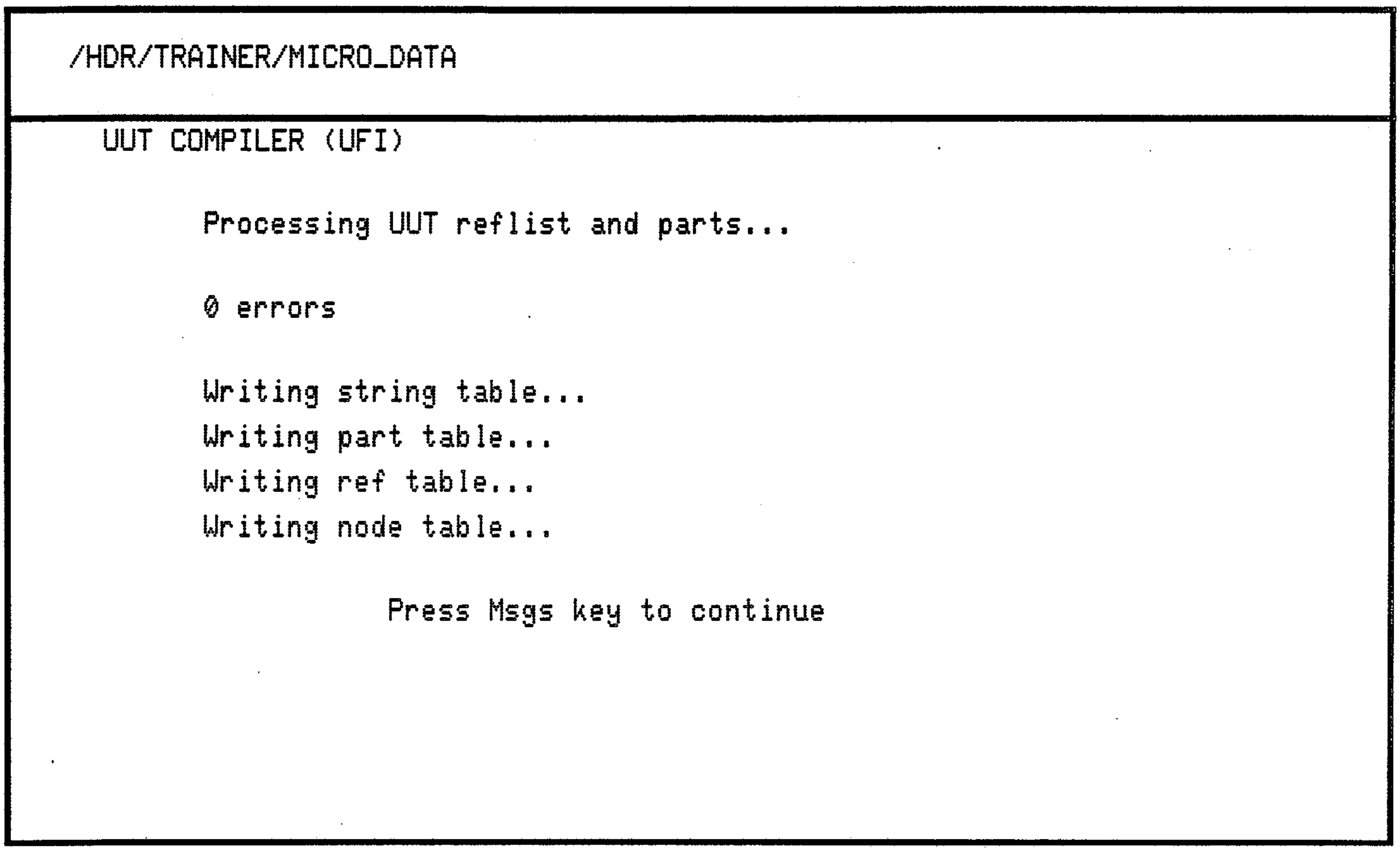
The Reference Designator (REFLIST) file is used by GFI/UFI to cross reference the part to the reference designator. It also indicates which measurement device to use for GFI/UFI.

Edit the Reference Designator List:

```
    EDIT: /hdr/trainer/reflist    TYPE: REF
```

Are all of the required references listed and are the correct testing devices selected?

QUIT editing the REFLIST. Where did you return to?



```
/HDR/TRAINER/MICRO_DATA

  UUT COMPILER (UFI)

        Processing UUT reflist and parts...

        0 errors

        Writing string table...
        Writing part table...
        Writing ref table...
        Writing node table...

                   Press Msgs key to continue
```

Screen 1

Step 3: Compile to Learn Responses

At UUT level, [F3] (Compile) is used to perform TL/1, GFI or UFI compilations.

Press [F3]. The following prompt will appear:

```
    COMPILER TYPE TL/1
```

Use [Field Select] to select UFI. Press [RETURN].

You are promped for the database type. Enter:

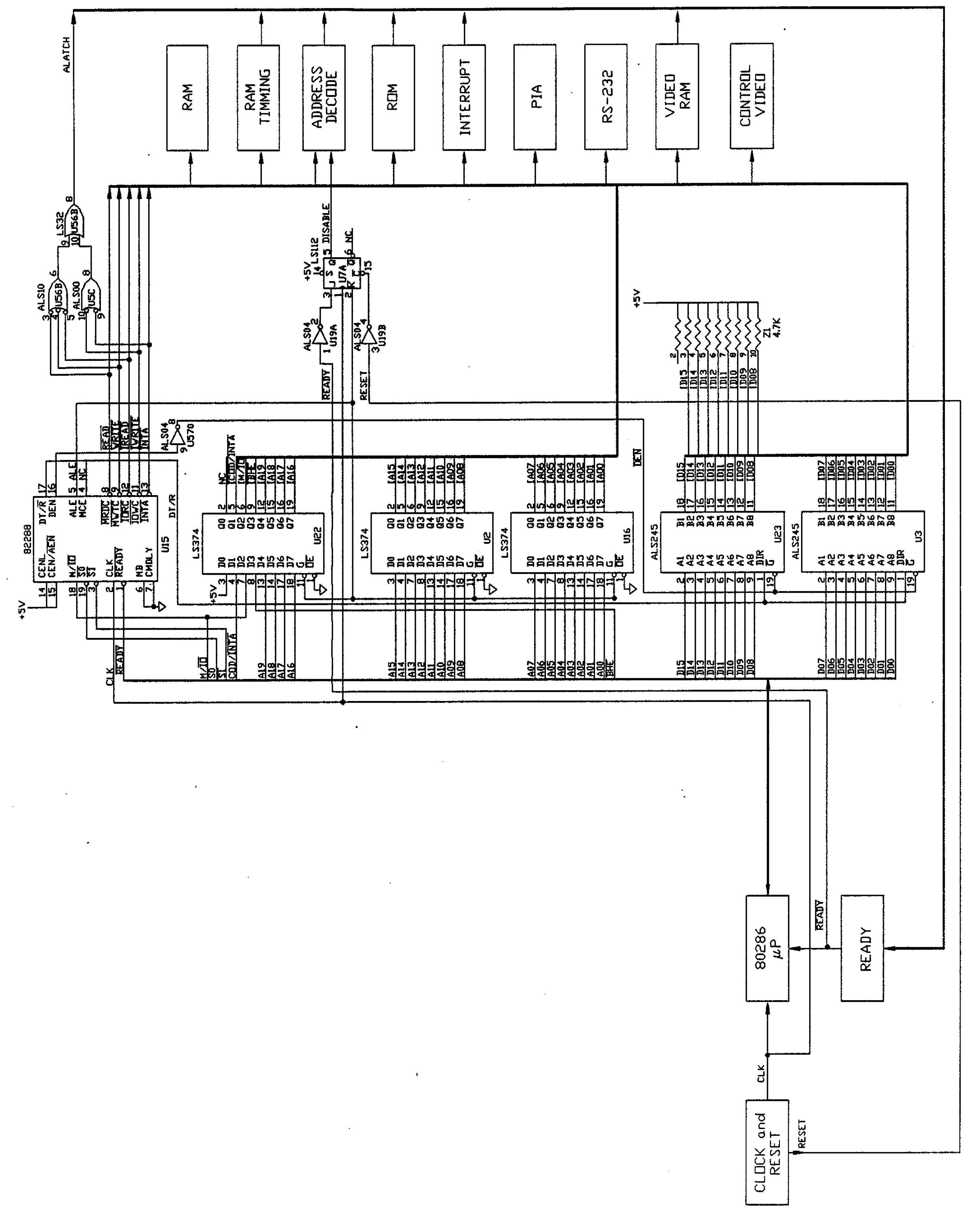
```
COMPILE database to TROUBLESHOOT UUT
```

Use [Field Select] to select LEARN RESPONSES. Press [RETURN] to begin compiling the database.

If the compile is successful, the screen 1 will be displayed on the programmers display.

Press [Msgs] on the programmers keyboard to return to the directory.

IMPORTANT: The Step 3 sequence must be performed any time there is a change to the parts library or the reference designator list.


ALATCH
RAM
RAM TIMMING
ADDRESS DECODE
ROM
INTERRUPT
PIA
RS-232
VIDEO RAM
CONTROL VIDEO
LS32 U56B
ALS10 U56B
ALS00 U5C
LS112 U7A
DISABLE
ALS04 U19A
ALS04 U19B
READY
RESET
ALS04 U57D
Z1 4.7K
+5V
82288 U15
LS374 U22
LS374 U2
LS374 U16
ALS245 U23
ALS245 U3
DEN
CLK
80286 μP
READY
CLOCK and RESET

Step 4: Develop the Stimulus Routine

A Stimulus routine controls the beginning and end of a measurement period and the activity on the node that causes a predictable and repeatable response. The stimulus routine also determines the direction of data flow. The stimulus routine NEVER evaluates the results of the measurement if it's used for UFI or GFI.

Stimulus routines are most often developed in the immediate mode.

Data Stimulus Commands

- rampdata
- toggledata
- rotate

Address Stimulus Commands

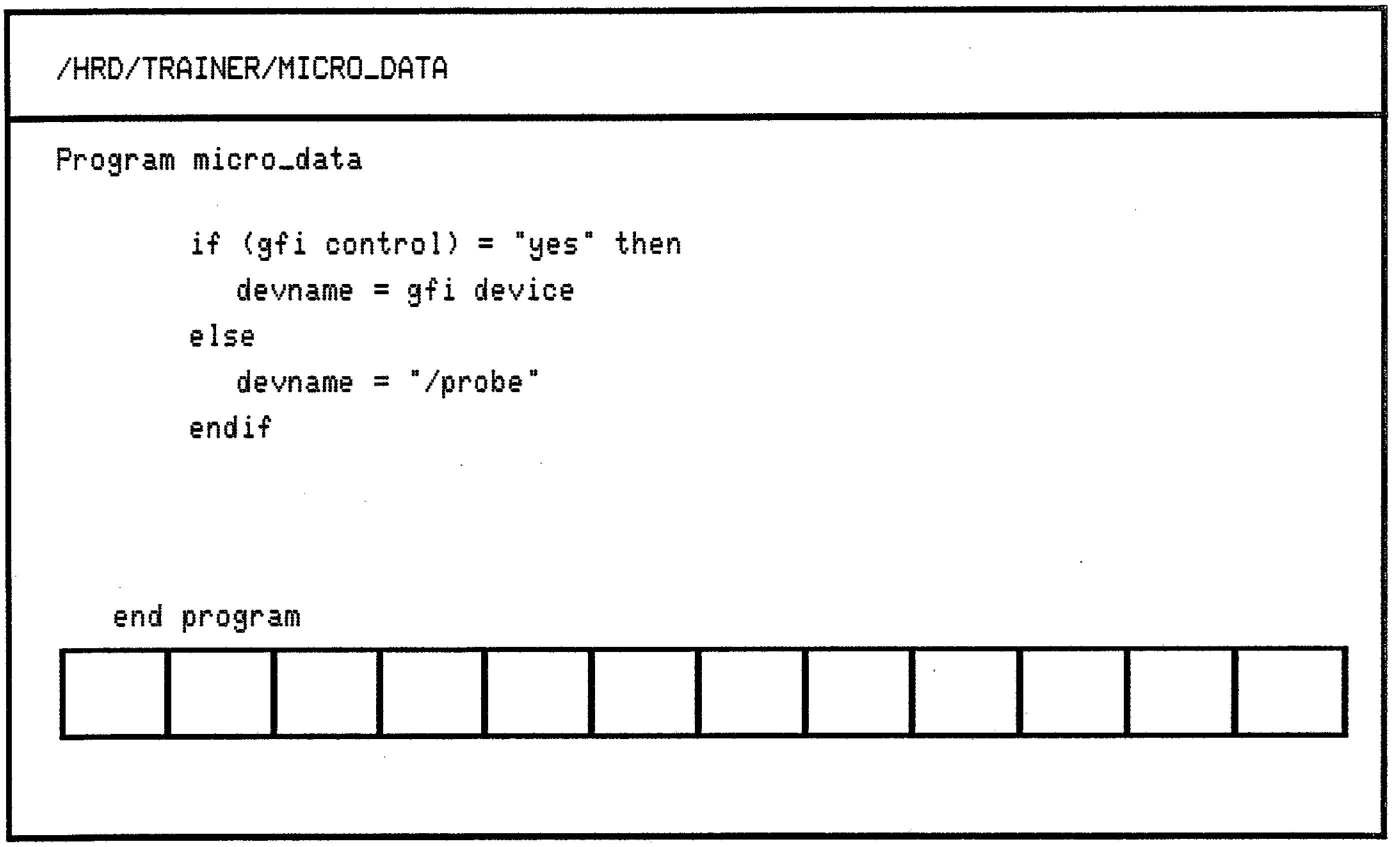
- rampaddr
- toggleaddr

The next step in our UFI process is to develop our stimulus routine for the data bus out. This step can be done at any time.

Write down the steps developed in immediate mode to stimulate the data bus from the microprocessor.

ARM

SHOW



```
/HRD/TRAINER/MICRO_DATA

Program micro_data

        if (gfi control) = "yes" then
           devname = gfi device
        else
           devname = "/probe"
        endif



    end program
```

Screen 2

Step 5: Stimulus Programs

Once the stimulus routine has been developed, you can begin writing the stimulus program.

In some instances, certain inputs are driven by more that one source. If this is the case, you need to write additional programs for each of those sources.

When writing a stimulus program used for more than one node, be sure that the measurement for each node is unique. For example, when you are testing a RAM chip, you may inadvertently place the measuring device on the wrong point. If the measurements where all the same for each RAM output, the test may still pass.

It is very important that a stimulus program not simultaneously exercise multiple sources on a net. Each source capable driving any given net must have it's own stimulus program. For stimulus programs that drive the data bus, ensure that those programs clearly separate the data direction and signal sources.

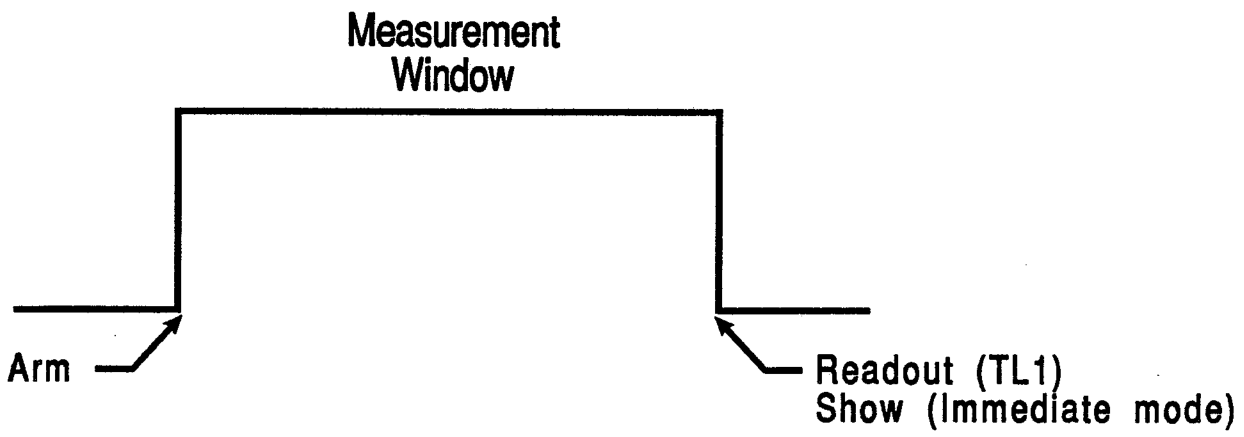
Stimulus Program Structure

The structure of a stimulus program is broken down into three basic parts.

- Define
- Setup
- Stimulus routine

The Define portion of a stimulus program describes the measurement device to be used. Note the illustration on the facing page. If the gfi control command returns the string "yes", the measurement device will be assigned by the gfi device command. If gfi control returns a "no", the program assigns the measurement device as "/probe".

The Setup portion of a stimulus routine contains the required setups for address space, circuit initialization, pod and the measurement device setups.


Measurement Window
Arm
Readout (TL1)
Show (Immediate mode)

The Stimulus routine contains the arm, stimulate, and readout commands. The arm command begins the measurement, the stimulus provides the activity required, and the readout command ends the measurement. Remember, The stimulus routine causes activity at a node to obtain a predictable, repeatable response.

A very important part of writing a stimulus program is naming it. Especially when they are used for UFI. Remember, the stimulus routine determines the direction of data flow. When a node fails under UFI, the name of the program that stimulated the failing node is given. The technician can use this information to aid in determinating where to test next. The following stimulus program is named "micro_data". The name implies that the stimulus exercises data lines from the microprocessor.

Write the stimulus program:

```
EDIT: /hdr/trainer/micro-data         TYPE:
PRORAM
```

Enter the routine that determines if the program has been called by GFI/UFI. If so, then let the database determine the measurement device. If not, assign the measurement device as "/probe".

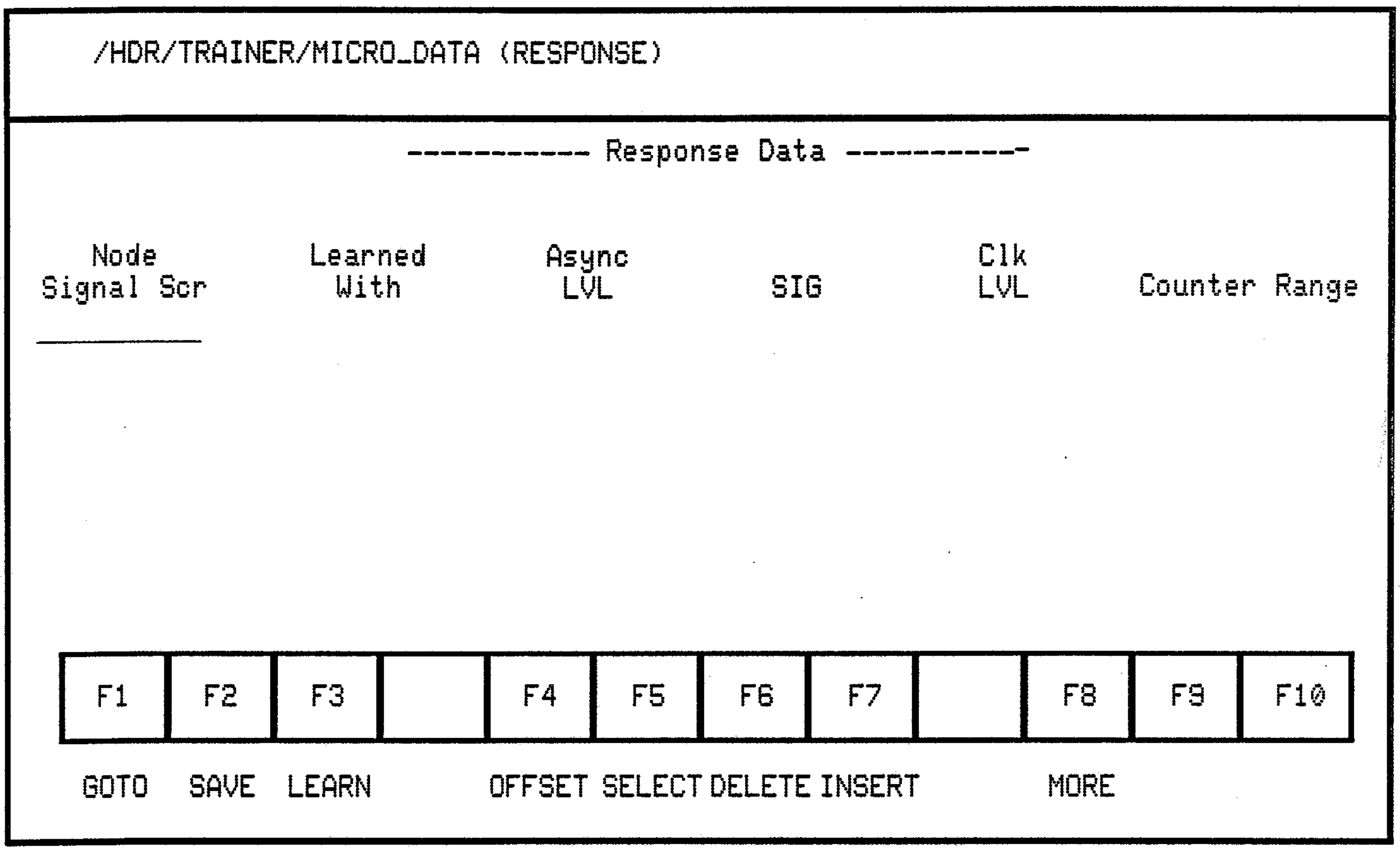
Enter the TL/1 commands required to setup the address space, circuit, pod, and measurement device.

Enter the data bus stimulus routine that was developed in Step 4.

Check the program by pressing [F10] (CHECK) and selecting the current TL/1 options.

After all errors (if any are correctd, DEBUG the program.

[QUIT] the debugger, [SAVE] ,then [QUIT] the program. At UUT level, perform a TL/1 compile using the current TL/1 options.



```
   /HDR/TRAINER/MICRO_DATA (RESPONSE)

                         ----------- Response Data -----------

    Node          Learned       Async                     Clk
  Signal Scr       With          LVL          SIG         LVL      Counter Range
  __________


      F1    F2    F3          F4     F5     F6     F7          F8    F9    F10

     GOTO  SAVE  LEARN       OFFSET SELECT DELETE INSERT       MORE
```

Screen 3

Step 6: Stimulus Response Files

The purpose of the stimulus response file is two-fold. First, the response file will contain responses obtained from a known-good UUT. Second, because the response file has the same name as the stimulus program, it provides the link between the stimulated nodes and the program that stimulates them.

Edit the stimulus response file:

```
EDIT: /hdr/trainer/micro_data  TYPE:RESPONE
```

The display should resemble Screen 3.

In the Node Signal Src column enter the U3 outputs as follows:

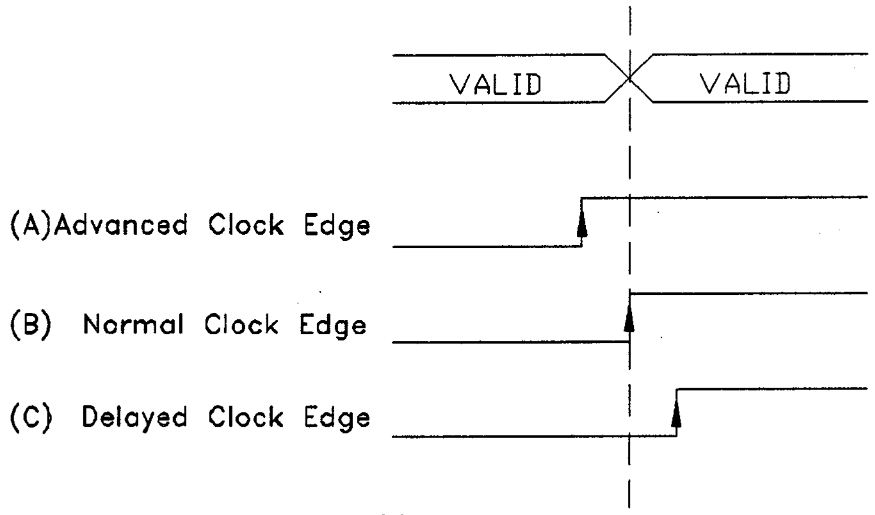
U3-11
U3-12
U3-13
U3-14
U3-15
U3-16
U3-17
U3-18

NOTE: Use [↓] to move to the next line.

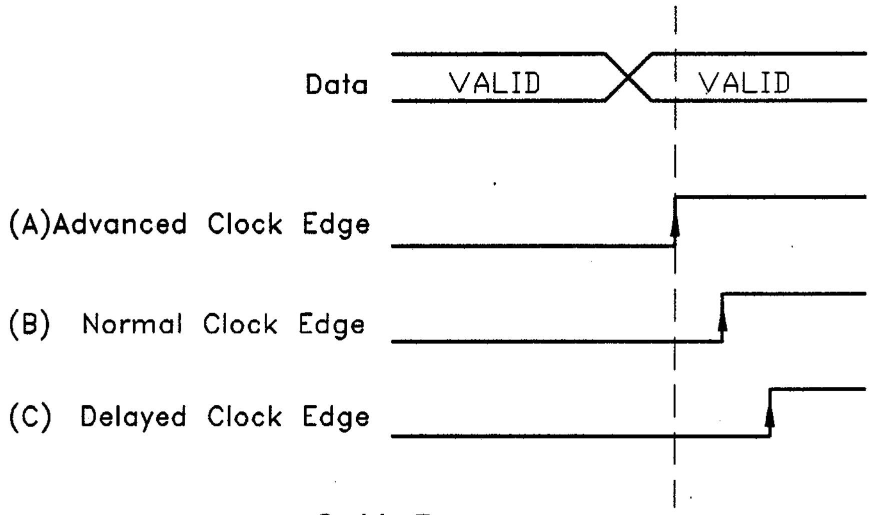
The Learned With column will contain the UFI or GFI device upon completion of the Learn. This column cannot be edited.

The SIG, Async LVL, Clk LVL and Counter Range columns will contain the appropriate responses for each node. These columns can be edited. At Version 6.0 the LVL columns may contain the ? character for "don't care" levels.

The Counter Mode column will display the counter mode ie. TRANS or FREQ upon completion of the Learn. This column cannot be edited.



Unstable Response



Stable Response

Press [F8] (MORE). The MORE function key will cause the 9100 to display another column called Priority Pin. This column is used for GFI only and identifies the pin to be tested next if the response for the current node does not match the expected response.

Press [F8] again to return to the response screen.

Press [Begin File] to place the cursor at the first node U3-11.

Press [F3] (LEARN) function key on the programmer's keyboard. When prompted:

```
    USE CURRENT LEARN OPTIONS [YES]
```

use the [Field Select] key to select NO then press [RETURN].

`Learn Using:` Indictes if the next LEARN is for UFI or GFI. A UFI learn gathers responses using only the measurement device specified by the signal source pin. A GFI learn examines all other pins on the node and gathers responses using both measrement devices if required.

Use the [Field Select] key to select UFI.

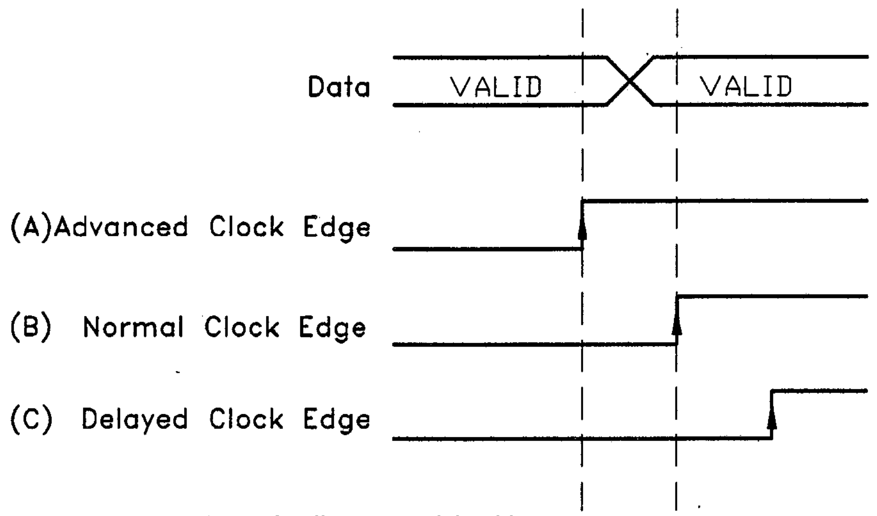
`Learn for:` Indicates the number of pins covered by LEARN. The following three options may be selected using [Field Select]:

ONE NODE The line of the edit window where the cursor is locatd is examined. The node entered as the signal source is learned.

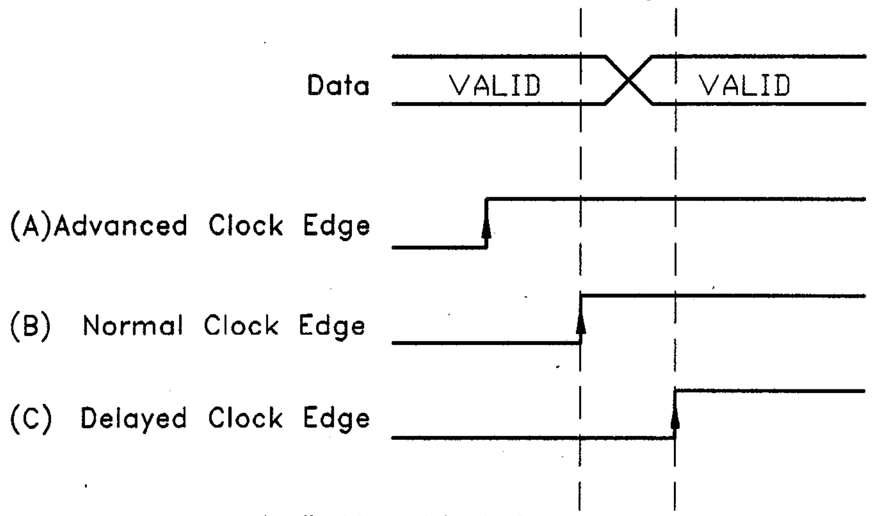
ONE REF The line of the edit window where the cursor is located is examined. The signal source pin for every line with the same reference designator is learned.

ALL REFS Every signal source in the response file is learned.

Select ONE NODE.



Marginally Unstable (-) Response



Marginally Unstable (+) Response

Repeat Stimulus: The LEARN operation is performed three times to insure the marginal timing situations are detected. This selection controls how many times the LEARN operation isrepeated. A number between 1 and 99 may be entered. The multiple executions of the LEARN utility is used to tell if the responses are:

- Stable
- Unstable
- Marginally Stable

Enter 1 .

Press F3 (LEARN) to learn the responses for U3-11.

Learn the remainder of U3 responses.

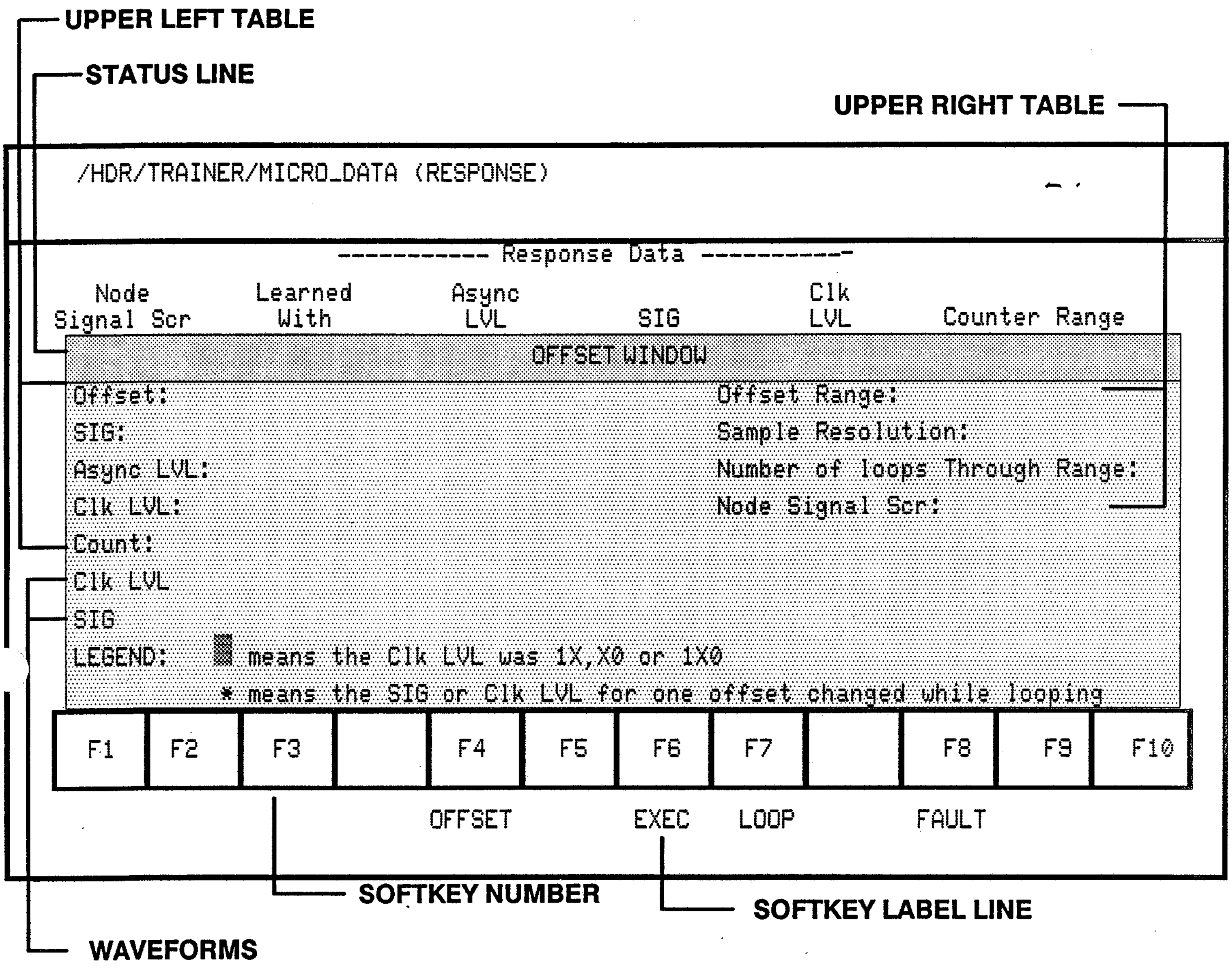
Enter the U23 outputs below the U3 entries and learn their responses.

Press F2 (SAVE).

Merging Responses

LEARN merges all three responses and if the same response is measured each time, the response is recorded. Signatures will be shown intensified.

- A signature recorded with a "-" indicates the advanced clock signature was different.
- A signature recorded with a "+" indicates the delayed clock signature was different.
- A "*" indicates all three responses were different.



```
/HDR/TRAINER/MICRO_DATA (RESPONSE)

                    ----------- Response Data ----------
   Node          Learned        Async                       Clk
Signal Scr        With           LVL          SIG           LVL       Counter Range
                                  OFFSET WINDOW
Offset:                                             Offset Range:
SIG:                                                Sample Resolution:
Async LVL:                                          Number of loops Through Range:
Clk LVL:                                            Node Signal Scr:
Count:
Clk LVL
SIG
LEGEND:  ▨ means the Clk LVL was 1X,X0 or 1X0
         * means the SIG or Clk LVL for one offset changed while looping
 F1    F2    F3         F4    F5    F6    F7          F8    F9    F10
                        OFFSET      EXEC  LOOP        FAULT
```

Screen 4

Response File "OFFSET" Window

Press the OFFSET softkey F4 to enable the Offset feature.

Notice that the Offset Window (Screen 4) overlays the response file editor screen.

Upper Left Table:

`Offset:`	The offset for the current sample
`SIG:`	The CRC signature for the current sample.
`Async LVL:`	The asynchronous level for the current sample.
`CLK LVL:`	The Clocked level for the current sample.
`Count:`	The count range for the current sample.

Upper Right Table:

`Offset Range:`	The range of offsets being sampled.
`Sample Resolution:`	The resolution between samples (1 = high resolution, 9 = low resolution).
`Number of Loops Through Range:`	Number of time the offset range has been swept.
`Node Signal Scr:`	The pin being sampled.

Waveforms:

Clk LVL:

The clocked level waveform displays the three states: high, low, tri-state, and combinations of these states.

SIG: The CRC signature waveform uses two symbols:

= The parallel lines indicates that the sampled signature is the same as the previous sampled signature.

[The left bracket indicates the that sampled signature is different that the previous sampled signature

♦ The diamond shows the original offset

↑ The arrow points to the current sample in the SIG waveform. It moves automatically during EXEC or LOOP. When not in EXEC or LOOP, the arrow may be moved to any sample by use of the left-arrow and right - arrow keys. As the arrow is moved, the upper left table displays the response data for the pointed to sample.

Legend:

The Legend at the bottom of the screen shows the meaning of special symbols used in the waveforms.

The reverse video indicates the following states:

- Tri-state and high
- Tri-state and low
- Tri-state and high and low

The Upper Left Table will indicate which levels actually occured at the particular sample.

The (*) indicates which levels actually occured at the particular sample.

Softkeys:

OFFSET:	Toggles the offset window on and off.
EXEC:	Starts one sweep of the offset range. The TL/1 program is executed and samples taken at various offsets within the range.
LOOP:	Causes the sweeping of the offset range to be repeated until the looping is stopped.
FAULT:	Toggles the fault window on and off.

To use the Offset Window, determine which node to check for offsets.

Place the cursor on the appropriate line and press the OFFSET [F4] softkey.

Press the EXEC softkey. The following prompt appears:

```
EXECUTE PROGRAM ___________
```

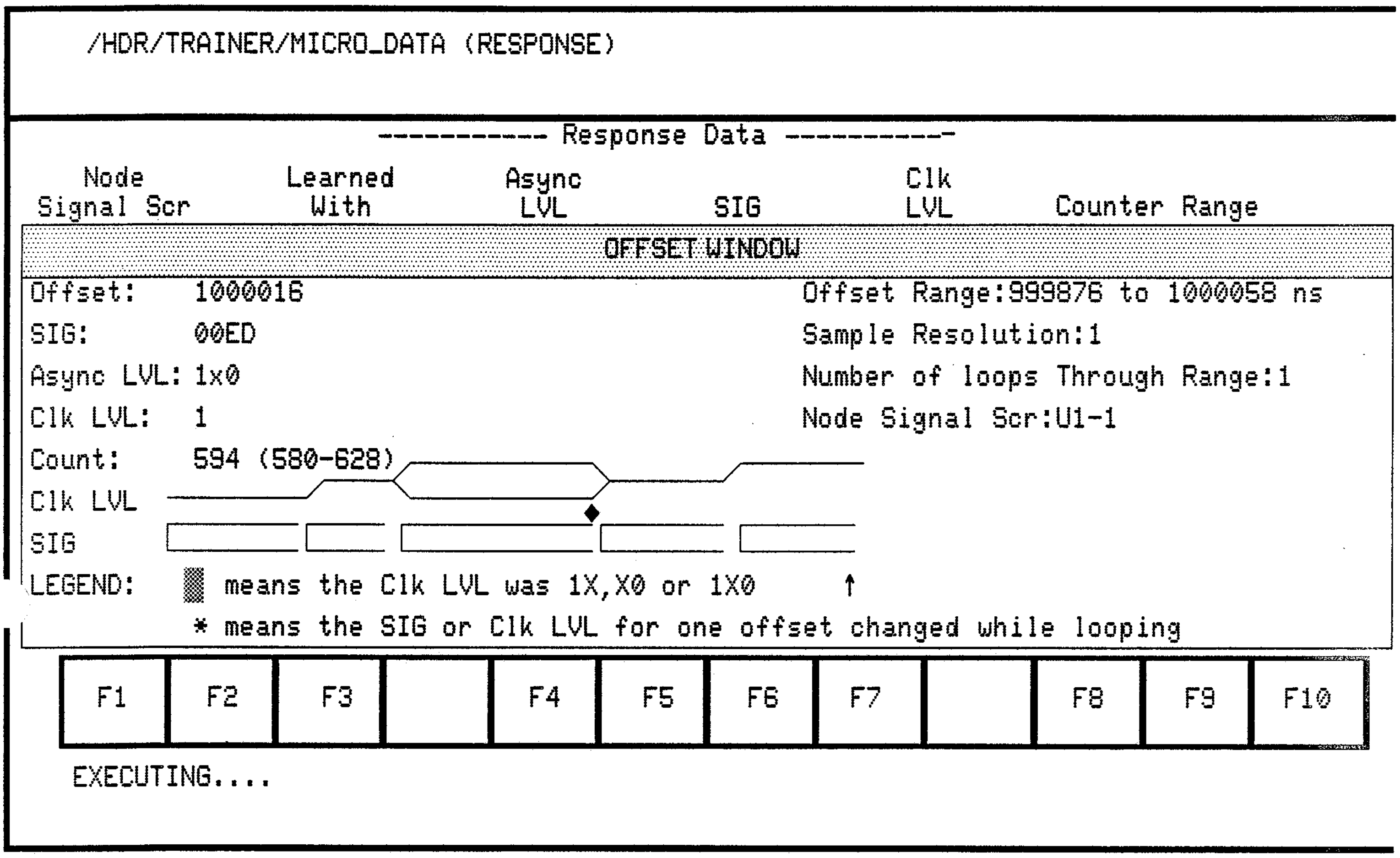
The name of the stimulus program appears on the prompt line as the default.

NOTE: The stimulus program cannot contain a setoffset command when executing from the Offset Window.

Press [RETURN]. The following prompt appears:

```
OFFSET RANGE (ns) FROM _____________
```

The number entered represents nanoseconds and is biased by the value 1000000 (decimal). The default value of 999000 appears on the prompt line (1000000 - 1000) to insure the sampling begins at the lowest offset for the measurement device.



```
   /HDR/TRAINER/MICRO_DATA (RESPONSE)

                    ----------- Response Data -----------
    Node          Learned        Async                  Clk
 Signal Scr        With           LVL          SIG       LVL      Counter Range
                              OFFSET WINDOW
Offset:     1000016                          Offset Range:999876 to 1000058 ns
SIG:        00ED                             Sample Resolution:1
Async LVL: 1x0                               Number of loops Through Range:1
Clk LVL:    1                                Node Signal Scr:U1-1
Count:      594 (580-628)
Clk LVL
SIG
LEGEND:     ▒ means the Clk LVL was 1X,X0 or 1X0       ↑
            * means the SIG or Clk LVL for one offset changed while looping

   F1   F2   F3        F4   F5   F6   F7        F8   F9   F10

  EXECUTING....
```

Screen 5

Press [RETURN]. The following prompt line appears:

```
    TO ______________
```

Again the number entered represents nanoseconds and is biased by the value 1000000 (decimal). The default value of 1001000 appears on the prompt line (1000000 + 1000) to insure the sampling ends at the highest offset for the measurement device.

Press [RETURN]. The following prompt line appears:

```
    SAMPLE RESOLUTION (1-9) ___________
```

The number entered represents the sampling resolution where 1 is the highest resolution (sample taken at each offset tap) and 9 is the lowest resolution (sample taken at every 9th offset tap). The default value of 1 appears on the prompt line.

Press [RETURN]. A prompt to probe a pin or clip a device appears.

Follow the instructions given on the screen. When the button is pressed, the status message "`EXECUTING...`" appears at the bottom of the screen and sampling begins (example shown in screen 5). **DO NOT** remove the measurement device until the entire offset range has been sampled. The status message "`COMPLETE`" will replace "`EXECUTING.....`" when execution is done.

You may now use the [→] and [←] to move through the offset range and examine the response data at each offset tap.

Changing the Calibration Delay

When the calibrated offset delay for an I/O module or the probe is not appropriate for a measurement, the setoffset command may be used to change the delay. For example, if you have a signature recorded with a "-", you need to delay the calibrated offset.

The setoffset command takes an argument for the desired offset value. This offset has a bias of 1000000. So if you want to program an offset for the probe with +30 stops for sync to pod data, do the following:

```
    sync device "/probe", mode "pod"
    sync device "pod", mode "data"
    status = setoffset device "/probe",
    offset 1000000 + 30
```

The offset command returns a 1 or 0. A 1 is returned if the delay could be programmed successfully. A 0 is returned if the delay requested is outside of the range of hardware.

Delays can be varied in 4-nanosecond steps for the probe, and 15 nanosecond steps for the I/O module.

The getoffset command is valuable for accessing the current offset, such as the calibration value. To get the cal value for the probe, the command may look like this:

```
cal_offset = getoffset device "/probe"
```

NOTE: Both the `getoffset` and `setoffset` commands reflect the values for the current sync mode only.

Move the cursor to the `Clocked LVL` column and Press [SELECT].

What happened?

Press [SELECT].

Press [SHIFT] with [SELECT].

What happened?

Only intensified responses are SAVE'ed. Select the signature and clocked level columns.

QUIT and SAVE all response information.

Adding Information to an Existing Response File

The micro_data response file is only partially complete for UFI troubleshooting. In order to be complete, all destination nodes must be entered and learned. This could be a long and monotonous chore.

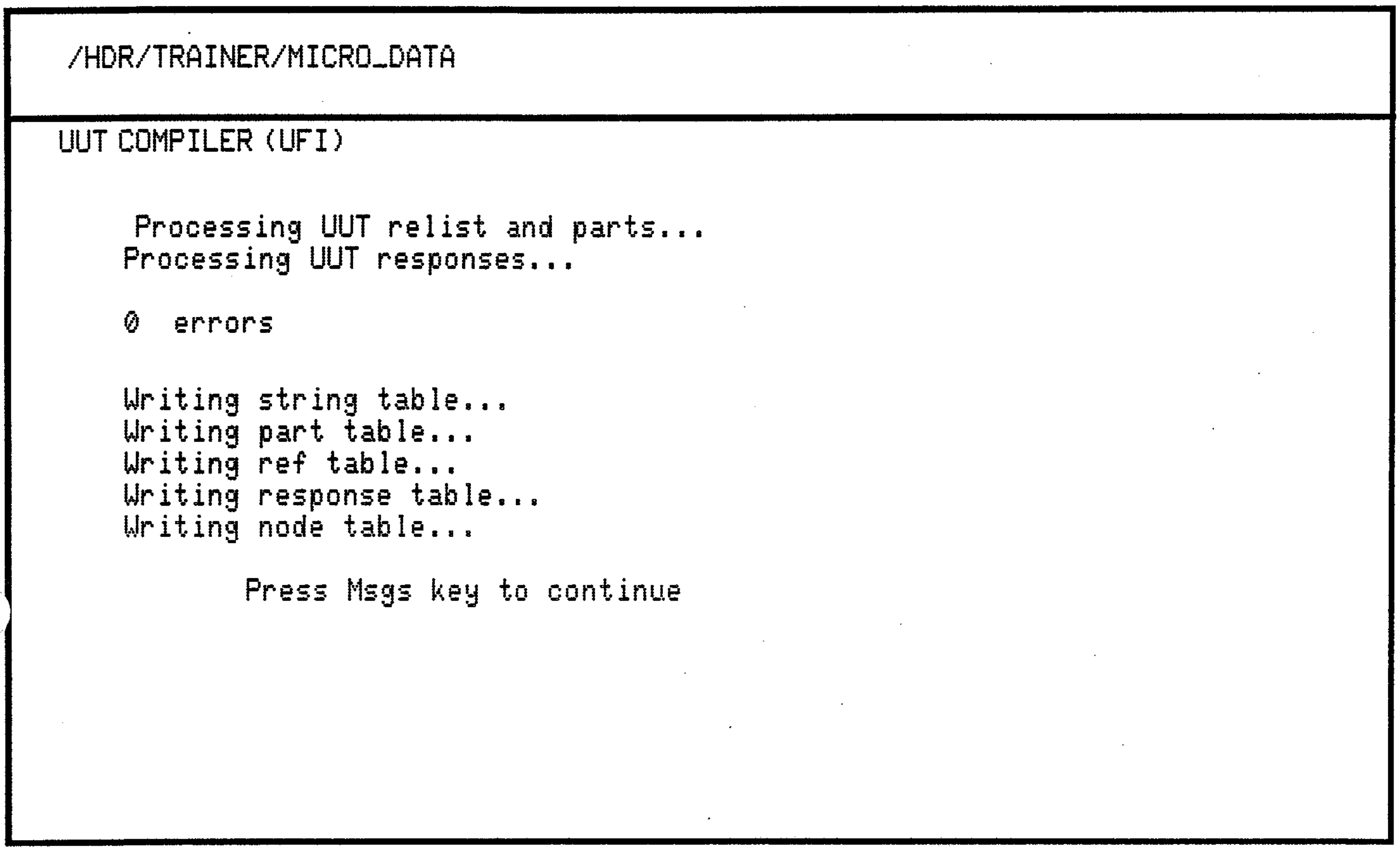
All the information required to enter the remainder of the responses is known. The destination nodes are on the schematic. The response at one end of a wire should be the same at the other end. Here is a quick and easy way to add destination nodes.

1. At UUT directory level, COPY the response file to a text file of the same name.
2. Edit the text file containing response information.

   NOTE: At the top of the file is all of the information displayed in the response file info window. DO NOT modify any of the info window text.
3. You can now use the editor's yank-and-paste feature to copy information from one or more lines. For example, the input responses of the buffers are identical to the output responses. All the lines containing U3 responses could be yanked and appended to the end of the file. The pin numbers would then need modification.
4. Add all of the data inputs to the buffers.

   NOTE: Maintain the same format. The columns should be the same as original response file.
5. Copy the text file back to the response file.

Keep these text files until you know that all of the responses for that stimulus are complete. It is often easier to add responses to the text file than the response file.



```
 /HDR/TRAINER/MICRO_DATA

UUT COMPILER (UFI)

     Processing UUT relist and parts...
    Processing UUT responses...

    0  errors

    Writing string table...
    Writing part table...
    Writing ref table...
    Writing response table...
    Writing node table...

           Press Msgs key to continue
```

Screen 6

Step 7: Compile To Troubleshoot

To use the response information for GFI/UFI, it must be compiled into the Database. To add the response information to the Database you need to perform a Compile To Troubleshoot operation.

At the UUT directory level,

Press: [F3] (COMPILE).

Select UFI then press [Return].

Select: TROUBLESHOOT UUT then press [Return].

Display should resemble Screen 6.

Step 8: Summary File

The Summary File can be used to tell the completeness of your database. It is very important to note that the summary file does not indicate how well the nodes are tested, just that they are tested.

To create a summary file, you must be at the UUT directory level.

Press: [F8] (SUMMARY).

The following prompt will appear at the bottom of the screen:

```
Generate GFI Summary to TEXT file ____
```

The summary of the UFI Database will be generated to this text file.

Type in the text file name you wish and press the return key. The summary will be generated to the text file and will appear on the 9100A display.

You may now review the summary file by editing the text file. If you ever wish not to generate a text file, you may press [SHIFT] with [SUMMARY]. Only the summary will be displayed.

Step 9: Testing

Testing should be done in two modes. First, test the nodes with no faults. Then test the node with faults.

UFI is invoked in the immediate mode by pressing the GFI key. When UFI probes a pin, the status of that pin is returned with no suggestions on where to test next. Instead, the message UNGUIDED MODE appears on the display.

Leave the editor by pressing [SHIFT] along with [QUIT].

Press: [GFI].

The following prompt will appear:

```
    RUN GFI UUT CD REF   PIN
```

In the UUT field, enter TRAINER using [HELP]. In the REF field enter U3 and in the PIN field enter 16.

Press: [ENTER].

The following prompt will be displayed:

```
    PROBE U3 PIN 16
    PRESS BUTTON ON PROBE WHEN READY
```

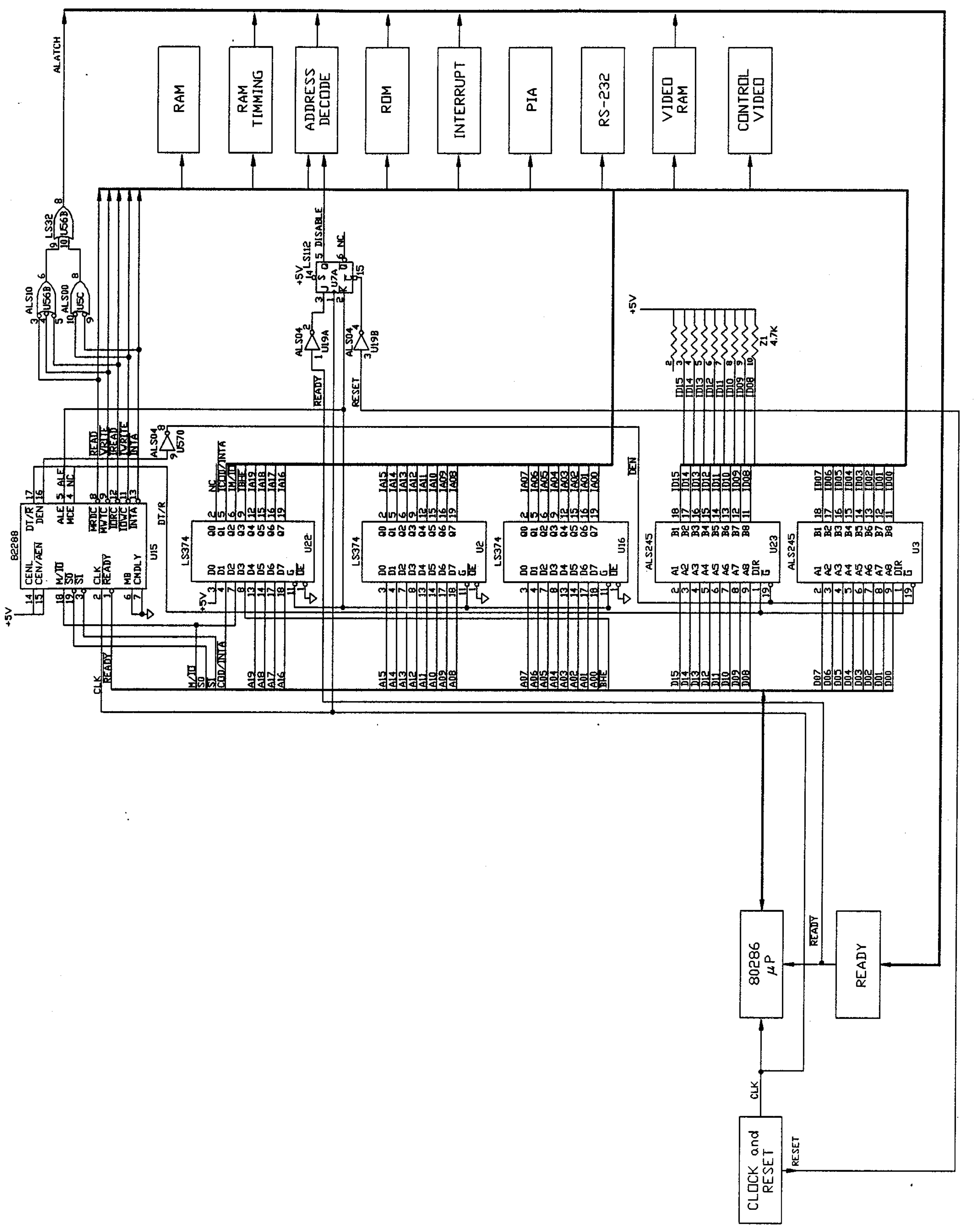
Probe U3 pin 16 and press the probe button.

You should see the following response:

```
    UNGUIDED MODE
    PIN GOOD AS OUTPUT
```

Set fault switch SW4-2 to the fault position, and test U3 pin 16 again.

What happened?

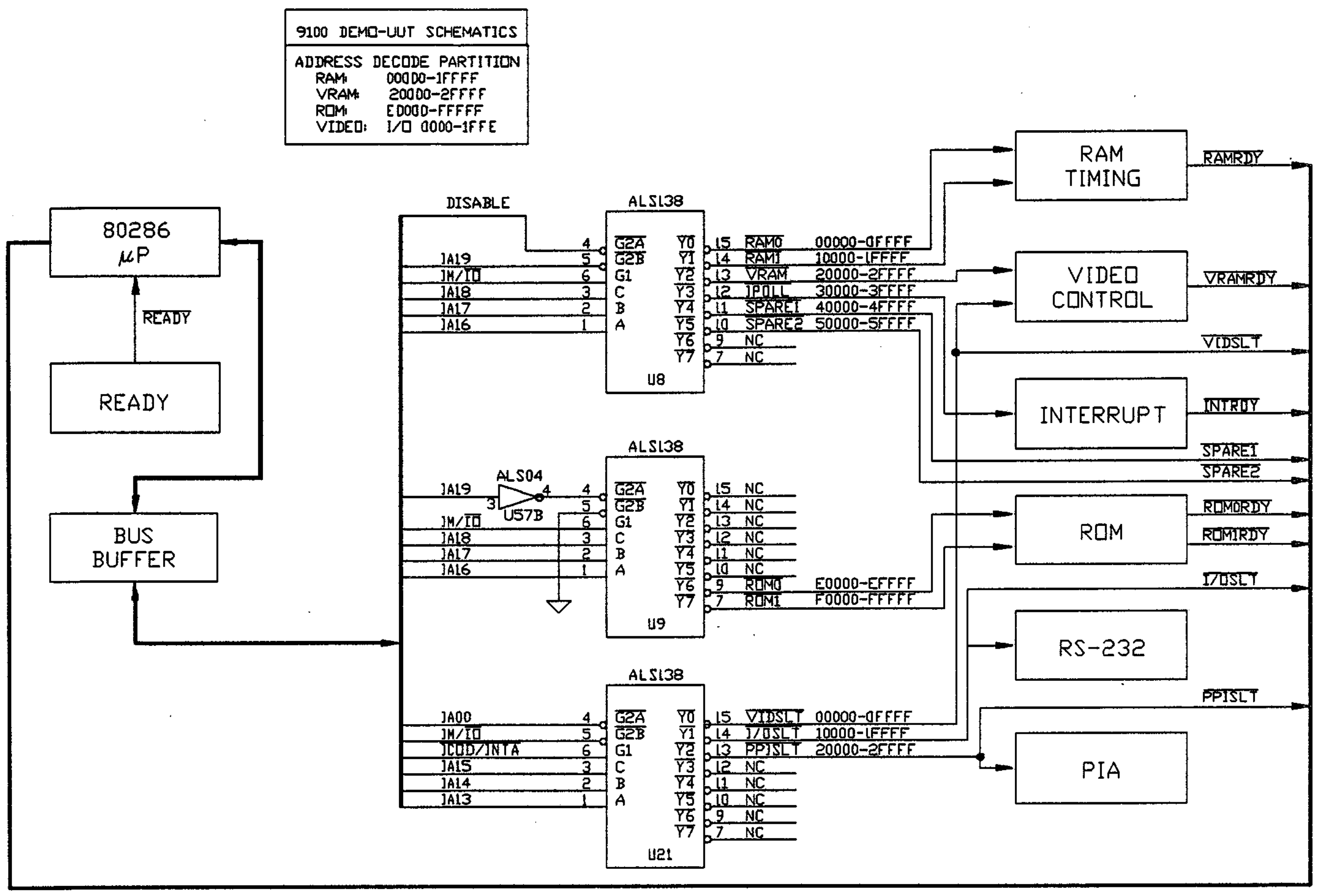

ALATCH
RAM
RAM TIMING
ADDRESS DECODE
ROM
INTERRUPT
PIA
RS-232
VIDEO RAM
CONTROL VIDEO
LS32 U56B
ALS10 U56B
ALS00 U5C
+5V
LS112 U7A
DISABLE
NC
ALS04 U19A
ALS04 U19B
READY
RESET
Z1 4.7K
82288
U15
ALS04 U57D
LS374 U22
LS374 U2
LS374 U16
ALS245 U23
ALS245 U3
DT/R
DEN
CLK
M/IO
S0
S1
COD/INTA
80286 µP
READY
CLOCK and RESET

Lab Exercise

Address Bus

Perform all the steps necessary to test the address bus from the microprocessor. Include U14, U22, U2, and U16 address lines.

When complete, perform a GFI test on U2-2 in the immediate mode and confirm that the pin is good. Then, set switch 1-1 open and test U2-2 again.

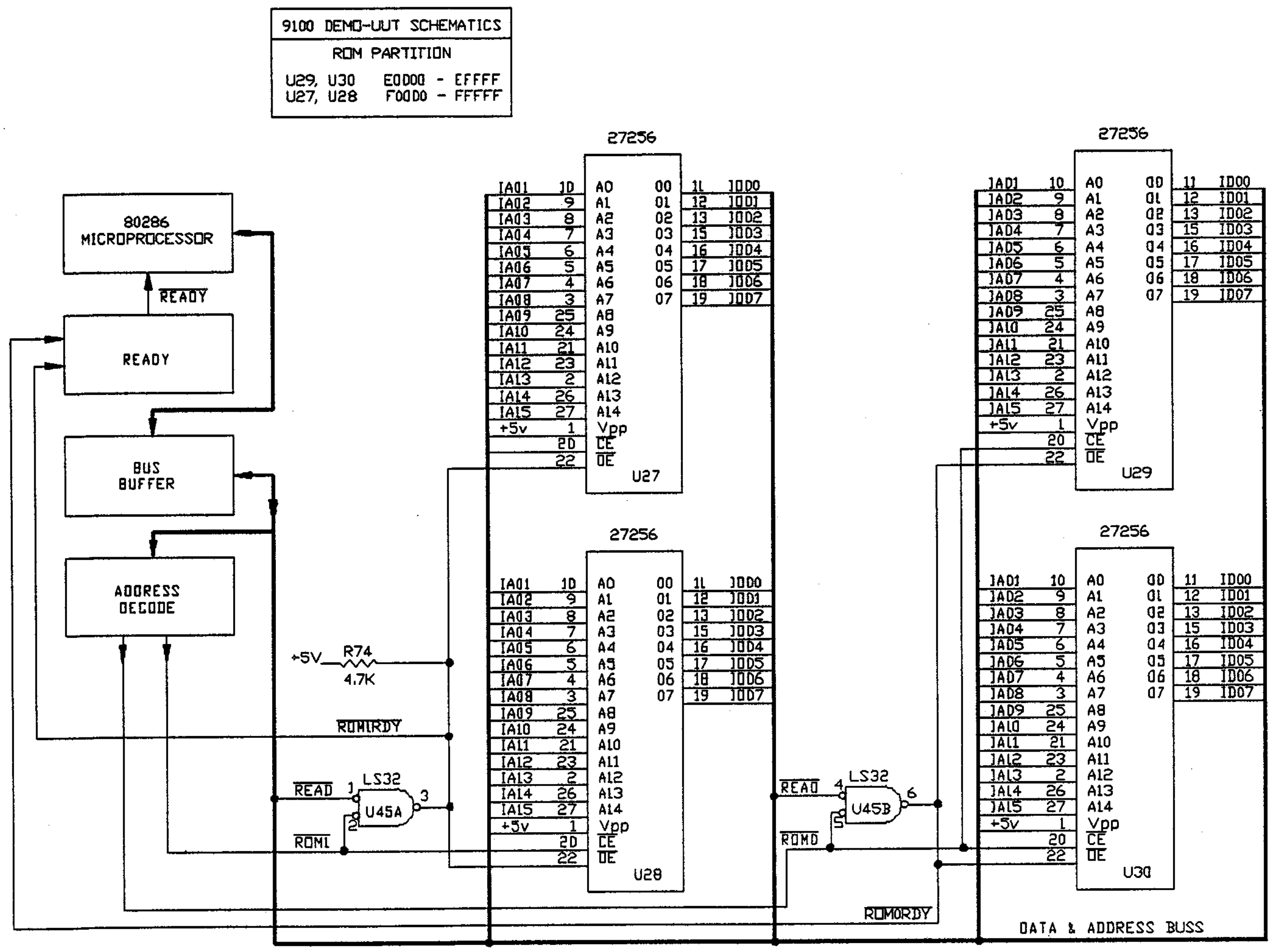

9100 DEMO-UUT SCHEMATICS
ADDRESS DECODE PARTITION
RAM: 00000-1FFFF
VRAM: 20000-2FFFF
ROM: E0000-FFFFF
VIDEO: I/O 0000-1FFE
80286 µP
READY
READY
BUS BUFFER
DISABLE
ALS138
U8
JA19
JM/IO
JA18
JA17
JA16
G2A G2B G1 C B A
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
RAM0 00000-0FFFF
RAM1 10000-1FFFF
VRAM 20000-2FFFF
IPOLL 30000-3FFFF
SPARE1 40000-4FFFF
SPARE2 50000-5FFFF
NC
ALS138
U9
ALS04
U57B
ROM0 E0000-EFFFF
ROM1 F0000-FFFFF
ALS138
U21
JA00
JM/IO
JCOD/INTA
JA15
JA14
JA13
VIDSLT 00000-0FFFF
I/OSLT 10000-1FFFF
PPISLT 20000-2FFFF
RAM TIMING
RAMRDY
VIDEO CONTROL
VRAMRDY
VIDSLT
INTERRUPT
INTRDY
SPARE1
SPARE2
ROM
ROM0RDY
ROM1RDY
I/OSLT
RS-232
PPISLT
PIA

ROM Address Decode Circuit

Perform all the steps necessary to test:
ROM address decoder outputs U9,
ROM select pins U45-3 and U45-6,
ROMS U27, U28, U29 ,
U30 decode inputs.

Add the decode inputs to the Address Stimulus Response file.

When complete, perform a GFI test on U27-22 in immediate mode and confirm that the pin is good. Then, set Fault switch 1-2 and test U27-22 again.

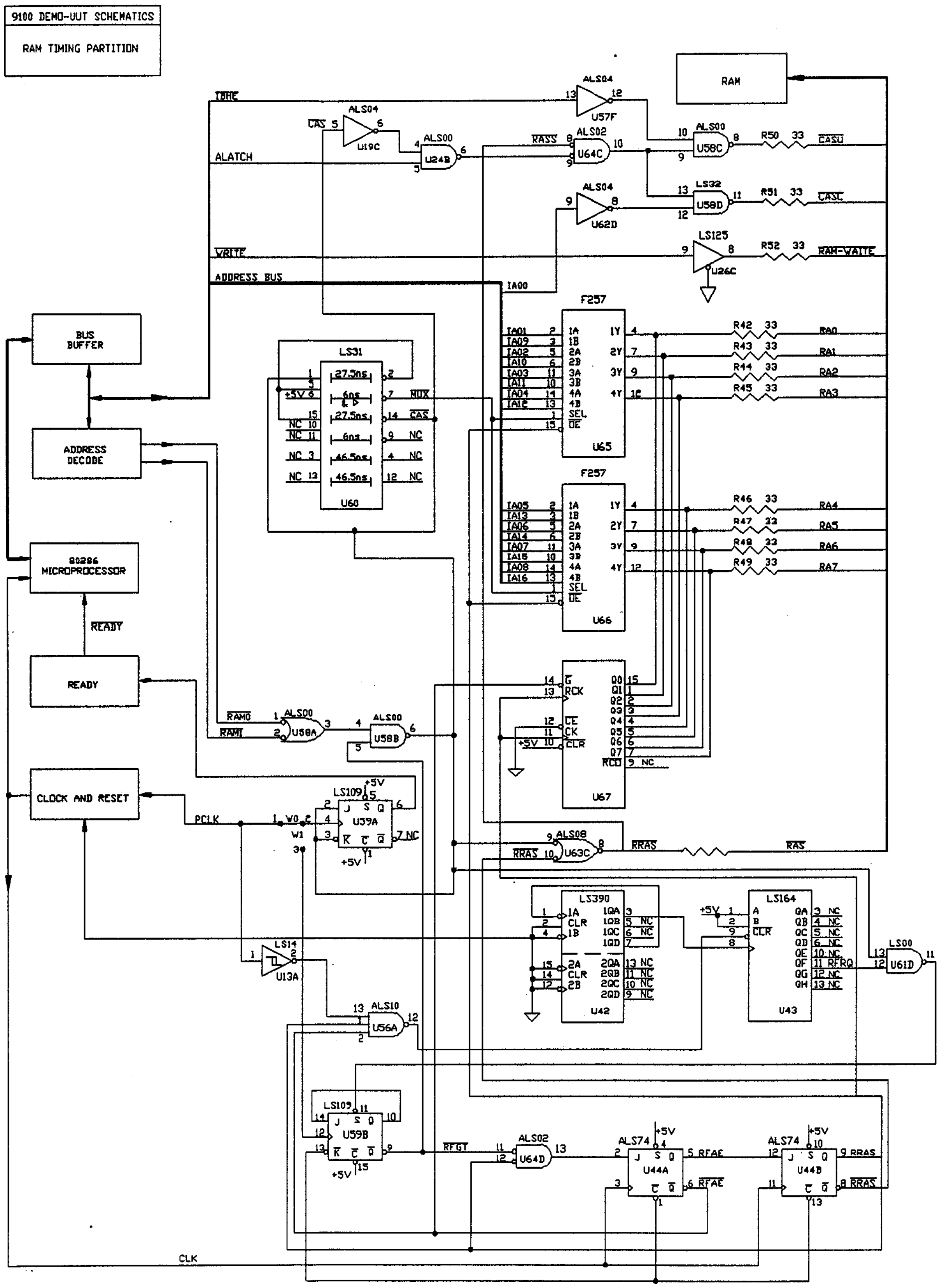

9100 DEMO-UUT SCHEMATICS
ROM PARTITION
U29, U30 E0000 - EFFFF
U27, U28 F0000 - FFFFF
80286 MICROPROCESSOR
READY
BUS BUFFER
ADDRESS DECODE
27256
U27
U28
U29
U30
R74 4.7K
+5V
ROM1RDY
ROM0RDY
READ
ROM1
ROM0
LS32
U45A
U45B
DATA & ADDRESS BUSS

ROM 1 Data Bus

Perform all the steps necessary to test the data bus from ROM 1.

Add the ROM 1 address and data inputs to the appropriate response file.

When complete, perform a GFI test on U27-17 in immediate mode and confirm that the pin is good. Then, set Fault switch 4-2 and test U27-17 again.


9100 DEMO-UUT SCHEMATICS
RAM TIMING PARTITION
RAM
BUS BUFFER
ADDRESS DECODE
80286 MICROPROCESSOR
READY
CLOCK AND RESET
ALATCH
WRITE
ADDRESS BUS
F257
U65
U66
U67
LS31
U60
LS390
U42
LS164
U43
RASS
CASU
CASL
RAM-WRITE
RRAS
RAS
PCLK
CLK
RFGT
RFAE
RFRQ

RAM Row Addressing

Perform all the steps necessary to test the RAM address bus (RA0 - RA7) from the RAM address multiplexers (U65, U66) during Row address time only, to inputs of U34.

Add the multiplexor address inputs to the appropriate response file.

When complete, perform a GFI test on U34-11 in immediate mode and confirm that the pin is good. Then, set Fault switch 4-8 and test U34-11 again.

Section 3 GFI: Microprocessor Emulation

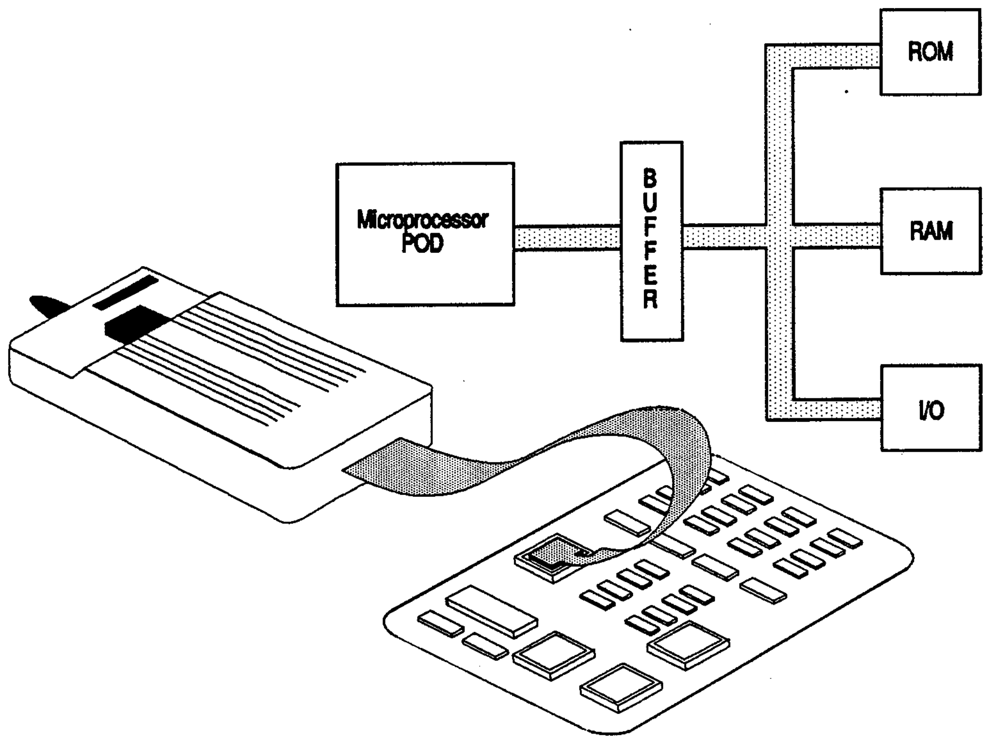
Creating a GFI Database

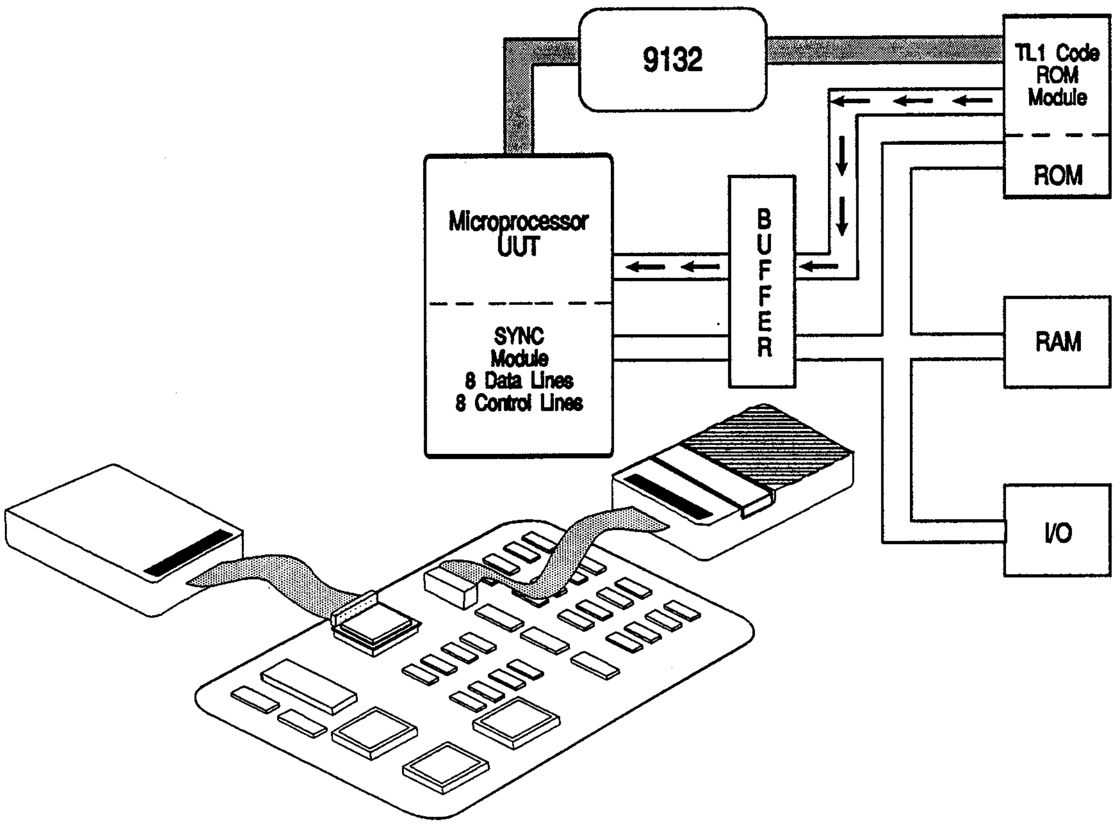
1. Enter NODELIST
2. Enter Part Descriptions
3. Enter Reference Designator List
4. Compile to Learn Responses
5. Develop Stimulus Routine
6. Write Stimulus Program
7. Learn Stimulus Responses
8. Compile to Troubleshoot UUT
9. Perform a Summary
10. Test

Creating a GFI Database

1. Verify NODELIST using TRAINERB (UUT)
2. Verity part description of a 74LS374.
3. Verify Reference Designator of U2 (I/O Mod)
4. Compile to Learn Responses (GFI).
5. Develop stimulus routine (discuss only).
6. Write stimulus program (use ADDR_OUT).
7. Learn stimulus response (A15 and IA15).
8. Compile to troubleshoot UUT (GFI).
9. Perform a Summary (look at U2).
10. Test
    a. Edit: /hdr/trainerb/go_nogo and observe how the program is written.
    b. Edit: /hdr/trainerb/test_ram and observe how the program is written.
    c. In immediate mode execute UUT TRAINERB program go_nogo.
    d. Repeat step 10c. with fault switch 1-1.

Section 4
Memory Emulation


ROM
Microprocessor POD
BUFFER
RAM
I/O

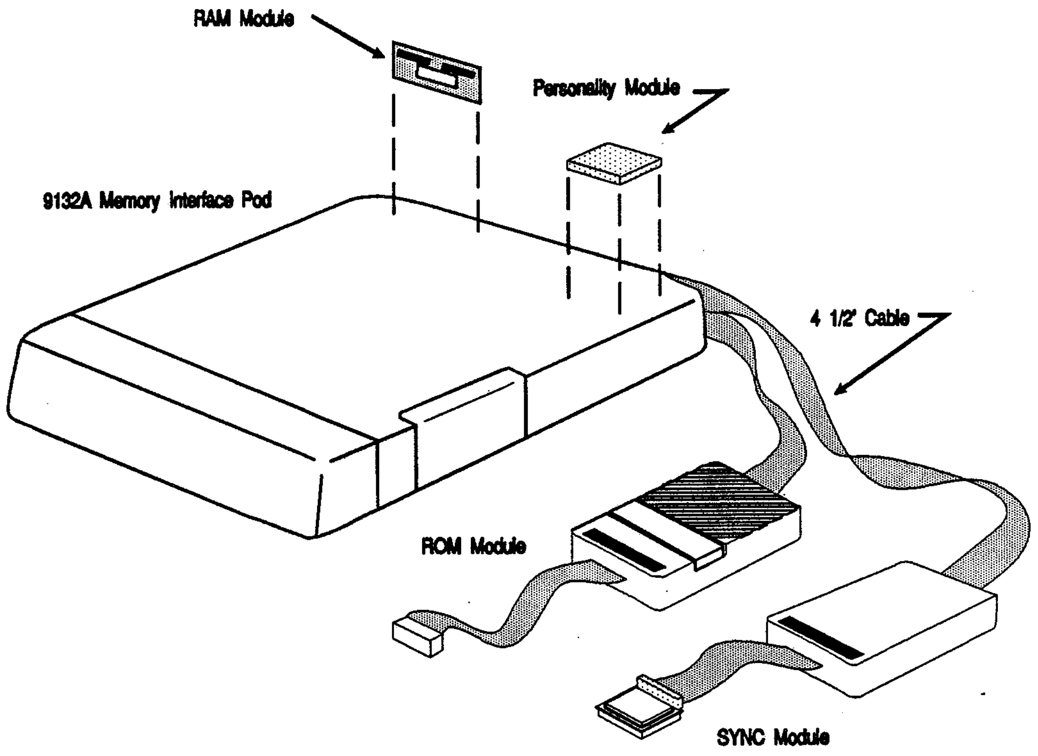

9132
TL1 Code ROM Module
ROM
Microprocessor UUT
SYNC Module
8 Data Lines
8 Control Lines
BUFFER
RAM
I/O

Overview

The 9132A Memory Interface Pod allows you to use any Fluke 9100-Series Digital Test System/Station to troubleshoot equipment that uses a variety of microprocessors (µP) and ROMs.

The Memory Emulation Pod allows the 9100A to gain access to the UUT by replacing the Boot Memory with ROM Modules and connecting 17 lines of logic analysis onto the µP. 16 of the lines are used for logic analysis while 1 is used to control reset of the UUT. We gain access to the UUT through the boot memory by resetting the UUT which forces the µP to the reset address which is typically the boot address. Once this has been done, all of the µP overhead is handled by the 9132A passing code to the µP through the ROM modules inserted into the boot memory.

In the case of a kernel fault which would not allow the µP to run due to bad address, data, or control lines, the 9132 would reset the UUT and feed the µP patterns that will isolate faults through a diagnostic algorithm within the 9100A.


RAM Module
Personality Module
9132A Memory Interface Pod
4 1/2' Cable
ROM Module
SYNC Module

Description of the Pod

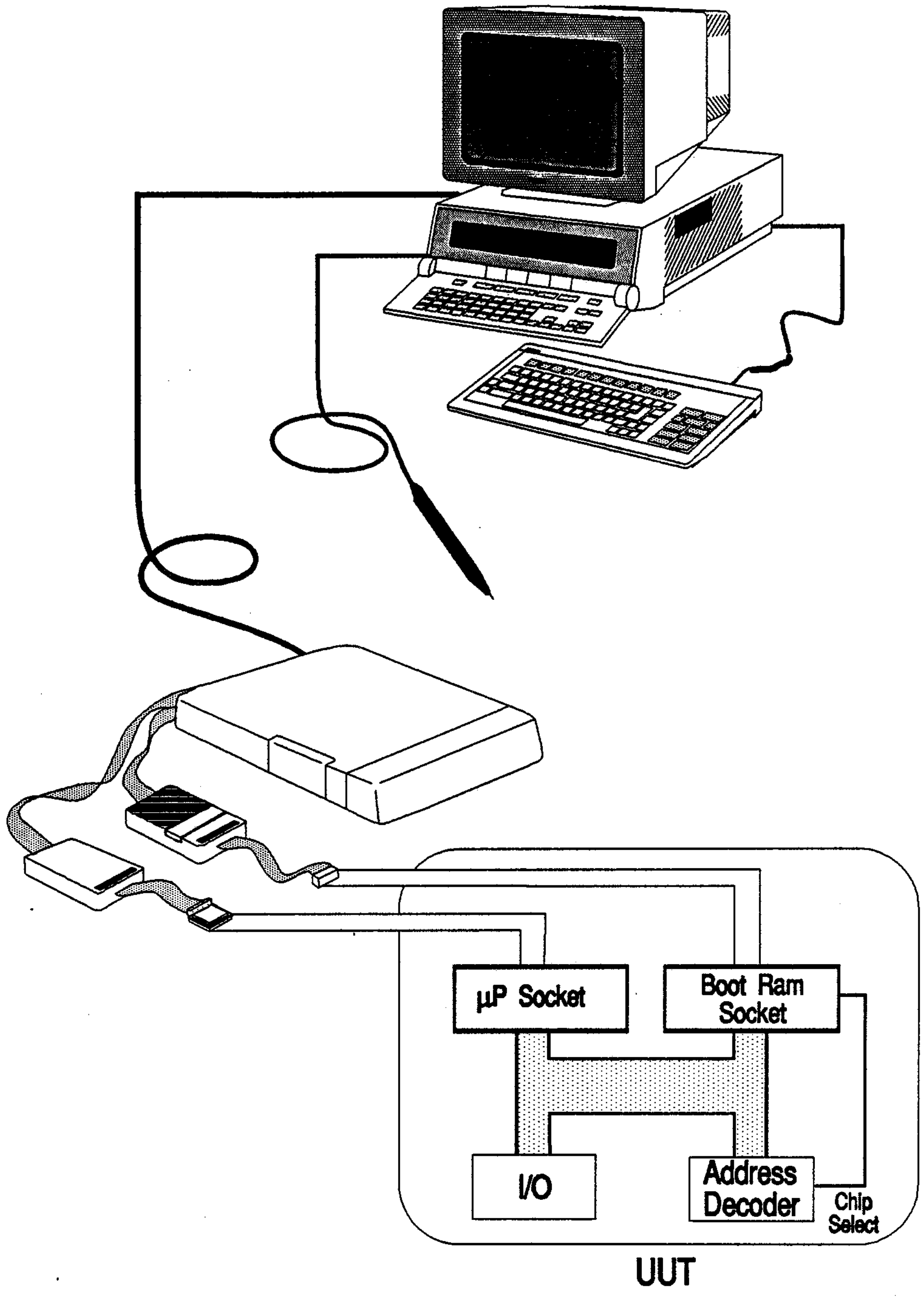
The Pod connects to the Mainframe through a round shielded cable, and connects to the UUT through a set of ROM modules that are inserted into the UUT's boot ROM sockets. The UUT's ROMs are removed from the UUT and are replaced by the Pod's ROM modules. The UUT's boot ROMs are placed in sockets located on the ROM modules. In addition, a Sync Module is connected to various lines on the UUT to control timing and reset for the UUT processor.

The Pod consists of a base unit and several attached parts. One Sync Module and up to four ROM Modules may be attached to the Pod. The Pod contains the software and supporting hardware that is required to do the following:

Receive and execute commands from the Mainframe

Report UUT fault conditions to the Mainframe

Exercise the UUT


µP Socket
Boot Ram Socket
I/O
Address Decoder
Chip Select
UUT

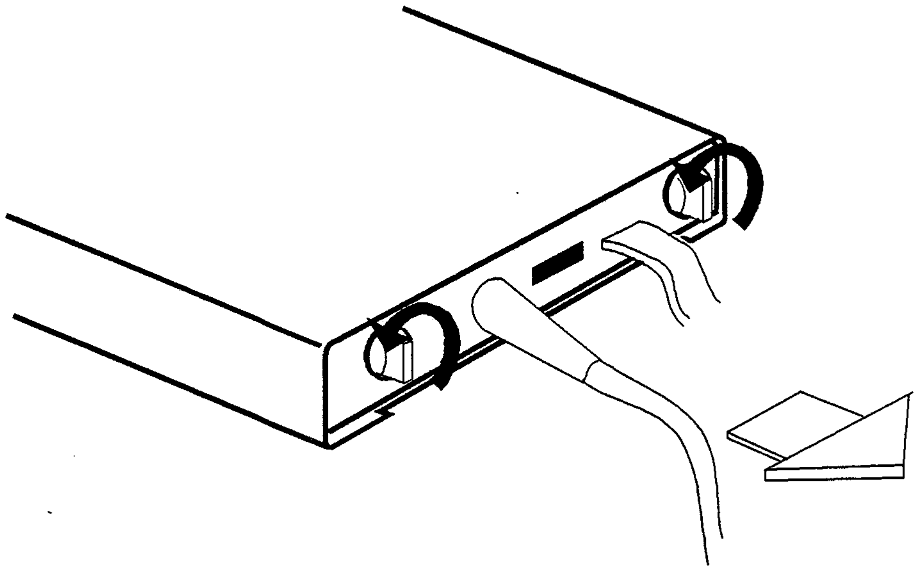
Description of the Pod

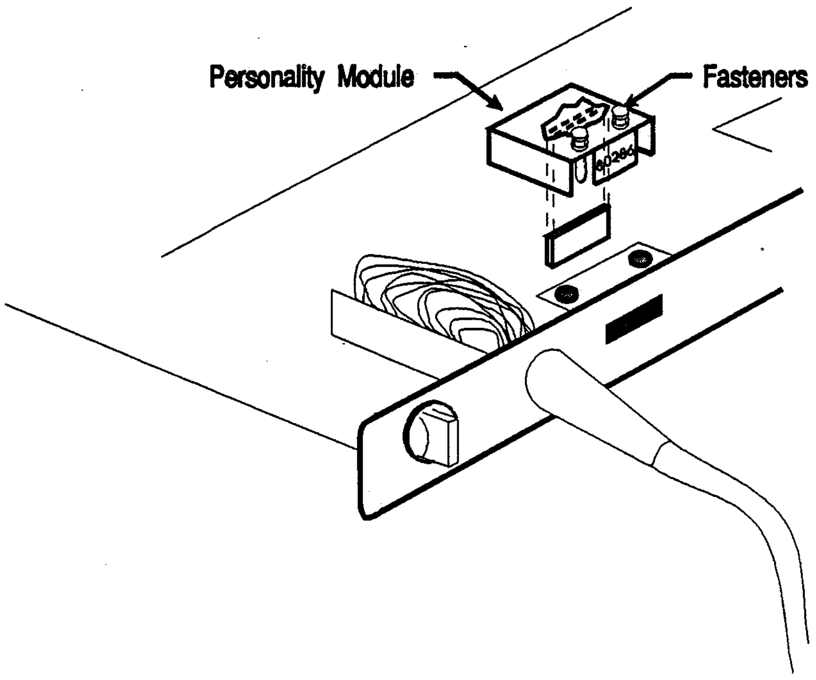
The Mainframe supplies operating power for the Pod. The UUT provides external timing signals required by the Pod, which allows the Pod to synchronize its internal functions to those of the UUT.

Overvoltage protection circuits or fuses on each line to the UUT guard against damage to the Pod. The overvoltage protection circuits guard against voltages of +12V to -7V on any pin of the ROM Module plug. Multiple faults, especially of long duration, may cause Pod damage.

A power-level sensing circuit monitors the voltage level of the UUT power supply. If UUT power drops below an acceptable level, the Pod notifies the Mainframe of a bad power supply condition.

A self-test socket on the Pod enables the mainframe to check Pod operation. The ROM Module cable plugs and the Sync Module adapter cable are connected to the self-test socket during self test operation, which allows the mainframe to investigate the Pod's internal functions.




Personality Module
Fasteners

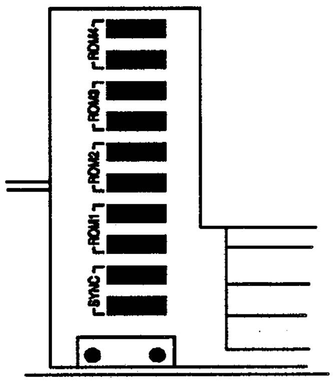
Installation of the Pod and Modules

1. Check that the Mainframe power is OFF.

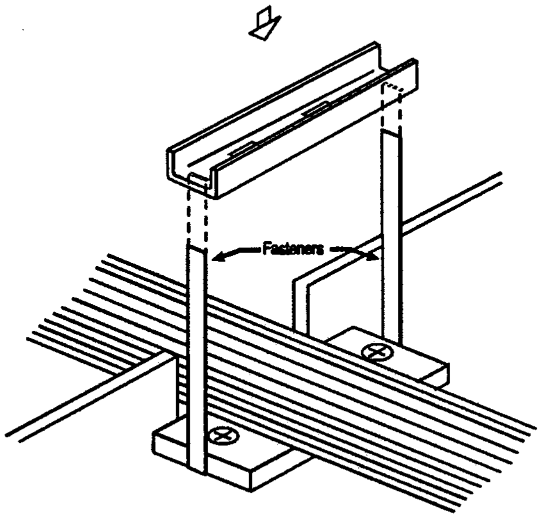
2. Open the back panel of the Pod by turning the thumbscrews on each side and then pull the panel out from the case.

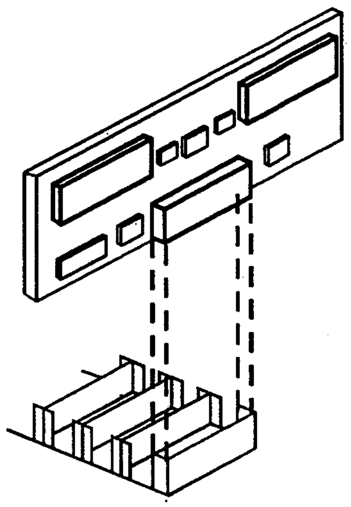
3. To configure the Pod for the 80286 processor, an 80286 Personality Module must be installed in the Pod.

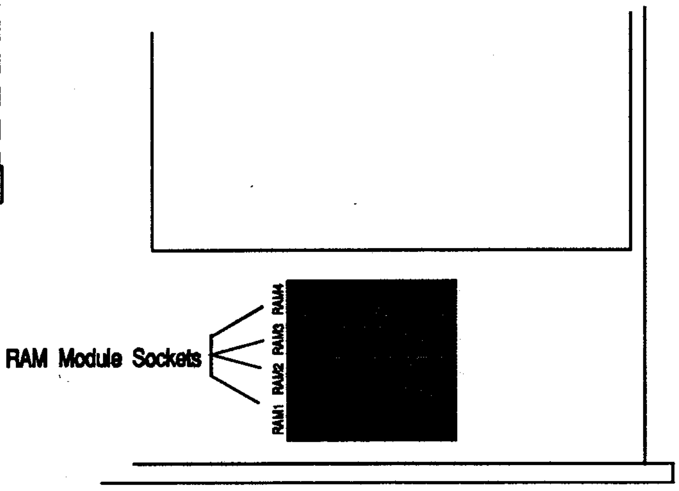
4. Verify that the correct Personality Module is installed.



Rear Panel







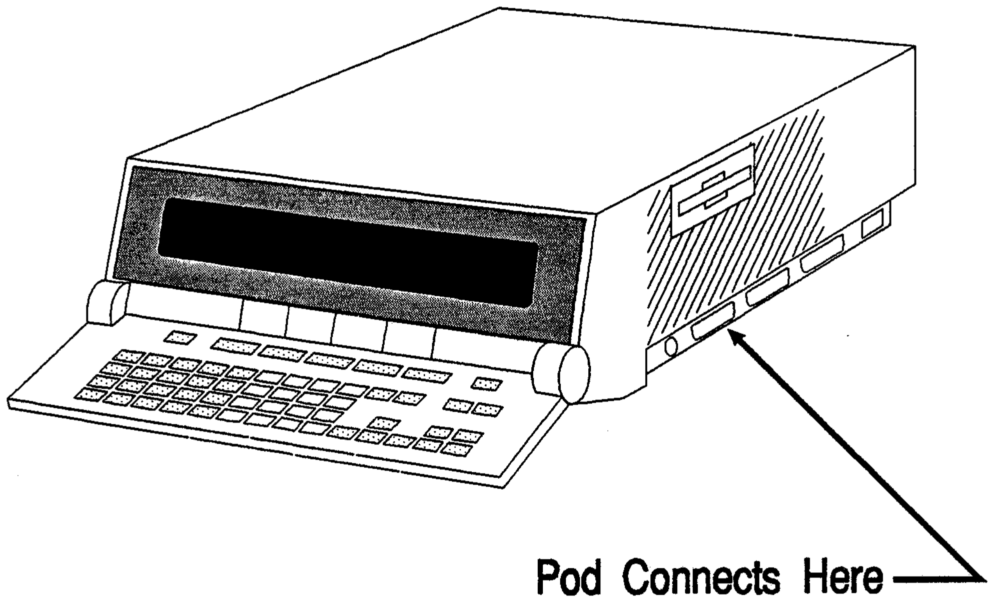
Rear Panel

Installation of the Pod and Modules

5. Install the Sync and ROM Modules next. Plug each module into their corresponding sockets.

6. Verify that the RAM Modules are installed.

7. Close back of pod panel.


Pod Connects Here

Installation of the Pod and Modules

8. Connect the Pod to the Mainframe.

9. Turn power ON to the Mainframe.

MAIN: COPY DISK FROM DR1 TO HDR

Master User Disk

The 80286 database for the 9100-Series Mainframe is contained on one 3.5 inch floppy disk supplied with the 9132A-80286.

To install the database on a Mainframe with a hard disk drive, insert the disk into the Mainframe floppy drive, press:

MAIN MENU K

SOFT KEYS

then select

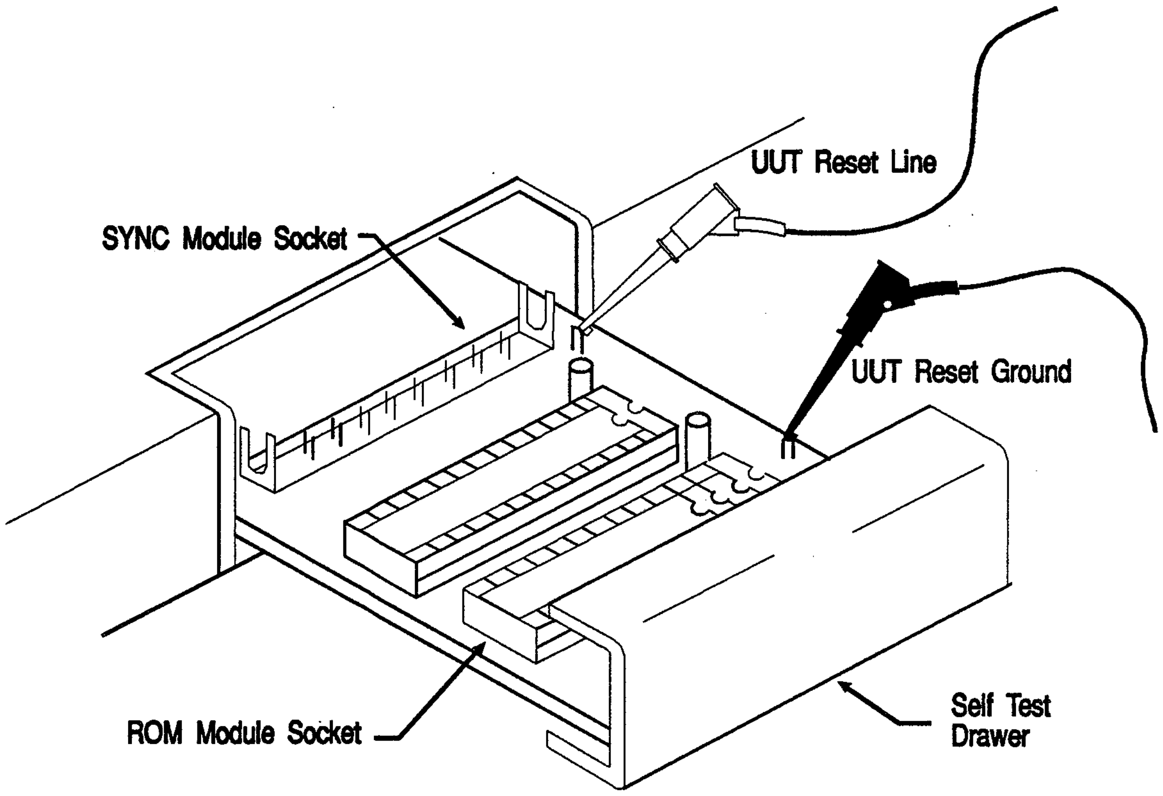
```
COPY DISK FROM DR1 TO HDR
```

ENTER YES

(The database only needs to be installed once.) Once the database is installed on the Mainframe, place the original floppy disk in a safe place in case it is needed in the future.

Copy Master User Disk/80286 from DR1 to HDR.

Press RESET on the 9100 keypad.


UUT Reset Line
SYNC Module Socket
UUT Reset Ground
ROM Module Socket
Self Test Drawer

Pod Self Test

1. Make sure the Pod is connected properly to the Mainframe.

2. Open the self test socket drawer on the right-hand side of the Pod.

3. Insert the Sync Module Adapter Board cable into the socket on the self test pca.

4. Connect the UUT Reset line flying lead to the reset line test connector and the UUT Reset ground line flying lead to the ground line test connector on the self test pca.

5. Insert the ROM Module 1 cable plugs into the ZIF self test sockets. (ROM Module sockets must be empty.)

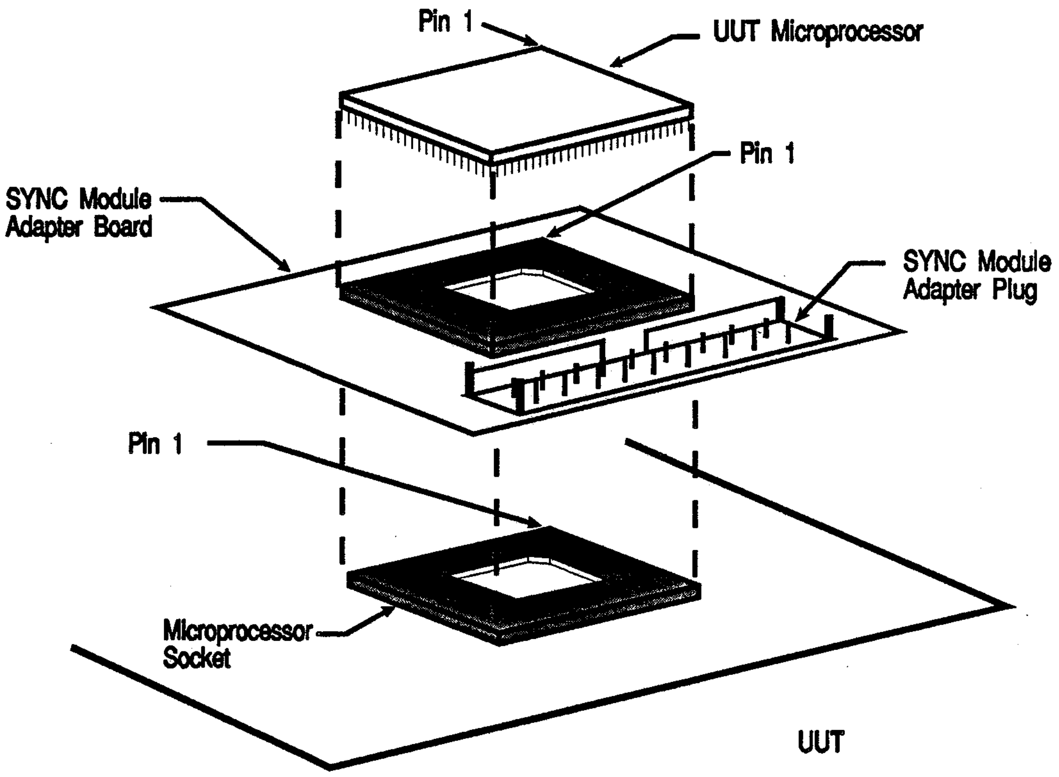
6. Press the MAIN MENU key on the Mainframe to obtain the display:

   ```
   MAIN: SELFTEST POD
   ```

   Press ENTER YES

7. Repeat steps 5 and 6 for ROM Module 2.

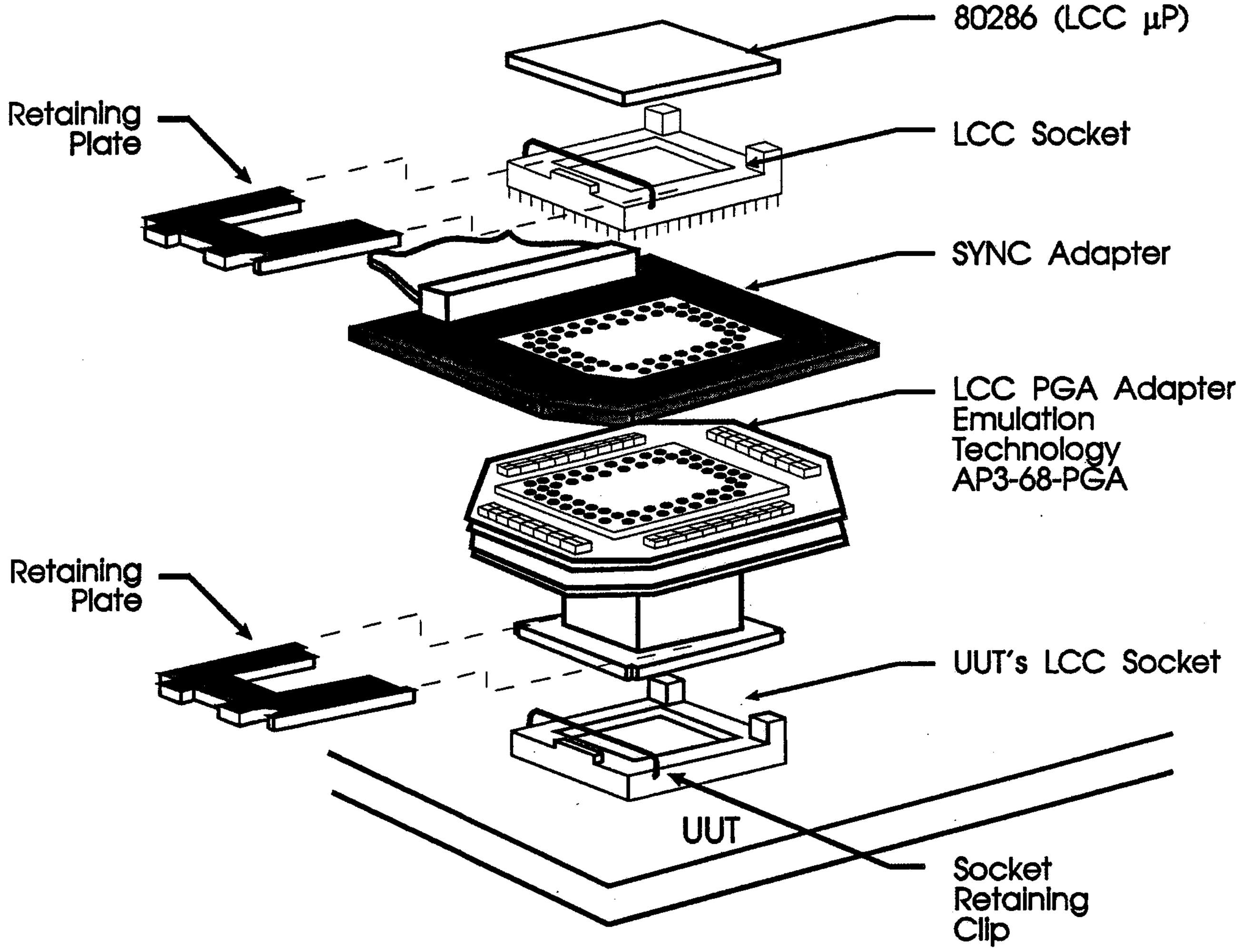
If the self test fails refer to Appendix F in the Pod manual.

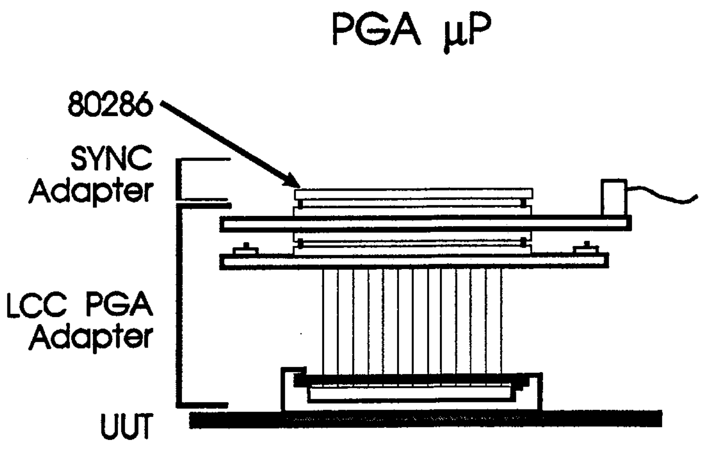

Pin 1
UUT Microprocessor
Pin 1
SYNC Module Adapter Board
SYNC Module Adapter Plug
Pin 1
Microprocessor Socket
UUT

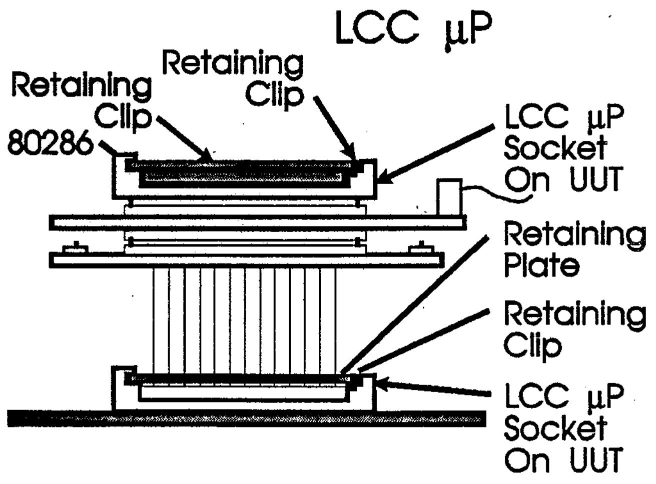
Connecting the Pod to the UUT

Sync Module Connection to the UUT

1. PGA (Pin Grid Array)up to PGA socket adapter.

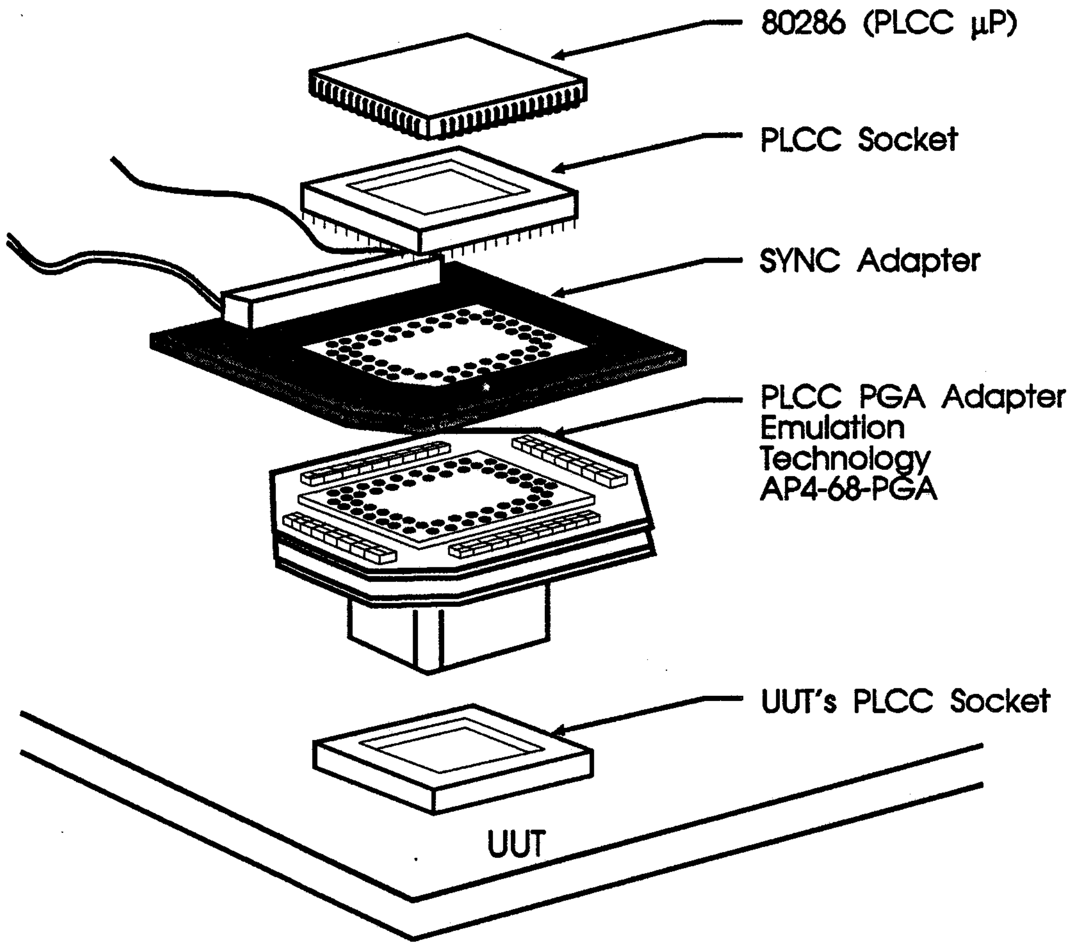

80286 (LCC µP)
Retaining Plate
LCC Socket
SYNC Adapter
LCC PGA Adapter Emulation Technology AP3-68-PGA
Retaining Plate
UUT's LCC Socket
UUT
Socket Retaining Clip

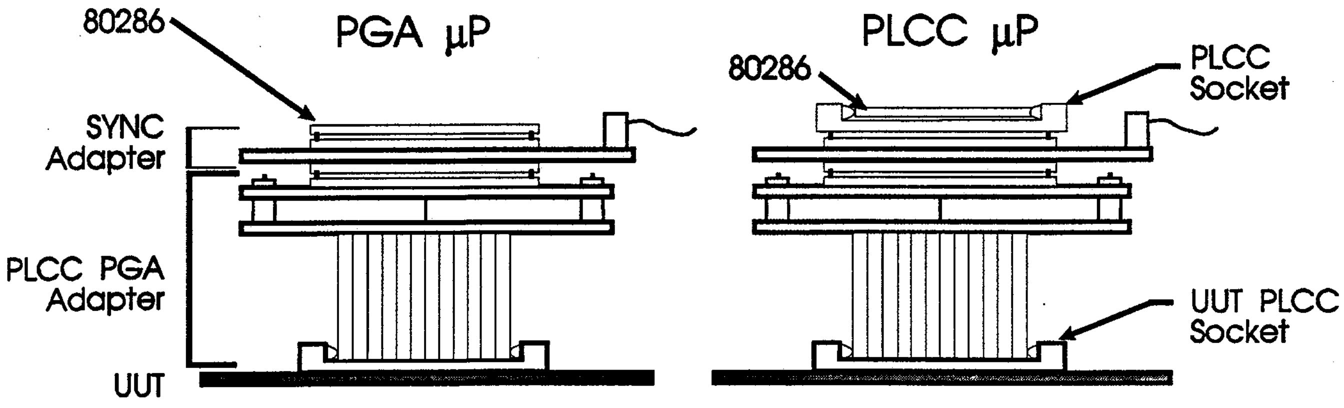

PGA µP
80286
SYNC Adapter
LCC PGA Adapter
UUT


LCC µP
Retaining Clip
Retaining Clip
80286
LCC µP Socket On UUT
Retaining Plate
Retaining Clip
LCC µP Socket On UUT

Sync Module Connection to the UUT

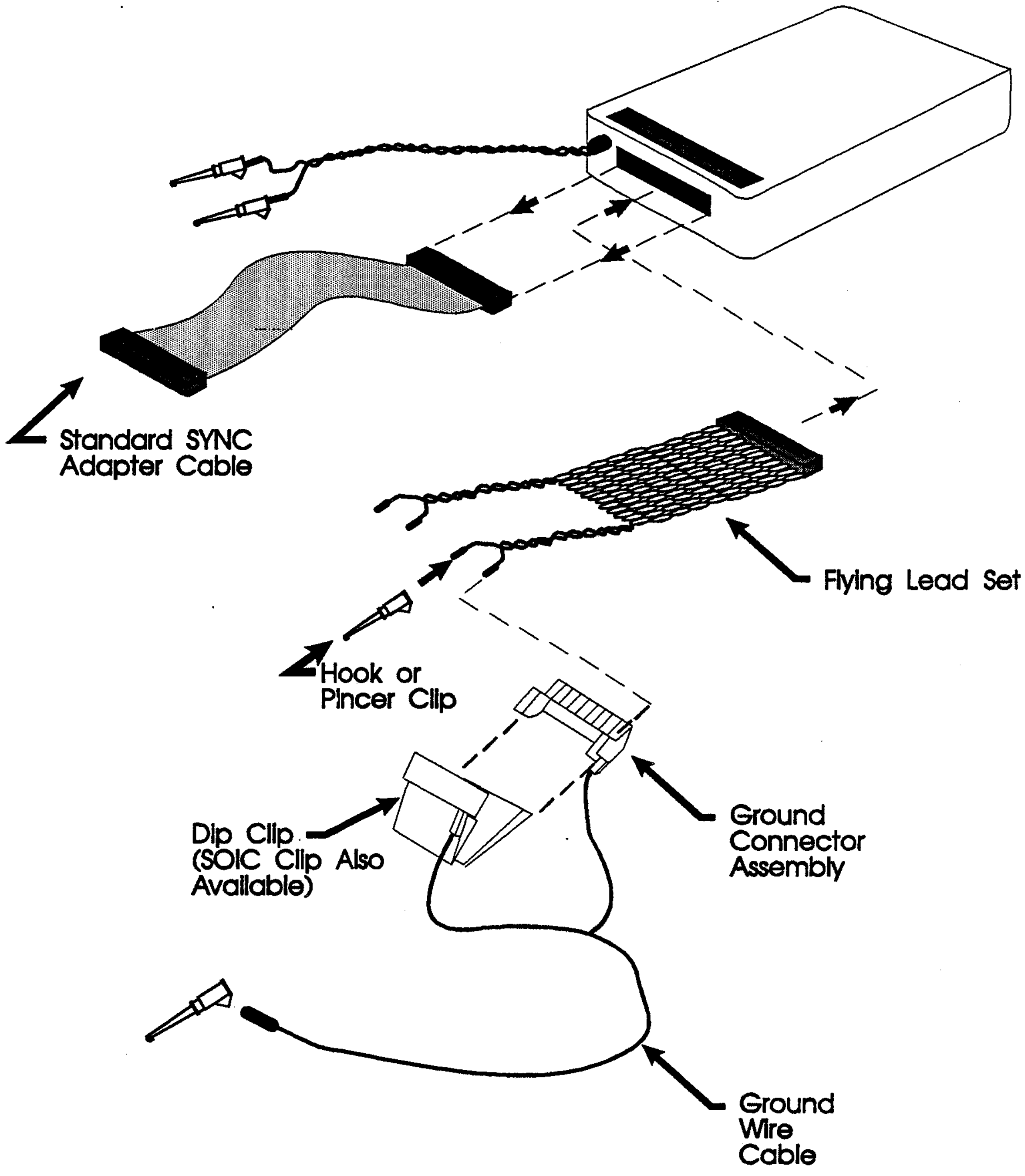
2. LCC up to LCC (Leadless Chip Carrier) socket adapter.


80286 (PLCC µP)
PLCC Socket
SYNC Adapter
PLCC PGA Adapter
Emulation
Technology
AP4-68-PGA
UUT's PLCC Socket
UUT


PGA µP
80286
SYNC
Adapter
PLCC PGA
Adapter
UUT
PLCC µP
80286
PLCC
Socket
UUT PLCC
Socket

Sync Module Connection to the UUT

3. PLCC (Plastic Leaded Chip Carrier) up to PLCC socket adapter.


Standard SYNC Adapter Cable
Flying Lead Set
Hook or Pincer Clip
Dip Clip (SOIC Clip Also Available)
Ground Connector Assembly
Ground Wire Cable

Sync Module Connection to the UUT

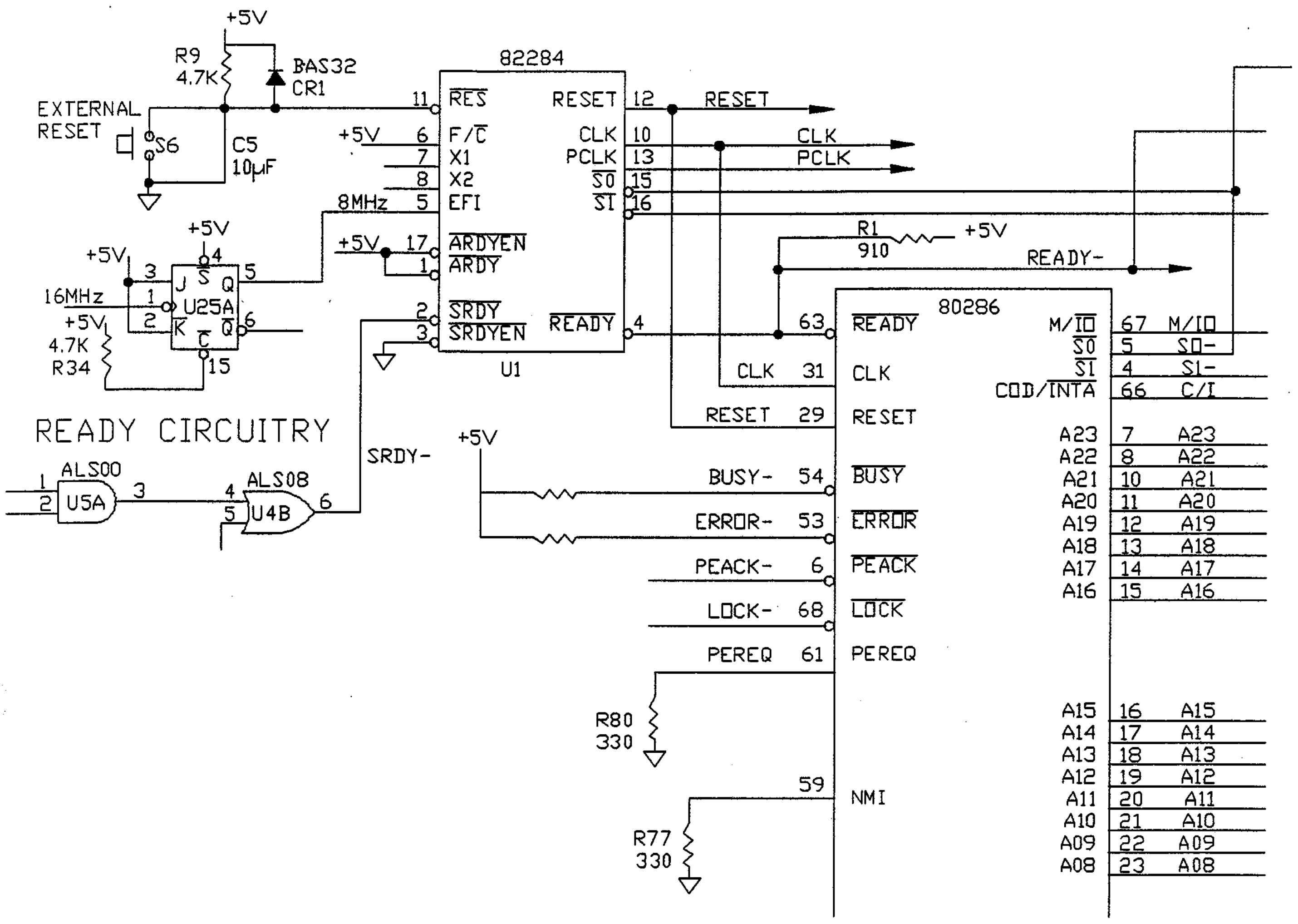
4. Flying lead set.

PIN NUMBER	SIGNAL	PIN NUMBER	SIGNAL
1	GND	2	D0
3	GND	4	D1
5	GND	6	D2
7	GND	8	D3
9	GND	10	D4
11	GND	12	D5
13	GND	14	D6
15	GND	16	D7
17	GND	18	CLK
19	GND	20	RESET
21	GND	22	$\overline{\text{READY}}$
23	GND	24	HOLD
25	GND	26	HLDA
27	GND	28	$\overline{\text{S0}}$
29	GND	30	$\overline{\text{S1}}$
31	GND	32	PEREQ
33	GND	34	(KEY)

Sync Module Connection to the UUT

5. Connect the 68 pin PGA socket saver to the ZIF socket on the UUT.

6. Connect the PGA adapter board to the socket saver that is plugged into the ZIF socket. Connect the Sync Module to the PGA adapter board. (Be sure that power is removed from the UUT)


+5V
R9
4.7K
BAS32
CR1
82284
EXTERNAL
RESET
S6
C5
10μF
11 RES
RESET 12
RESET
+5V 6 F/C
CLK 10
CLK
7 X1
PCLK 13
PCLK
8 X2
S0 15
8MHz 5 EFI
S1 16
+5V
+5V 17 ARDYEN
1 ARDY
R1 910 +5V
READY-
+5V
3 J S Q 5
16MHz 1 U25A
+5V 2 K C Q 6
4.7K R34
15
2 SRDY
3 SRDYEN
READY 4
U1
80286
63 READY
M/IO 67 M/IO
S0 5 S0-
S1 4 S1-
COD/INTA 66 C/I
CLK 31 CLK
RESET 29 RESET
READY CIRCUITRY
ALS00
U5A
ALS08
U4B
SRDY-
+5V
BUSY- 54 BUSY
ERROR- 53 ERROR
PEACK- 6 PEACK
LOCK- 68 LOCK
PEREQ 61 PEREQ
R80 330
59 NMI
R77 330
A23 7 A23
A22 8 A22
A21 10 A21
A20 11 A20
A19 12 A19
A18 13 A18
A17 14 A17
A16 15 A16
A15 16 A15
A14 17 A14
A13 18 A13
A12 19 A12
A11 20 A11
A10 21 A10
A09 22 A09
A08 23 A08

Reset Connection to UUT

The Sync Module UUT RESET line can be connected to any of several different points on an 80286-based UUT.

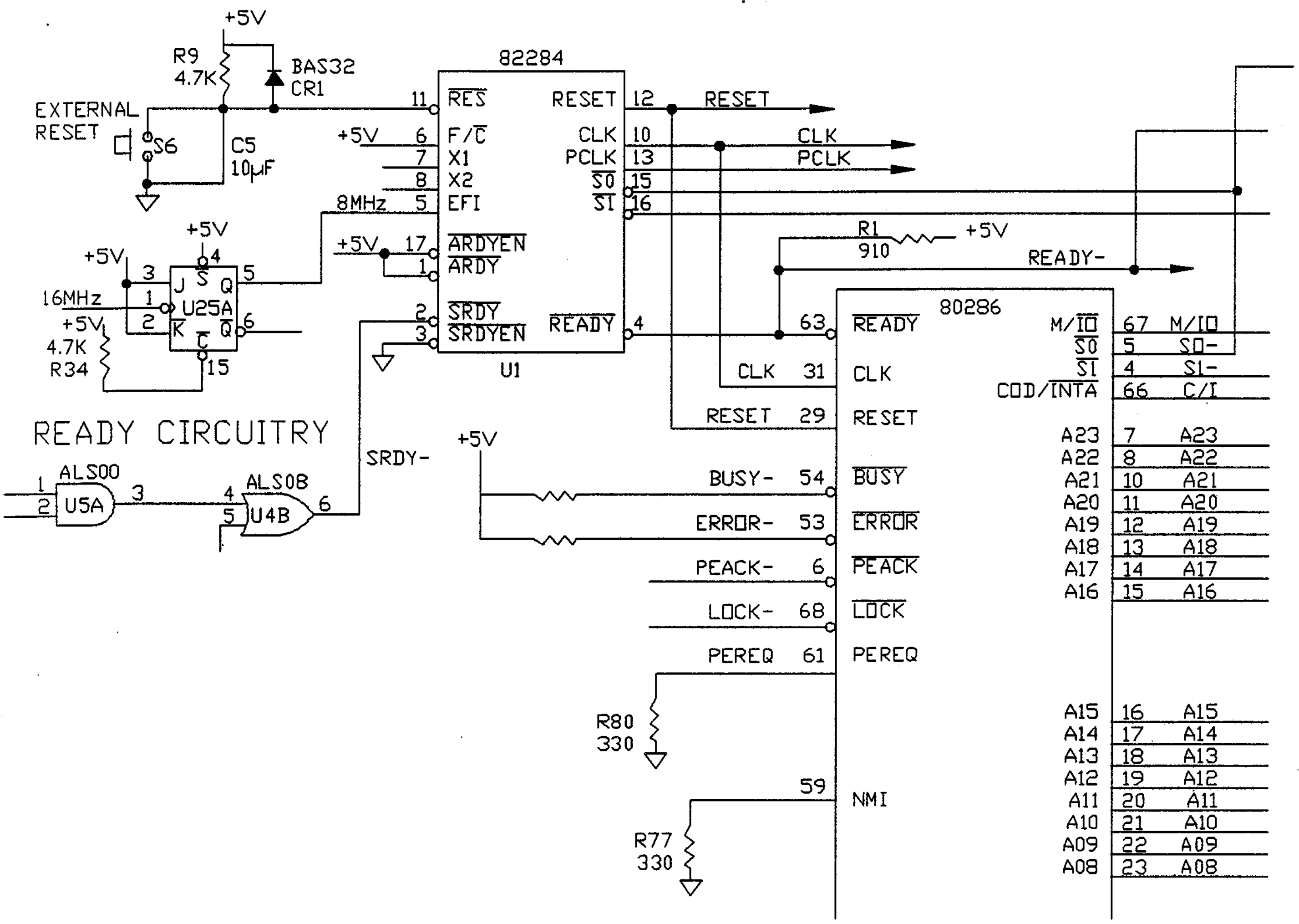
The Sync Module UUT RESET line CANNOT be directly clipped to the 80286 microprocessor. It must be connected to a point that allows the microprocessor RESET input to be synchronized to the CLK signal. It must also allow synchronization of UUT CLK (Phase1/Phase2) and of any UUT bus controller circuits that use CLK.

For a full functional test, the best connection position is one that resets as much of the UUT as possible, like a full system reset. This connection allows the test to start as closely as possible to actual UUT reset start-up conditions. Since this connection usually allows UUT hardware to clear a fault condition which the software cannot, this position is also the “safest”.

Therefore we will connect the Sync Module Reset lead to U1-11.

Connect the Reset lead and the ground lead on the Demo board.

(be sure that power is removed from the UUT)



READY CIRCUITRY
82284
U1
80286
+5V
R9 4.7K
BAS32 CR1
EXTERNAL RESET
S6
C5 10μF
8MHz
16MHz
U25A
R34 4.7K
ALS00 U5A
ALS08 U4B
SRDY-
RES
F/C
X1
X2
EFI
ARDYEN
ARDY
SRDY
SRDYEN
RESET
CLK
PCLK
S0
S1
READY
R1 910
READY-
BUSY-
ERROR-
PEACK-
LOCK-
PEREQ
NMI
R80 330
R77 330
M/IO
COD/INTA
C/I
S0-
S1-
A23
A22
A21
A20
A19
A18
A17
A16
A15
A14
A13
A12
A11
A10
A09
A08

Test Conditions that Cause a Reset

Certain tests conducted by the Pod reset the UUT. Once a reset occurs, some UUTs may require various components to be initialized before testing of the UUT can continue. The tests and conditions that cause a UUT reset are:

Bus Test

STIM_ADR

STIM_DAT

Entering RUN UUT

Exiting RUN UUT

At the first read, write, or HyperRAM access to the UUT after any of the above conditions, after a QWK_RD or QWK_WR, after UUT power has been removed and restored, or after certain types of UUT faults.

ROM Module Connection to UUT

1. Be sure that **power is removed** from the UUT.
2. Remove boot ROM that corresponds to bits 8-15 from the UUT.
3. Install high-byte boot ROM into ROM Module #2.
4. Plug ROM Module #2 into the socket that the high-byte boot ROM had resided.
5. Remove boot ROM that corresponds to bits 0-7 from the UUT.
6. Install low-byte boot ROM into ROM Module #1.
7. Plug ROM Module #1 into the socket that the low-byte boot ROM had resided.
8. If your boot ROMs are soldered-in then refer to Appendix C of the Pod manual.
9. Verify all connections then turn power ON the UUT.(Be sure that power is already on the Mainframe)

We are now ready to setup and calibrate the Pod. There are three ways to do this.

1. The Interactive Setup and Calibration routine.
2. Front panel operations by pressing SETUP.
3. By TL/1 commands.

On the next few pages we will setup and calibrate the Pod using the Interactive Setup and Calibration routine.

Pod Setup

To begin the setup procedure press the [POD R] key on the 9100 keypad. Then select SETUP [F5] and press [ENTER YES]. (Wait for the software to be accessed.)

Select desired actions:

```
INFO    SETUP    CALIBRT    CHECK     QUIT
```

INFO: Information on using POD SETUP

SETUP: Configure the pod for testing your UUT.

CALIBRT: To calibrate the pod for proper operation in your UUT.

CHECK: Verifies that the pod is properly setup and ready to perform tests on faulty UUT's like the known good UUT presently connected.

QUIT: Takes you back to the POD menu

XFER_ADR

Values: 0 to FFFFFF

Default: FFFFF0

Specifies the address that the Pod uses to communicate with the UUT. The transfer address can be set to any UUT address as long as reads or writes to the address do not cause the UUT to halt. Each time data is communicated with the Pod, the UUT microprocessor reads and saves data from the specified address, transfers data to the Sync Module with a write cycle, then restores the original data with a second write cycle. Pod accesses at the XFER_ADR are made with memory byte bus cycles.

ROM_TYPE

Values: 27256, 2716, 2732, 2764, 27128, 27512, OTHER

Default: 27256

Specifies the type of ROM used as the UUT boot ROM.

ROM_MODS

Values: 2, 1

Default: 2

Specifies the number of ROM Modules connected to the Pod. For a word-wide boot ROM, two ROM modules are used. For a byte-wide boot ROM, one ROM Module is used.

RST_POL

Values: LOW, HIGH

Default: LOW

Allows you to set the polarity of the reset signal sent to the UUT by the Pod.

Select INFO and read the presented information. (use the ↓ key to display the next screen)

Select SETUP and the following steps will follow the text on the 9100 screen.

1.
   ```
   Verify UUT communications address.
   XFER_ADR is presently set to $XXX.
   OK CHANGE EXPLAIN
   ```

2.
   ```
   Verify UUT boot ROM type.
   ROM_TYPE is presently set to XXXX.
   OK CHANGE EXPLAIN
   ```

3.
   ```
   Verify number of ROM modules connected.
   ROM_MODS is presently set to X.
   OK CHANGE EXPLAIN
   ```

4.
   ```
   Verify that the ROM modules are connected
   to the UUT's boot ROM sockets.
   OK EXPLAIN
   ```

5.
   ```
   Verify that the SYNC module is connected
   to the UUT.
   OK EXPLAIN
   ```

6.
   ```
   Verify that the RESET lead from the SYNC
   module is connected to the UUT.
   OK EXPLAIN
   ```

7.
   ```
   Verify RESET pulse polarity.
   RST_POL is presently set to XXXXX.
   OK CHANGE EXPLAIN
   ```

RST_LEN

Values: 0 to 9999999

Default: 1000

Specifies the length (in microseconds) of the system reset sent to the UUT by the Pod.

BCYCLCLK

Values: SYNC_MOD, ROM_CE

Default: SYNC_MOD

Specifies desired "Bus Cycle Clock" signal source. Acceptable performance may be available with timing derived from the boot ROM enable signals, though the timing from the Sync Module is usually better.

DATAPRB

Values: YES, NO

Default: NO

Specifies whether the Pod requires that all data bus lines be probed in order to diagnose Bus Test faults. Some UUTs have insufficient data bus hold time to allow reliable data measurement through the Sync Module.

Pod Setup

8. Verify RESET pulse length (micro-seconds).
   RST_LEN is presently set to XXXX.
   OK CHANGE EXPLAIN
9. Verify UUT power is on.
   OK
10. UUT power sensed OK at all ROM module
    connections.
    CONT
11. To verify the RESET connection requires
    use of the probe.
    OK SKIP
12. Probe RESET at the microprocessor and
    press the button on the probe or any key.
13. The RESET pulse was good.
    CONT
14. Checking ROM module data paths. Passed.
    ROM module #1 data path is verified.
    CONT
15. To verify the data paths to the remaining
    ROM modules requires use of the probe.
    OK SKIP
16. Probe D8 at the microprocessor and press
    the button on the probe or any key.
17. ROM module #2 connection is OK.
    CONT
18. The basic pod setups are completed. The following items have been automatically set:

    Setup INTRFACE BCYCLCLK is SYNC_MOD.

    Setup INTRFACE DATAPRB IS NO.

The pod is now ready to begin calibrations. (Select the CALIBRT function).

BURST_SZ

Values: 1, 2

Default: 1

Specifies the ratio of boot ROM addresses accessed to boot ROM enables. Change this setup if your UUT accesses multiple ROM locations during a single ROM enable and uses such "bursts" of data to respond to instruction fetches.

ADR_STIM

Values: 0 to 255

Default: 4

Specifies the number of microprocessor bus cycles expected between UUT reset and the appearance of the stimulus address on the UUT address bus. UUT wait-states and bus width may have an effect on this value.

CY_SPLIT

Values: 1, 2

Default: 1

Specifies the bus width in bytes at the microprocessor divided by the number of ROM modules. Unlike BURST_SZ, this setting does not care if the ROM Modules get selected or not. This setting is the ratio of access width at the microprocessor to that at the ROM Modules.

Pod Setup

19. Select desired actions:

```
INFO    SETUP    CALIBRT    CHECK    QUIT
```

20. Select CALIBRT:

    INFO: General information about CALIBRT

    BUS_TEST: Affects setups which are necessary for proper execution of bus test.

    SYNC: Calibrations which affect the timing of sync pulses generated during UUT read and write accesses.

    RUN_UUT: Adjusts a setup which controls proper entry into the RUNUUT mode.

    MAINMENU: Takes you back to the POD SETUP menu.

Select INFO and read the presented information.

21. Select BUS_TEST

```
Calibrating BCYCLCLK and BURST_SZ . . .
```
(displayed briefly)

```
Completed.
Setup INTRFACE BCYCLCLK is now SYNC_MOD.
Setup INTRFACE BURST_SZ is now 1.

Checking ADR_STIM and CY_SPLIT . . .
(displayed briefly)

Probe A3 at the microprocessor and press the
button on the probe or any key.

Completed.
Setup CALIBRTN ADR_STIM is now 5.
Setup INTRFACE CY_SPLIT is now 1.

The bus test calibrations have been completed.
The pod is now ready to perform read/write
sync pulse calibrations.

(Select the SYNC function).
```

RUN_UUT

Values: 0 to 255

Default: 6

Specifies the number of microprocessor bus cycles expected between UUT reset and the fetch of the instruction at the RUN UUT starting address.

Pod Setup

22. Select SYNC

    ```
    Probe A3 at the microprocessor and press
    the button on the probe or any key.
    ```

    ```
    Calibrating read sync pulses . . .
    ```
    (displayed briefly)

    ```
    Calibrating write sync pulses . . .
    ```
    (displayed briefly)

    ```
    The sync pulse calibrations are completed.
    ```

    ```
    CONT
    ```

23. Select RUN_UUT

    ```
    Calibrating RUN_UUT. . .
    ```
    (displayed briefly)

    ```
    Completed.
    Setup CALIBRTN RUN_UUT is now 5.
    CONT
    ```

24. Select MAINMENU

25. Select CHECK

    ```
    Probe A3 at the microprocessor and press the
    button on the probe or any key.
    ```

    (The following is displayed briefly while checking)

    ```
    Checking Bus Test functions. . .
    Bus Test passed.
    Address stimulus passes.
    Data stimulus passes.

    Checking sync pulse calibrations. . .
    Sync pulse test passes

    Checking runuut calibration
    Runuut test passes

    Checking data transfer address. . .
    Data transfer test passes.

    All checks completed.
    The pod setup is good.
    CONT
    ```

SAVE SYSTEM SETTINGS IN USERDISK

Pod Setup

26. Select QUIT(Read message provided)

27. Save pod setups

We have just completed the setup of the Pod using the Interactive Setup and Calibration routine.

In the Imediate Mode perform a bus test with no faults to verify the setup is correct.

Refer to Appendix D of the Pod Manual and observe the parameters for setup and calibration that can be set using the front panel key SETUP MENU.

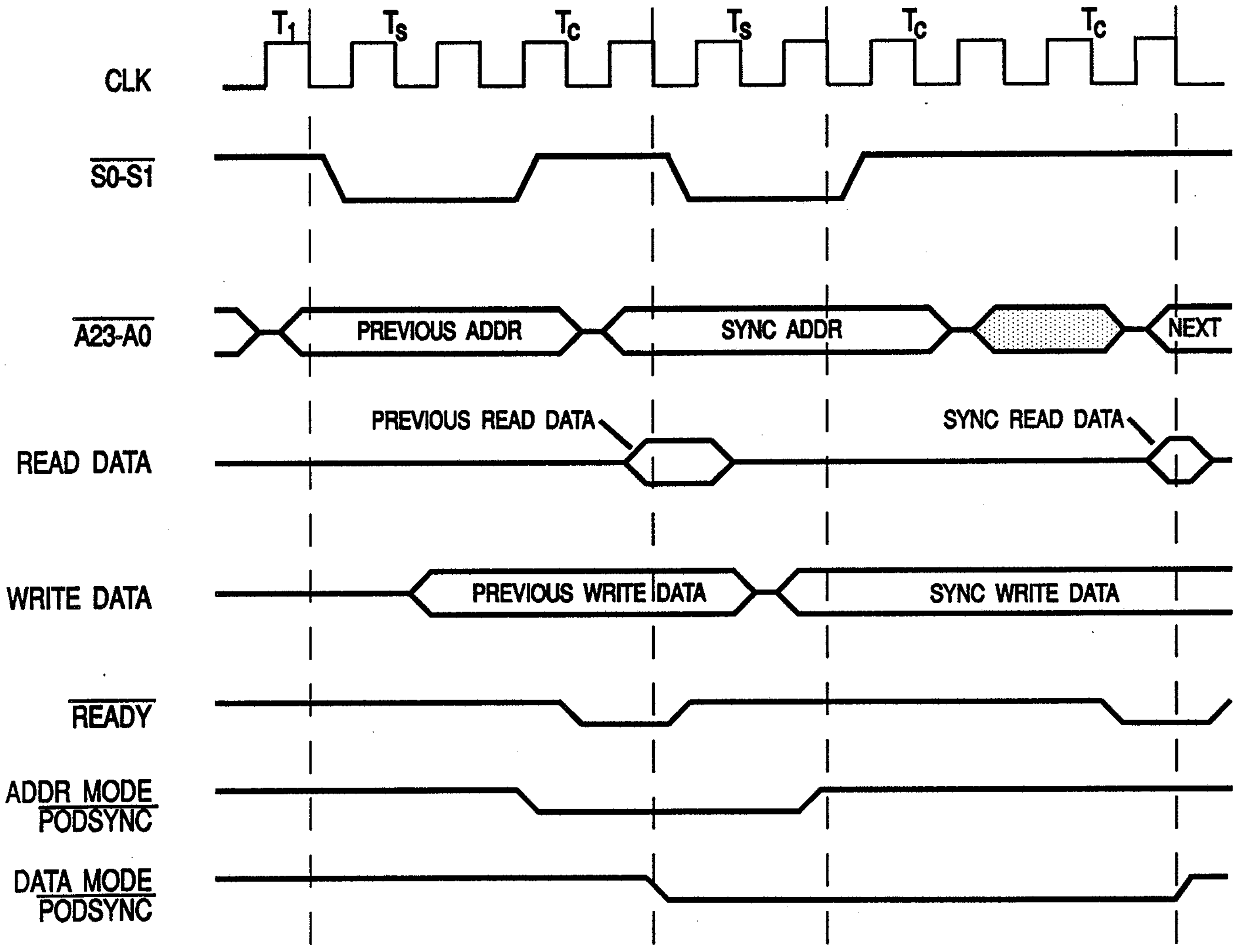
```
program setup

!-----------------------------------------------------------
!
!    This program will setup the Demo. Trainer Bd. for use
!    with the 80286-9132A Pod.
!
!-----------------------------------------------------------

end setup
```

SETUP Program

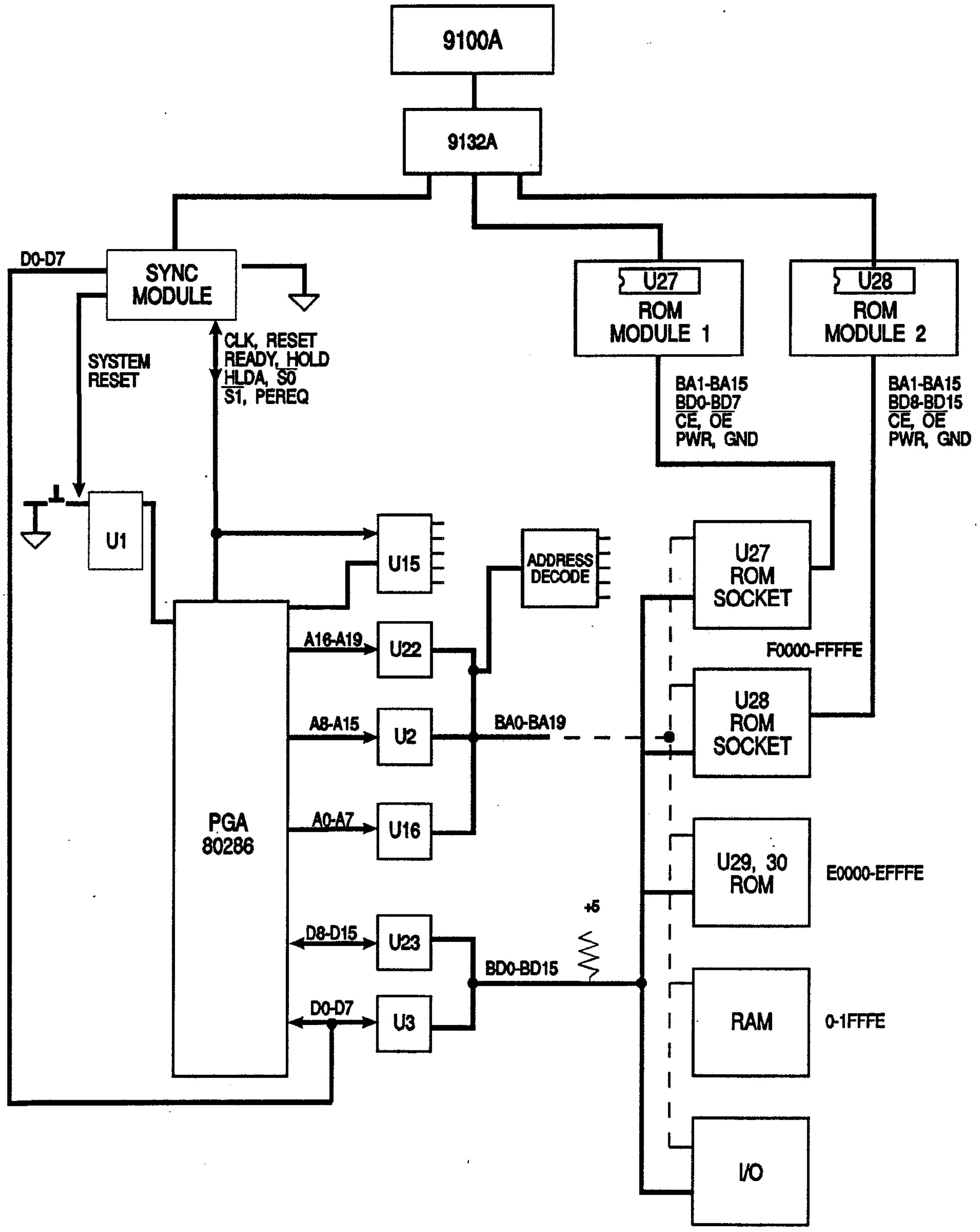
With the help of the Pod Manual Appendix E write a TL/1 program to setup the Demo board.


T1
Ts
Tc
Ts
Tc
Tc
CLK
S0-S1
A23-A0
PREVIOUS ADDR
SYNC ADDR
NEXT
PREVIOUS READ DATA
SYNC READ DATA
READ DATA
WRITE DATA
PREVIOUS WRITE DATA
SYNC WRITE DATA
READY
ADDR MODE
PODSYNC
DATA MODE
PODSYNC

Calibration of the Probe

The purpose of calibration is to ensure that the probe samples signals on the UUT at a known point in time with respect to the UUT timing signals such as CLK and READY. During calibration, delay lines in the Mainframe are adjusted so that the signals being sampled are correctly aligned in time with the clocking signals. The 9132A Pod monitors the 80286 timing signals and produces a PODSYNC signal that is sent to the Mainframe(with some delay). This PODSYNC signal corresponds to the address cycle of the 80286.

Press MAIN MENU on the keypad and calibrate the probe.


9100A
9132A
SYNC MODULE
D0-D7
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
D8-D15
U23
BD0-BD15
D0-D7
U3
+5
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
U29, 30 ROM
E0000-EFFFE
RAM
0-1FFFE
I/O

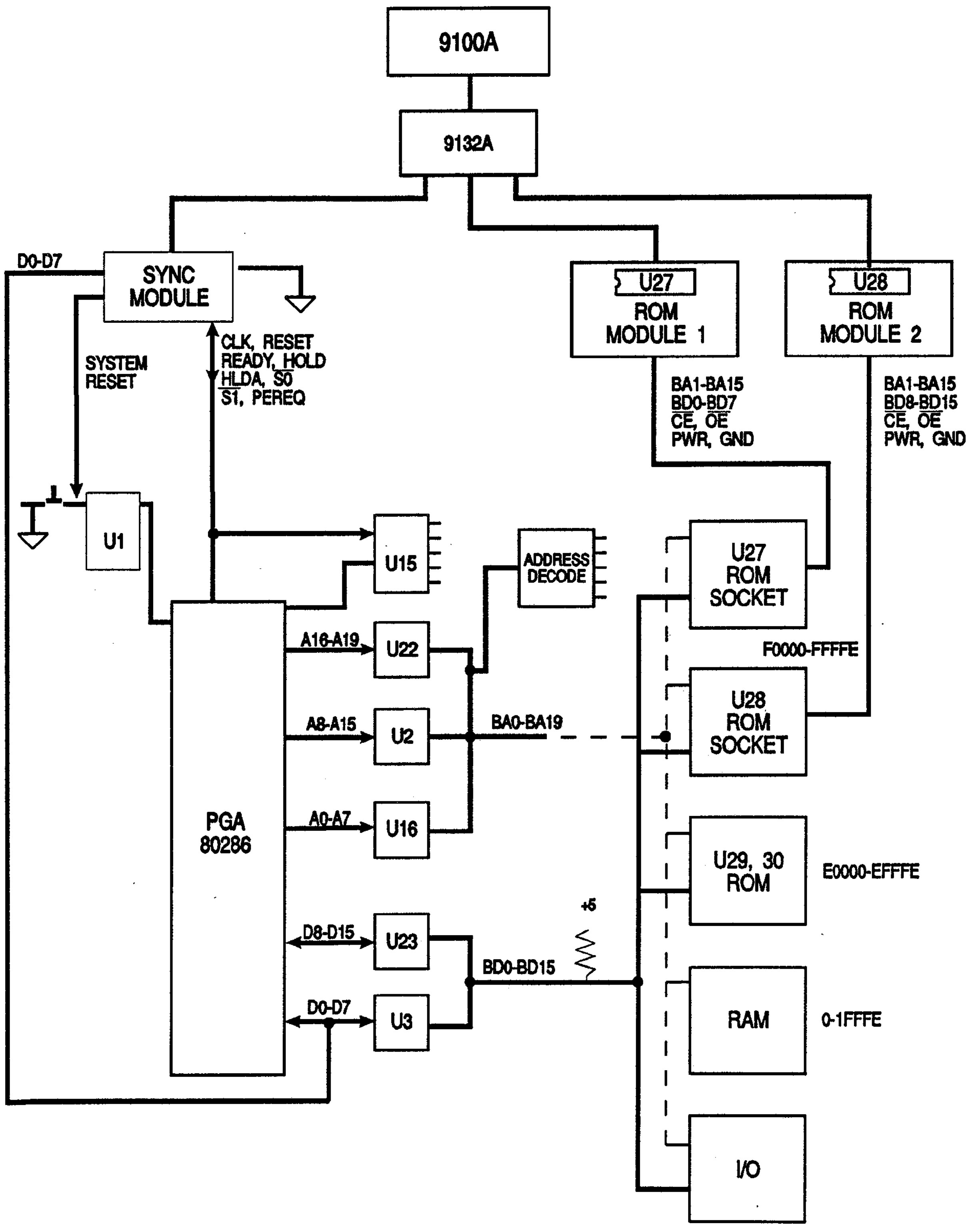
Bus Test

B_TEST

B_TEST is a TL/1 shell program, performed when the front panel key sequence “TEST BUS” is executed, that gives a quick go/no-go indication to the user. B_TEST performs the pass/fail portion of the Bus Test, then a UUT read at the XFER address. If both these tests pass with no fault found, the program returns with the “PASSED” string and exits. If a fault is found, the program prints a message on the Mainframe display and transfers execution to the program TEST_BUS for execution of the fault diagnostics.

B_TEST should be called from TL/1 to execute the UUT kernel test. This program returns quickly if no fault is found, but calls TEST_BUS for diagnostic routines if a fault is found.

Refer to Appendix E-6 of Pod Manual


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
PGA 80286
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

TEST_BUS

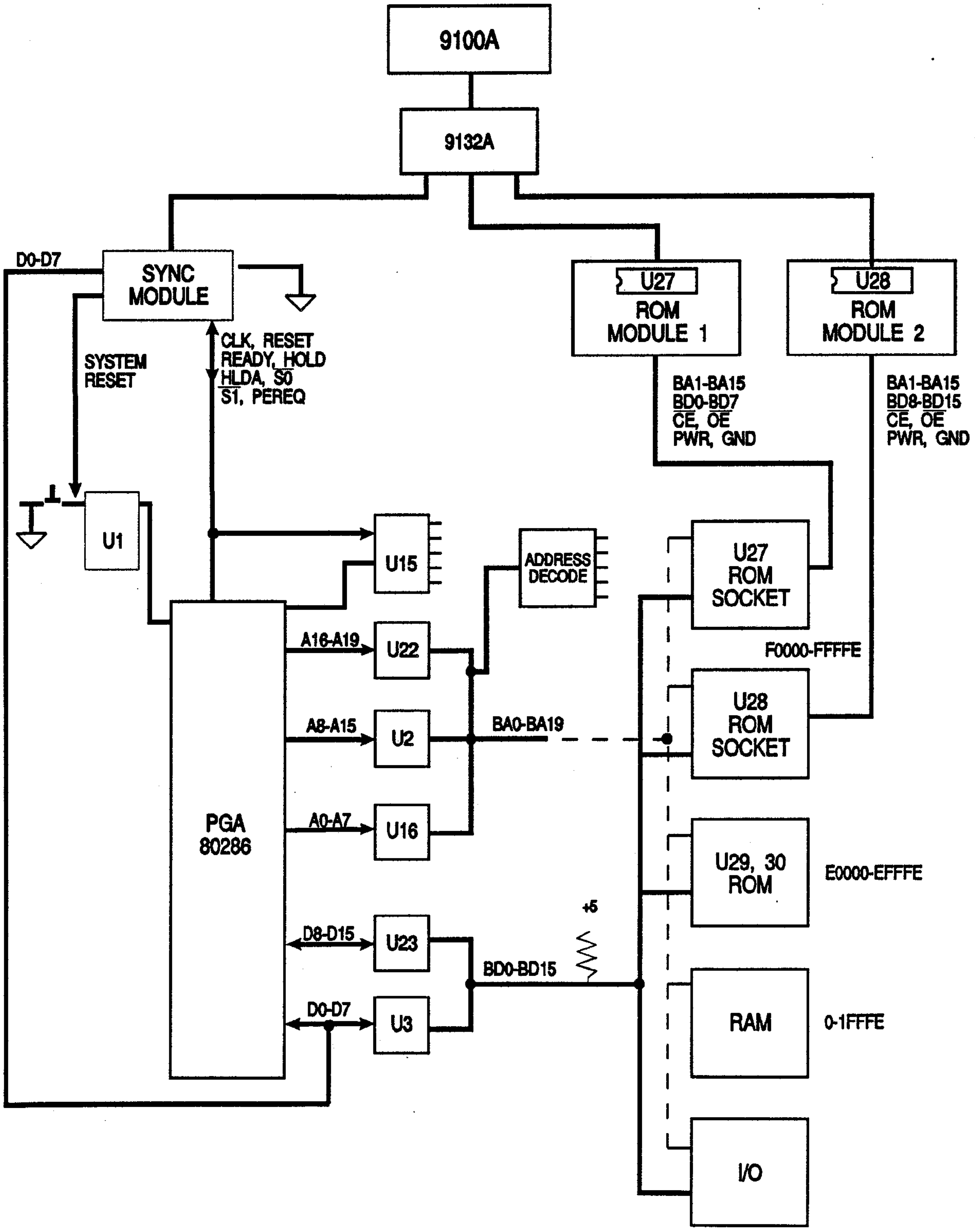
TEST_BUS ia a TL/1 program that performs a thorough test of the UUT kernel, detecting and diagnosing faults that prevent the Pod from performing normal reads and writes. This program first runs a pass/fail test on the UUT bus. If no fault is found there, it performs a UUT read at the XFER address. If no fault is found, the program exits with the pass/fail status set to "pass". If any fault is detected, TEST_BUS then begins to diagnose the fault.

TEST_BUS performs as many diagnostics as possible without any user intervention. The program checks for good power, clock not stopped, no stuck forcing lines, good reset overdrive connections, good ROM Module 1 chip select after reset, correct reset address after reset, data bus integrity after reset, and address bus integrity. Depending on what is detected during these tests, the user may be prompted to use the single-point probe to probe certain lines. When a fault is detected, the normal fault message is raised and shown on the Mainframe display. If the CONT key on the Mainframe keypad is pressed, the program continues to diagnose further, if possible.

Since TEST_BUS is called directly from B_TEST, there is seldom a reason to call TEST_BUS directly.

Run TEST BUS in the immediate mode with no faults.

Refer to Appendix E-7 of the Pod Manual.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

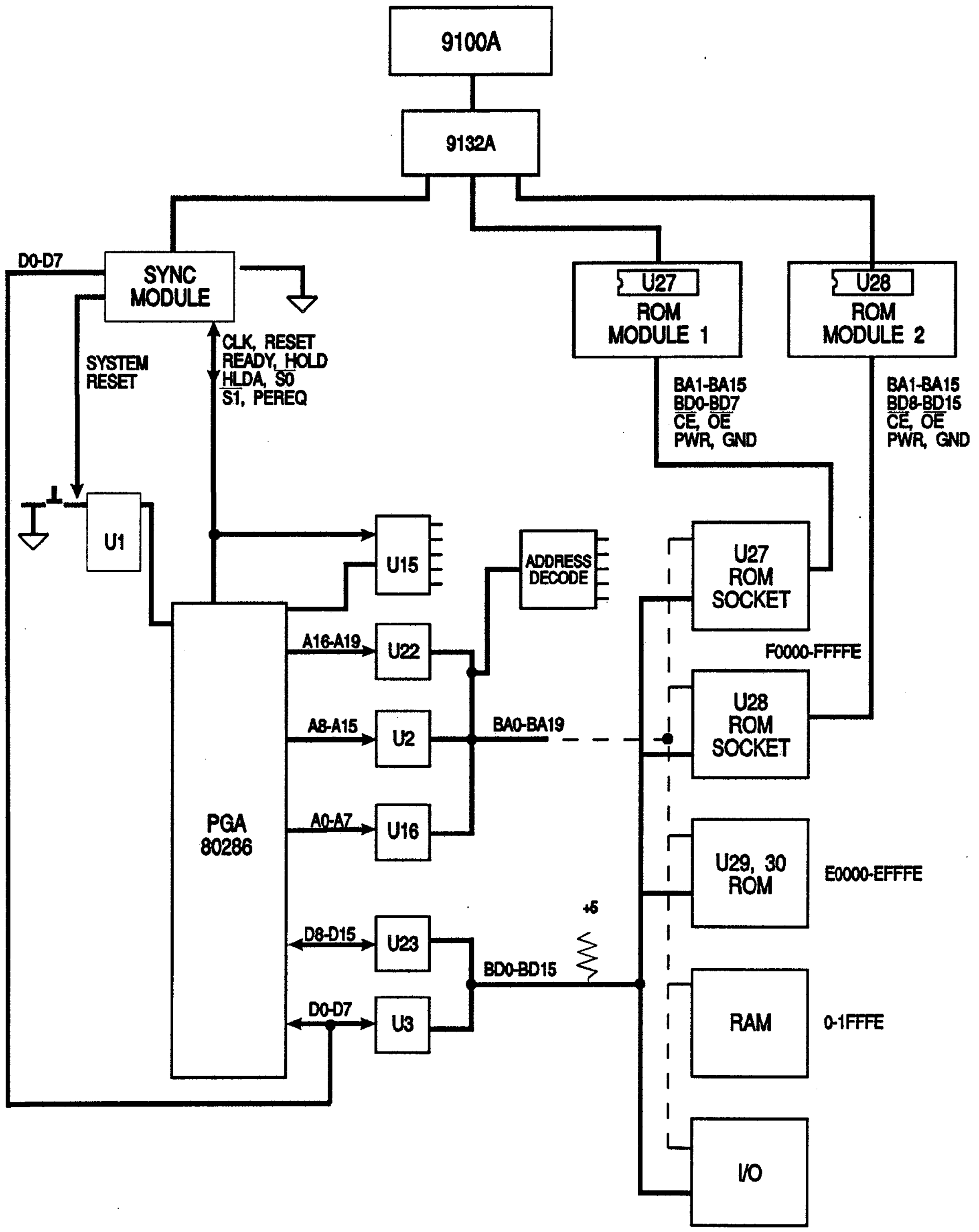
B_DIAG

B_DIAG is a TL/1 program that is executed when the front panel key sequence "DIAGNOSE BUS" is pressed. B_DIAG starts by prompting the user to probe the UUT microprocessor's CLK line (and reports the clock frequency measured by the probe), then prompts the user to probe the status and control lines, reporting in each case whether or not a problem was detected.

B_DIAG should be called if there is some undiagnosed fault in the UUT kernel after running B_TEST.

Run DIAGNOSE BUS in the immediate mode with no faults.

Refer to Appendix E-9 of the Pod Manual.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

Exercise 4-1

Set the following faults in the UUT one at a time and run TEST BUS in the immediate mode. In the space provided write the error message for each fault.

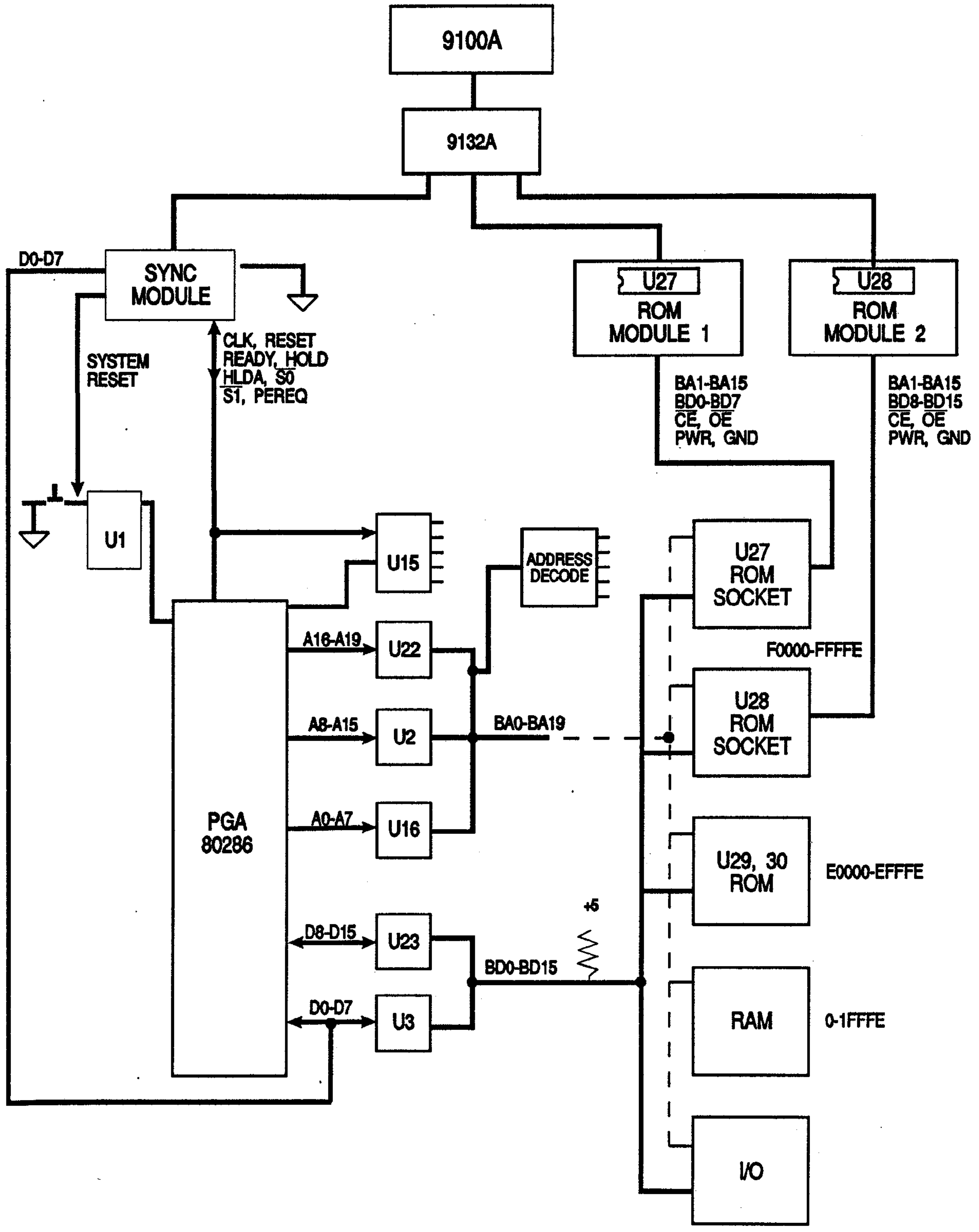
ERROR MESSAGE

Unbuffered

1. Data
   - a. 3-1 : ______________________________
   - b. 2-7 : ______________________________
2. Address
   - a. 2-5 : ______________________________
   - b. 2-3 : ______________________________
3. Control
   - a. 2-1 : ______________________________
   - b. S5 to run : ________________________

Buffered

1. Data
   - a. 4-2 : ______________________________
2. Address
   - a. 4-1 : ______________________________


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
PGA 80286
A16-A19
U22
A8-A15
U2
A0-A7
U16
BA0-BA19
D8-D15
U23
D0-D7
U3
BD0-BD15
+5
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
U29, 30 ROM
E0000-EFFFE
RAM
0-1FFFE
I/O

Fault Isolation

Press POD on the keypad. Your screen should look like:

```
    POD: QWK_RD
    ADDR OPTION: MEMORY WORD
    QWK_RD QWK_WR STIM_ADR STIM_DAT SETUP
```

STIM_DAT

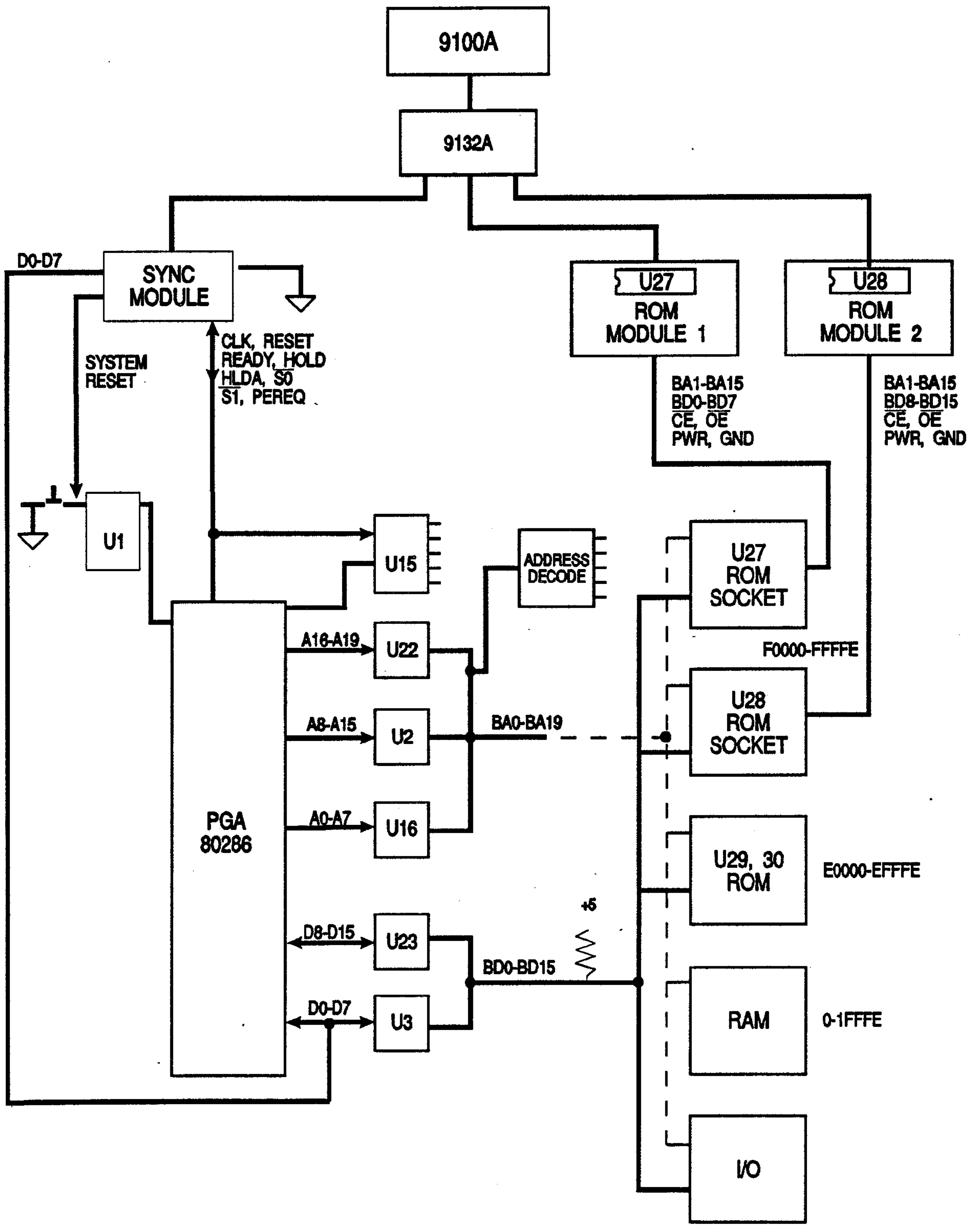
This program stimulates the data bus by resetting the processor and causing "data" to be fetched over the data bus by the microprocessor, while simultaneously generating a sync pulse. Because this program causes the microprocessor to put the reset address on the address bus, it is also useful for troubleshooting reset address faults and ROM Module chip select faults.

STIM_DAT is the most basic stimulus routine used for diagnosing UUT kernel faults by TEST_BUS. This program is useful in stimulus programs for testing inoperative kernels.

Arguments: DATA

Returns:	$0	No faults detected.
	$1	For any detected fault.
	$20	No CS detected at ROM Module 1.
	$40	Bad reset address detected at ROM Module 1.

Refer to Appendix E-11 of the Pod Manual.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

STIM_ADR

This program stimulates the address bus by causing the UUT microprocessor to place the lower 16 bits of the entered address on the UUT address lines, while simultaneously generating a sync pulse. For this program to return with no faults detected requires that the microprocessor be able to successfully fetch several words of data from the ROM Modules. This routine is useful for troubleshooting address bus faults (as long as they were not RESET ADDRESS or ROM1_CS_OE faults)

STIM_ADR is a stimulus routine used for diagnosing UUT kernel faults by TEST_BUS. This program is useful in stimulus programs for testing inoperative kernels.

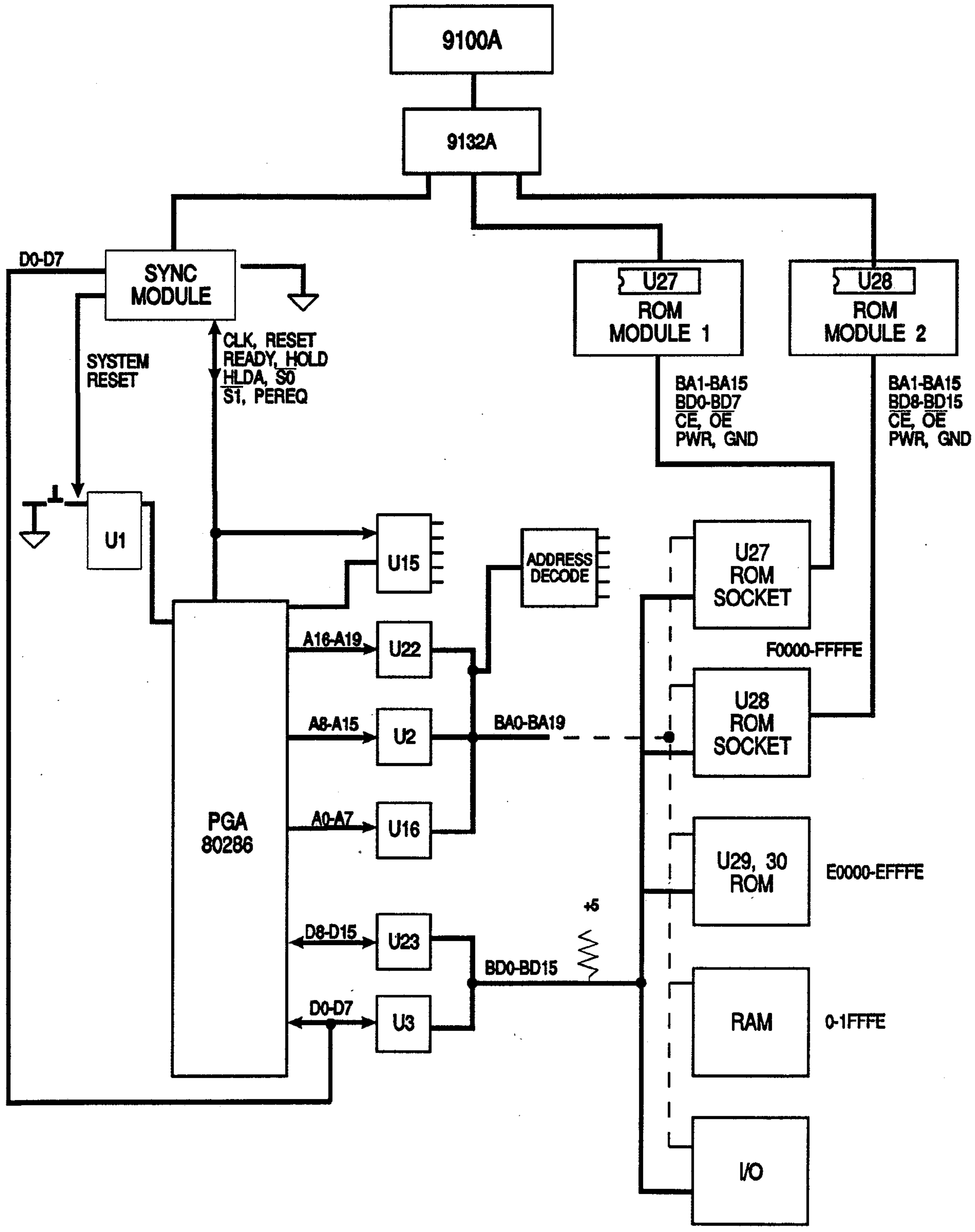
Arguments:	ADDR	
Returns:	$0	No faults detected. The address sensed at ROM
		Module 1 matched the command ADDR.
	$1	For any detected fault.

Refer to Appendix E-12 of the Pod Manual.

QWK_RD

This program causes the Pod to implement a quick looping read function. Once QWK_RD is entered, the program returns at once with the value of the data found at the given address. Though only one value is returned, the Pod continues to perform reads at the specified address. A sync pulse is generated for each read, as specified by the current sync mode. Reading continues until a Pod access of any kind is initiated. (Because the UUT is reset each time QWK_RD is exited, looping on this quick function is not recommended.)

Arguments:	ADDR
Returns:	The data at the specified address.


9100A
9132A
SYNC MODULE
D0-D7
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

QWK_WR

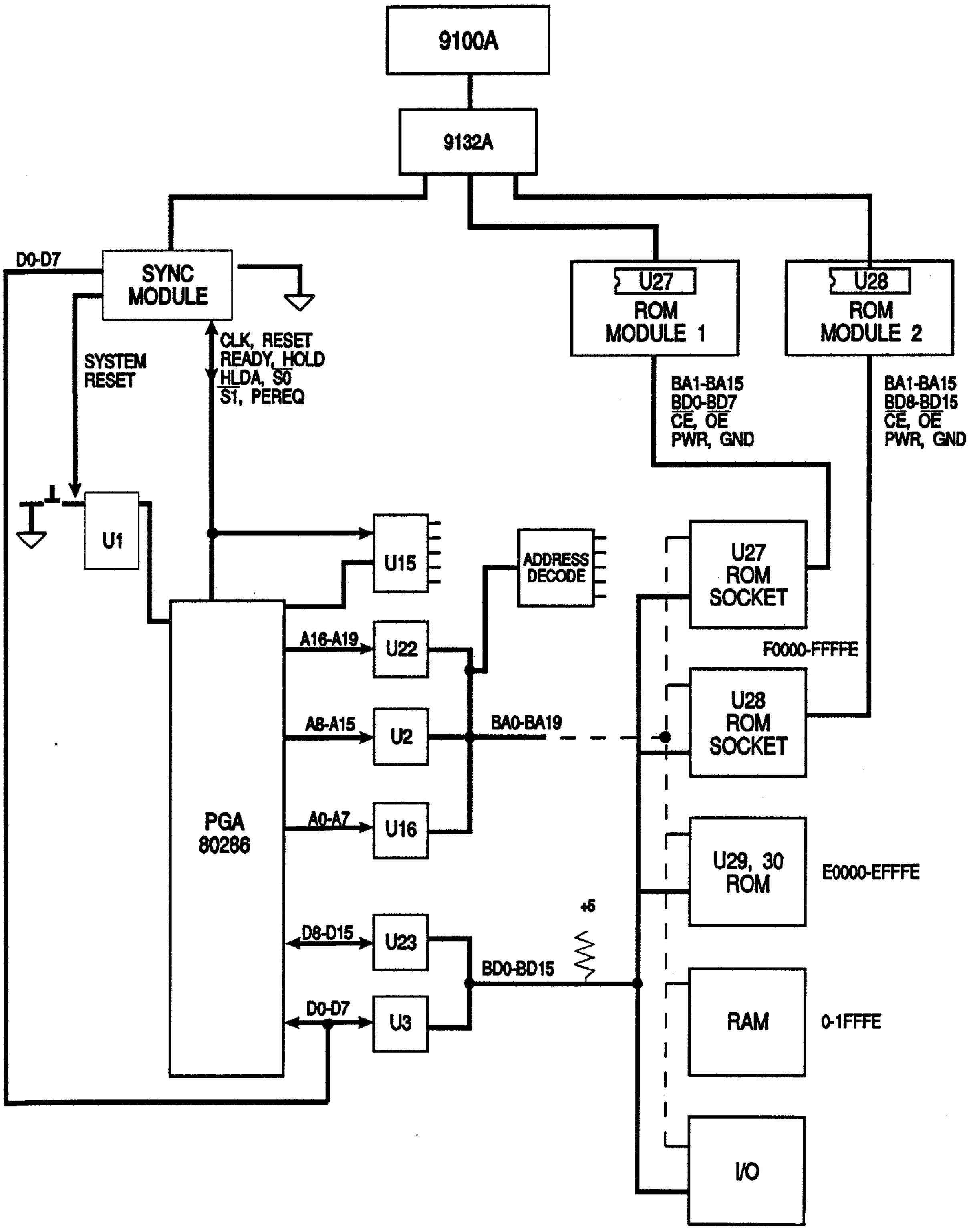
This program causes the Pod to implement a quick looping write function. Once QWK_WR is entered, the program returns at once. Though the program returns immediately, the Pod continues to perform writes at the specified address. A sync pulse is generated for each write, as specified by the current sync mode. Writing continues until a Pod access of any kind is initiated. (Because the UUT is reset each time QWK_WR is exited, looping on this quick function is not recommended.)

Arguments:	ADDR
	DATA
Returns:	Nothing

READ FAST FOREVER ADDR XXXXXX

WRITE FAST FOREVER DATA XXXX TO ADDR XXXXXX

The read/write fast forever functions are useful when troubleshooting stuck and shorted lines when you can only sync on freerun. They are also helpful when you can sync to data or address.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

Exercise 4-2

Set the following faults in the UUT one at a time and run TEST BUS in the immediate mode. Try each of the stimulus routines just discussed to find the bad node.

Fault 3-1

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________

Fault 2-7

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________

Fault 2-5

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________

Fault 2-3

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________

Fault 4-2

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________

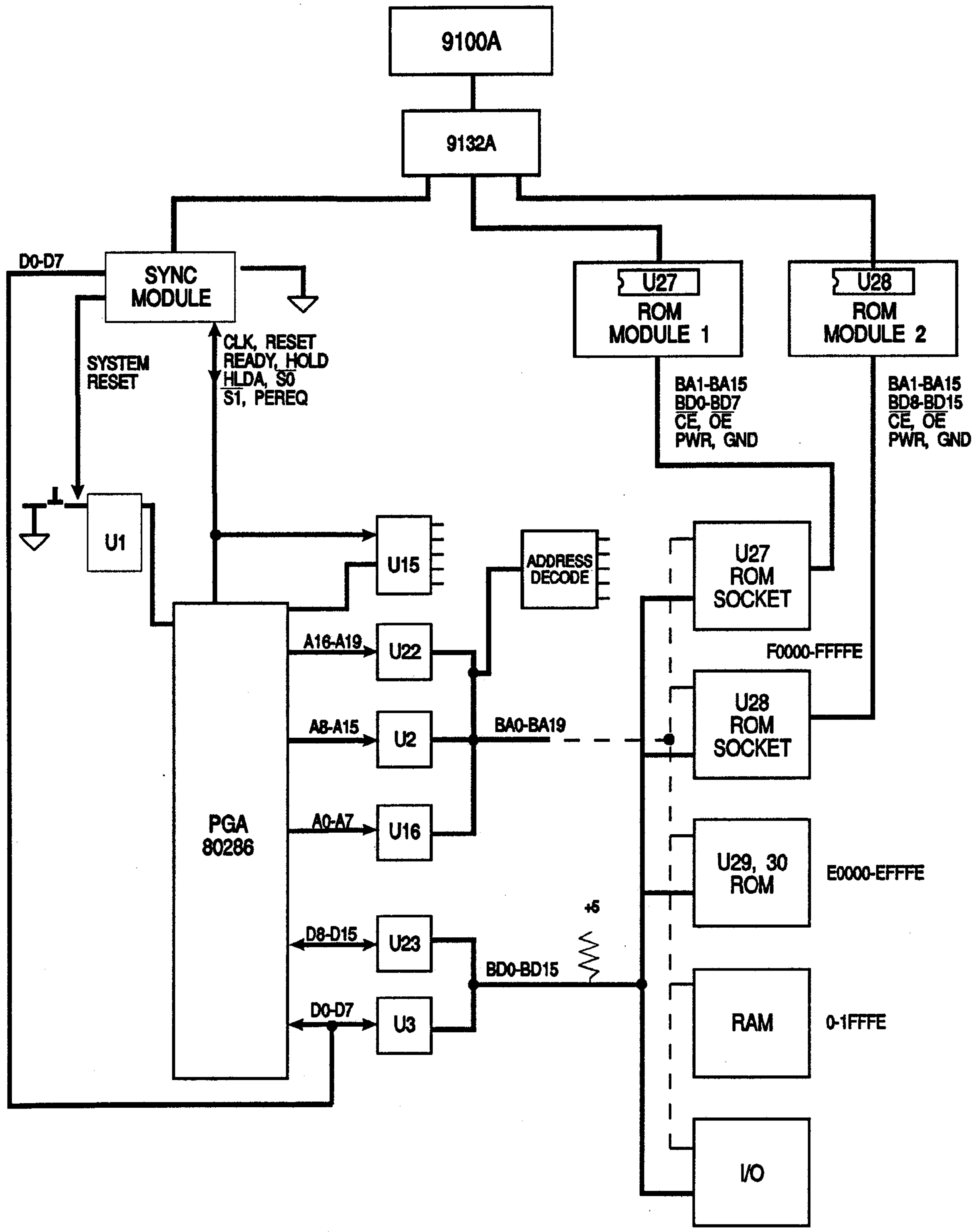
Fault 4-1

Fault Message: ______________________

Stimulus: ______________________

Sync: ______________________

Bad Node: ______________________


9100A
9132A
SYNC MODULE
D0-D7
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
PGA 80286
A16-A19
U22
A8-A15
U2
A0-A7
U16
D8-D15
U23
D0-D7
U3
BA0-BA19
BD0-BD15
+5
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
U29, 30 ROM
E0000-EFFFE
RAM
0-1FFFE
I/O

Testing the UUT RAM

RAM Fast Test

The RAM Fast test is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable.

HyperRAM Test

RAM fault coverage is essentially the same as the RAM Fast test. The difference is that the HyperRAM test is completed in much less time.

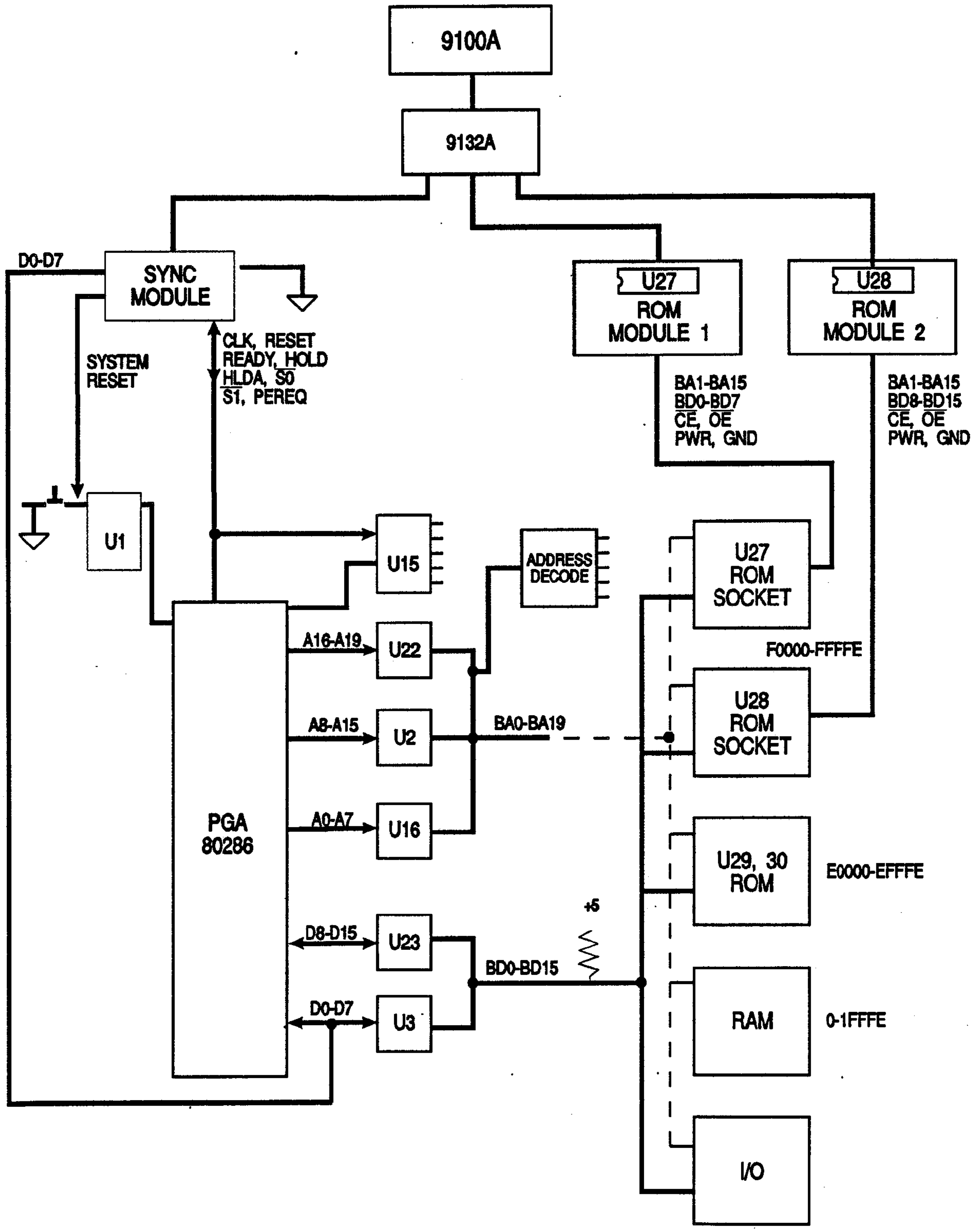
RAM Full Test

The RAM Full test is the most comprehensive RAM test algorithm available in the 80286 Pod. Besides the fault types detected by other algorithms available in the Pod, the RAM Full test makes several additional passes through memory to help find coupling faults.

Verify HyperRAM Test and RAM Fast Test with no faults. (Demo trainer RAM map is $0-$1FFFE)

Refer to Appendix E-12 of the Pod Manual.

Refer to Appendix E-16 of the Pod MAnual.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
PGA 80286
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

Exercise 4-3

Insert the following faults one at a time and determine the bad node. Fill in the blanks for each fault.

Fault 4-2

Tests	Fault Messages
1. ____________	__________________________
2. ____________	__________________________

Stimulus: ______________________________________

Sync: ____

Bad Node: ____________________________________

Fault 1-1

Tests	Fault Messages
1. ____________	__________________________
2. ____________	__________________________

Stimulus: ______________________________________

Sync: ____

Bad Node: ____________________________________

Fault 1-5

Tests	Fault Messages
1. ____________	__________________________
2. ____________	__________________________

Stimulus: ______________________________________

Sync: ____

Bad Node: ____________________________________

Fault 4-8

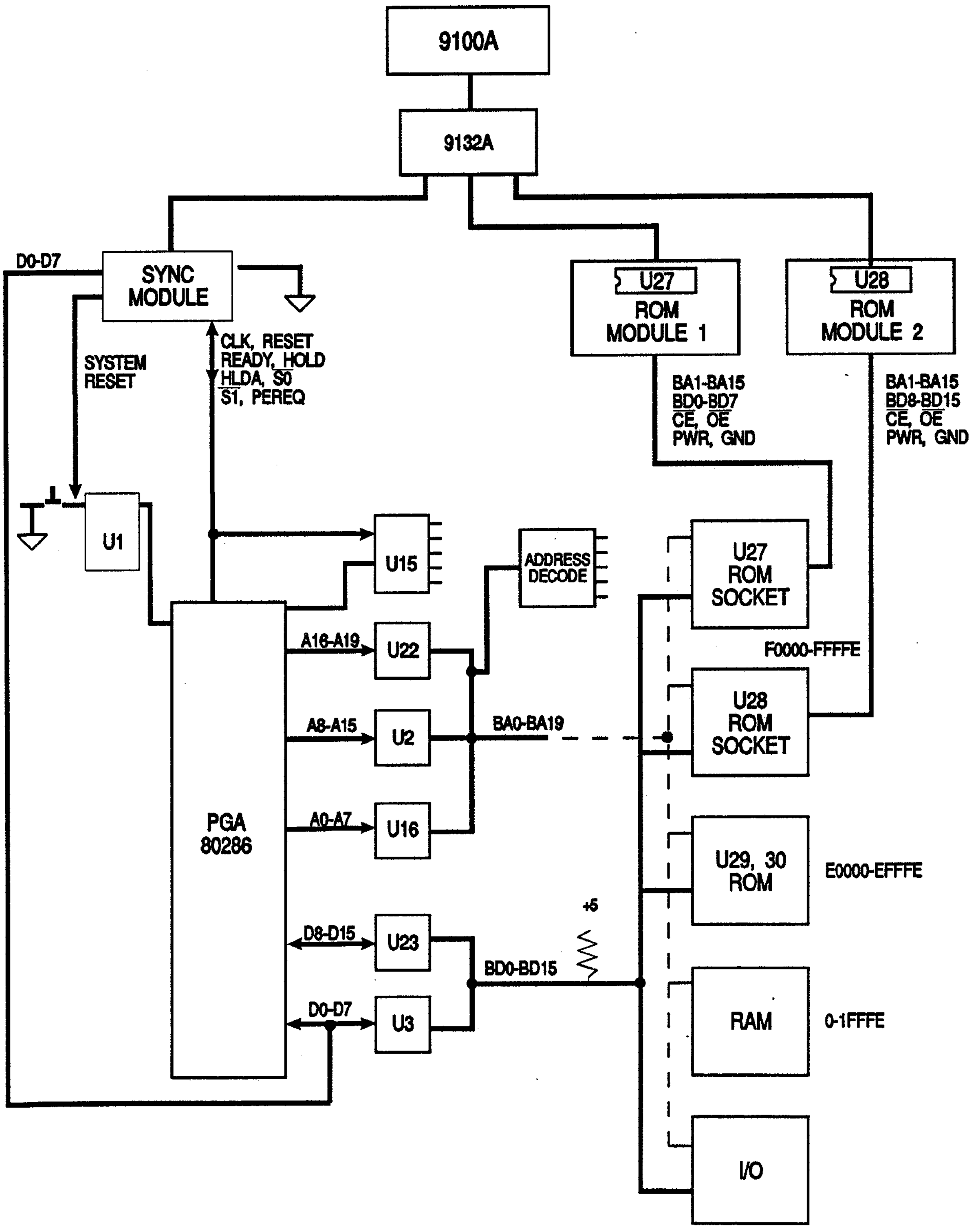
Tests Fault Messages

1. ____
2. ____

Stimulus: ______________________________________

Sync: ____

Bad Node: ____________________________________


9100A
9132A
SYNC MODULE
D0-D7
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
PGA 80286
A16-A19
U22
A8-A15
U2
A0-A7
U16
BA0-BA19
D8-D15
U23
D0-D7
U3
BD0-BD15
+5
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
U29, 30 ROM
E0000-EFFFE
RAM
0-1FFFE
I/O

Gathering Signatures

Using the self test socket

1. Turn the UUT power OFF.
2. Unplug ROM Module 1 from the UUT.
3. Open the self test socket door of the Pod.
4. Insert ROM Module 1 into the ZIF self test socket.
5. Press ROM on the keypad and select `GET SIG`.
6. Your screen should look like:

```
GET SIG ROM REF U27 ADDR 0 UPTO 7FFF MASK FF
STEP 1
ADDR OPTION: ST_ROM
```

7. Press ENTER YES and wait for the signature. `SIG = F387`
8. Repeat steps for ROM Module 2. `SIG = 8E6E`
9. Install ROM Modules back to the UUT.

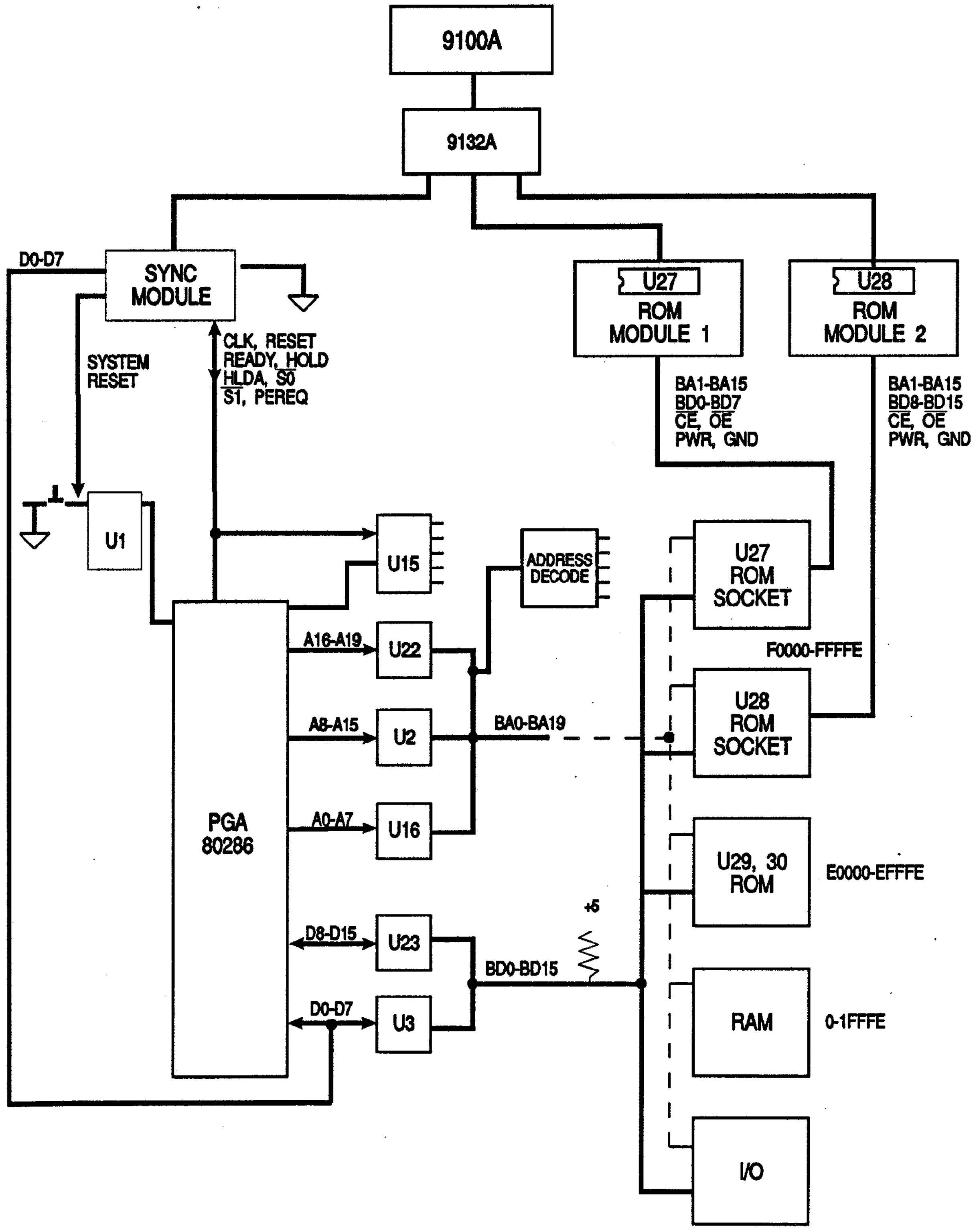
Using the ROM Module plugged into the UUT

1. Verify all connections.
2. Turn the UUT power ON.
3. Verify Bus Test passes.
4. Press ROM on the keypad and select `GET SIG`.
5. Your screen should look like:

```
GET SIG ROM REF U27 ADDR F0000 UPTO FFFFE MASK
FF STEP 2
ADDR OPTION : UUT_ROM
```

6. Press ENTER YES and wait for the signature. `SIG = F387`
7. Repeat steps for ROM Module 2. `SIG = 8E6E`

```
EXCEPT: U28 ADDR F0001 UPTO FFFFF MASK FF STEP 2
```


9100A
9132A
SYNC MODULE
D0-D7
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

From other UUT ROMs

1. Use normal method as used with the standard 80286 Pod.
2. Gather the signatures for U29 and U30.

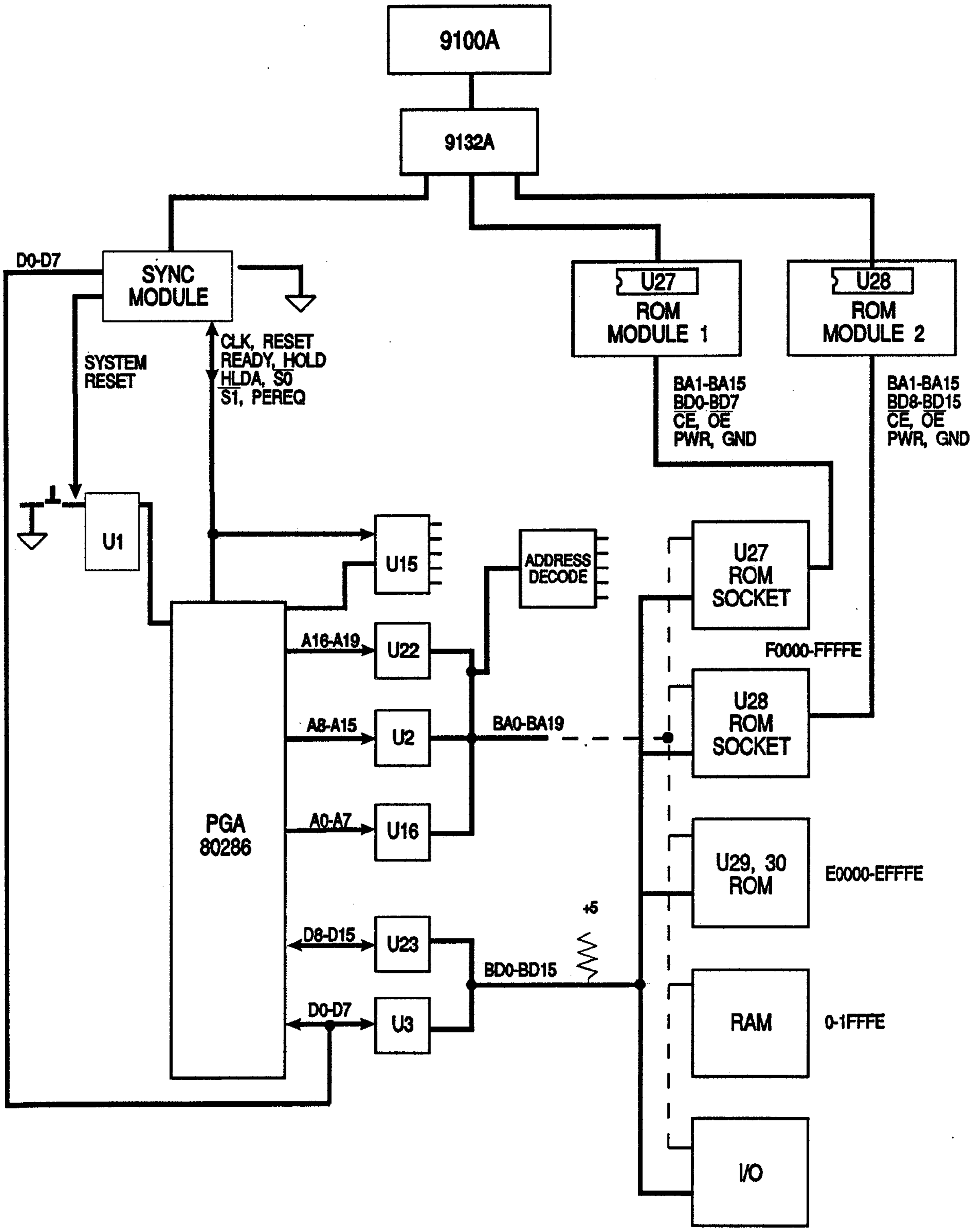
UUT Boot ROMs soldered to the UUT

To obtain the ROM signature of UUT boot ROM soldered onto the UUT, the Pod must be able to disable the boot ROM. (See Appendix C-6 of the Pod Manual)

ROM Test

A ROM Test must be performed using the same method as the signature was gathered.

Use TEST ROM FULL ALL REF to test the ROMs.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

Exercise 4-4

Insert the following faults one at a time and determine the bad node. Fill in the blanks for each fault.

Fault 1-2

Tests	Fault Messages
1. ____________	__________________________
2. ____________	__________________________
3. ____________	__________________________

Stimulus: ______________________________________

Sync: ____

Bad Node: ______________________________________

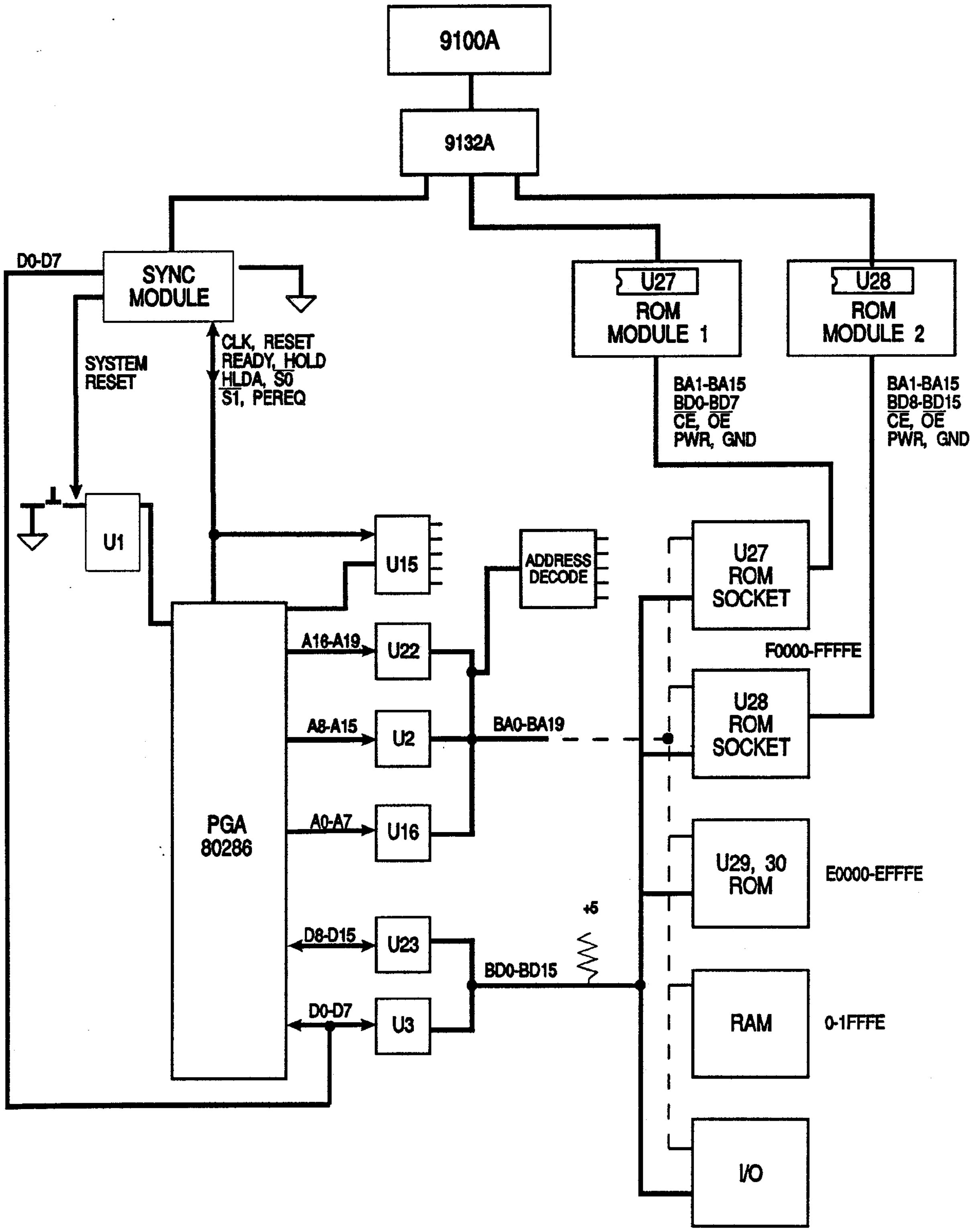
Fault 3-7

Tests	Fault Messages
1. ____________	__________________________
2. ____________	__________________________
3. ____________	__________________________

Stimulus: ______________________________________

Sync: ____

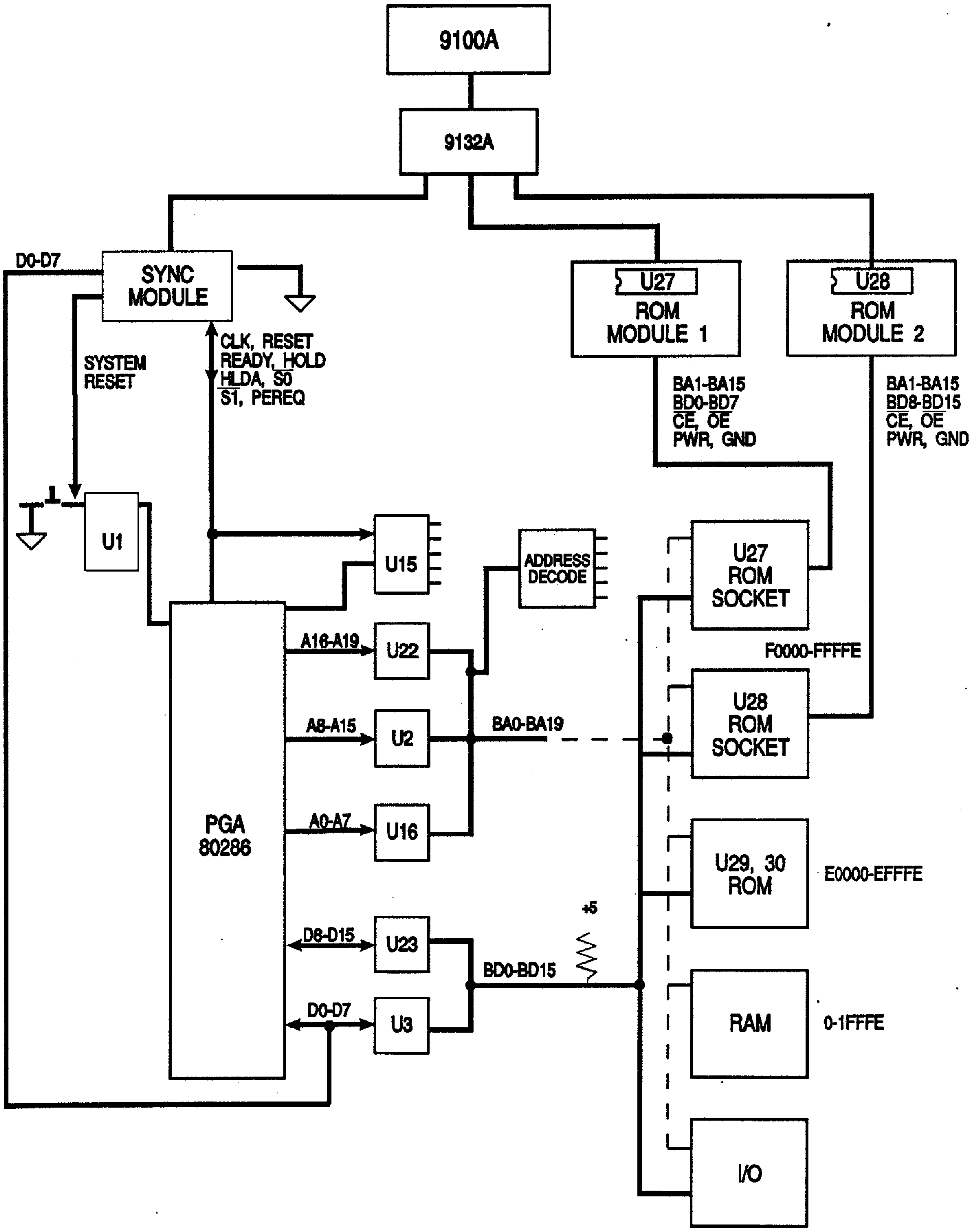
Bad Node: ______________________________________


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

Troubleshooting Hints

When you begin testing, certain conditions on the UUT may cause the Pod to behave unpredictably. If this happens, check the following list to see if one of the conditions exists on your UUT.

1. Some UUT's may require various components to be initialized before the UUT can be successfully tested by the Pod. You must determine what UUT components require initialization and the method for initializing these components.
2. If the cache system of the UUT overlays the UUT boot ROM area, the cache must be disabled for the Pod to operate correctly. If it is not disabled, the processor may fetch from the cache rather than the emulation memory in the Pod.
3. Some UUT's may require the setup attributes to be changed in order for the UUT to function correctly.
4. Ensure that the power supply voltage is at the correct level (in some cases an incorrect voltage may cause some tests to proceed correctly while others inexplicably fail).
5. Watchdog timers and other circuits than can disrupt operation may need to be disabled.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27
ROM MODULE 1
U28
ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
PGA 80286
A16-A19
U22
A8-A15
U2
A0-A7
U16
BA0-BA19
D8-D15
U23
D0-D7
U3
BD0-BD15
+5
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
U29, 30 ROM
E0000-EFFFE
RAM
0-1FFFE
I/O

Problems Due to a Marginal UUT

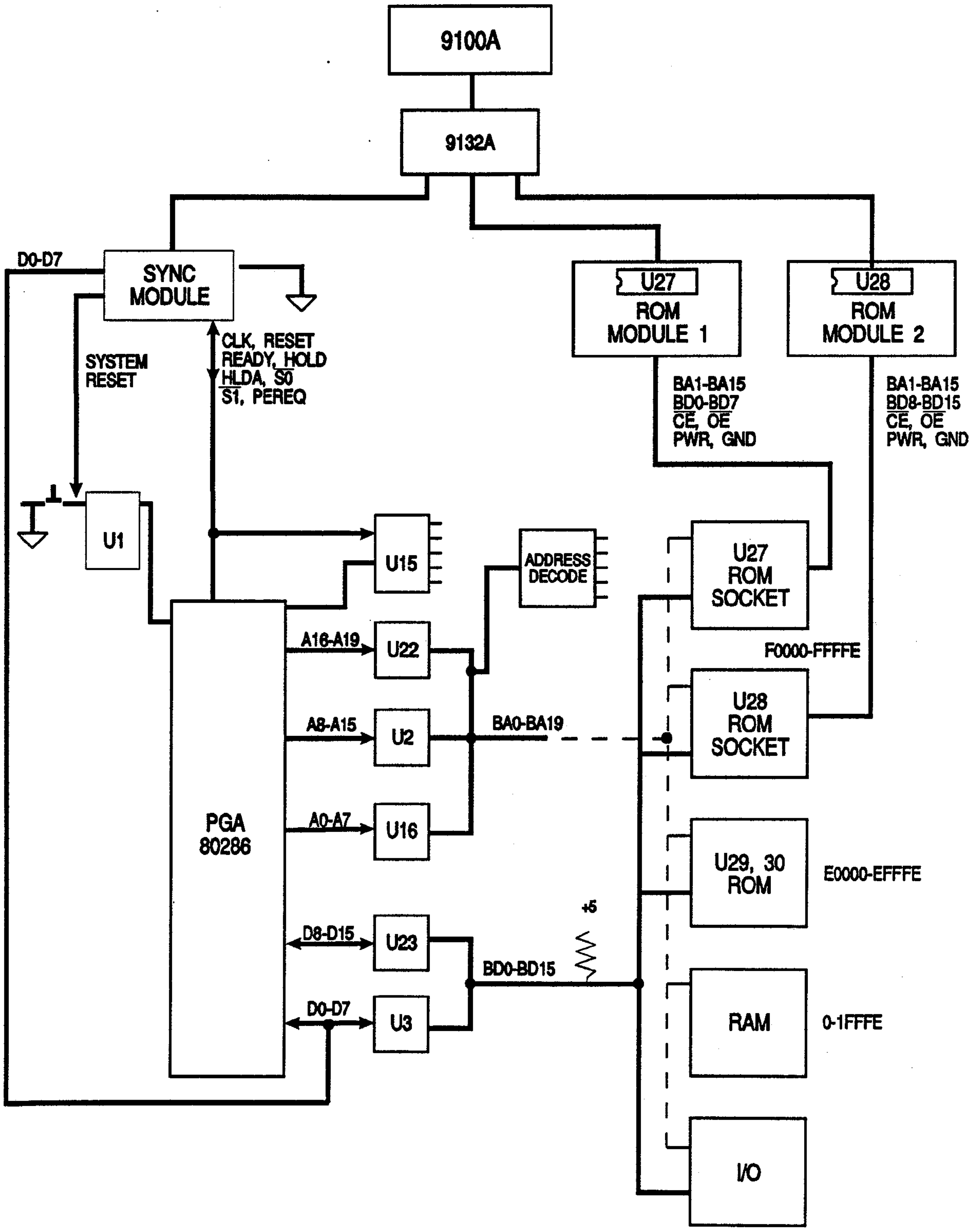
The Pod is designed to approximate, as closely as possible, the actual characteristics of the ROMs that it replaces in the UUT. However, the Pod does differ in some respects. In general, these differences tend to make marginal UUT problems more visible. A UUT may operate marginally with the UUT ROMs installed, but exhibit errors with the Pod plugged in. Since the Pod differences tend to make marginal UUT problems more obvious, the UUT becomes easier to troubleshoot. Various UUT and Pod operating conditions that may reveal marginal problems are described in the paragraphs that follow.

Some UUTs operate at speeds that approach the time limits for memory access. The Pod contributes a slight time delay that causes memory access problems to the boot ROM in the ROM Module sockets to become apparent.

As long as the UUT noise level is low enough, normal operation is unaffected. Removing the UUT from its chassis or case may disturb the integrity of the shielding to the point where intolerable noise could exist. The Pod may introduce additional noise. In general, marginal noise problems can be made worse (and easier to troubleshoot) through use of the Pod and Mainframe.

The Pod loads the UUT slightly more than the UUT ROMs. The Pod also presents more capacitance than the ROMs. These effects tend to make any bus drive problems more obvious.

The Pod slightly increases the normal load on the UUT clock. While this loading rarely has any effect on clock operation, it may make marginal clock sources more obvious.


9100A
9132A
D0-D7
SYNC MODULE
SYSTEM RESET
CLK, RESET
READY, HOLD
HLDA, S0
S1, PEREQ
U27 ROM MODULE 1
U28 ROM MODULE 2
BA1-BA15
BD0-BD7
CE, OE
PWR, GND
BA1-BA15
BD8-BD15
CE, OE
PWR, GND
U1
U15
ADDRESS DECODE
U27 ROM SOCKET
F0000-FFFFE
U28 ROM SOCKET
PGA 80286
A16-A19
U22
A8-A15
U2
BA0-BA19
A0-A7
U16
U29, 30 ROM
E0000-EFFFE
+5
D8-D15
U23
BD0-BD15
D0-D7
U3
RAM
0-1FFFE
I/O

GO_NOGO Program

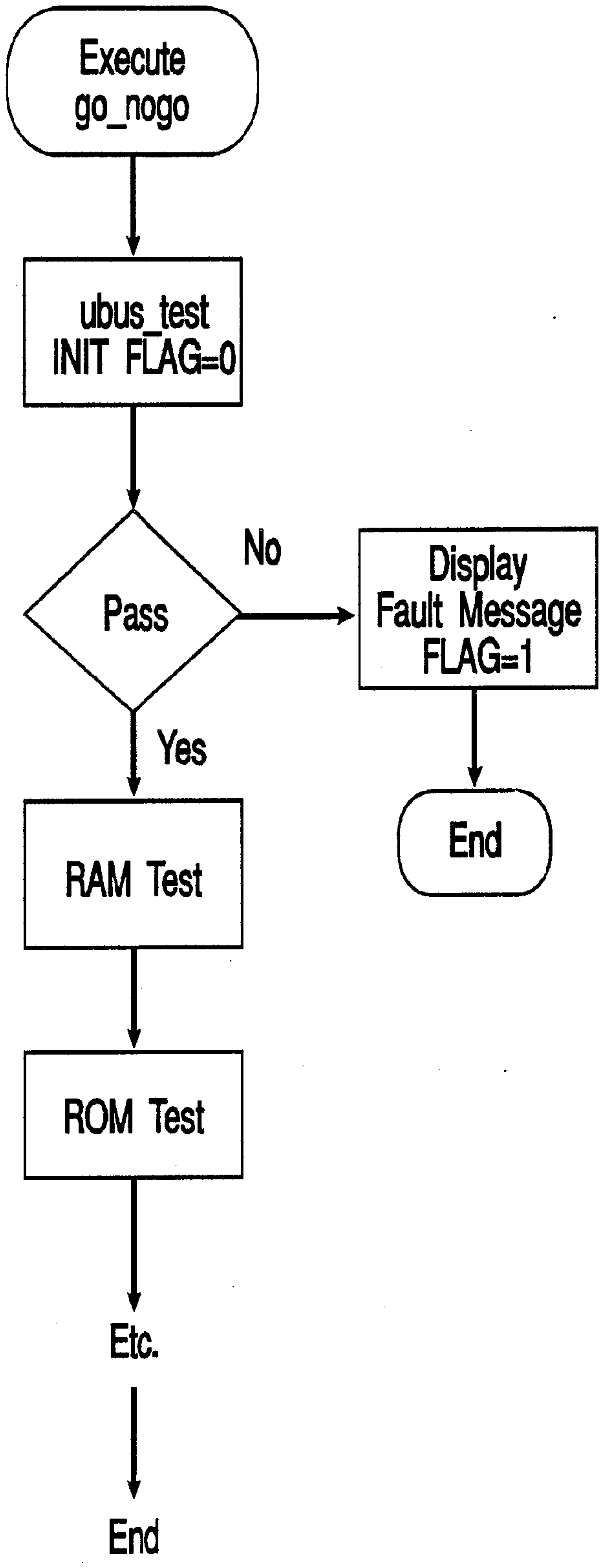
Write a GO_NOGO program to test the BUS, RAM, and ROM of the Demo Trainer Board.

Verify the GO_NOGO program works properly with no faults on the UUT.

Have instructor put faults in the Demo board and use your program to find the faults.

If time permits add TL/1 support programs to expedite fault isolation.

Section 5
UFI: Memory Emulation

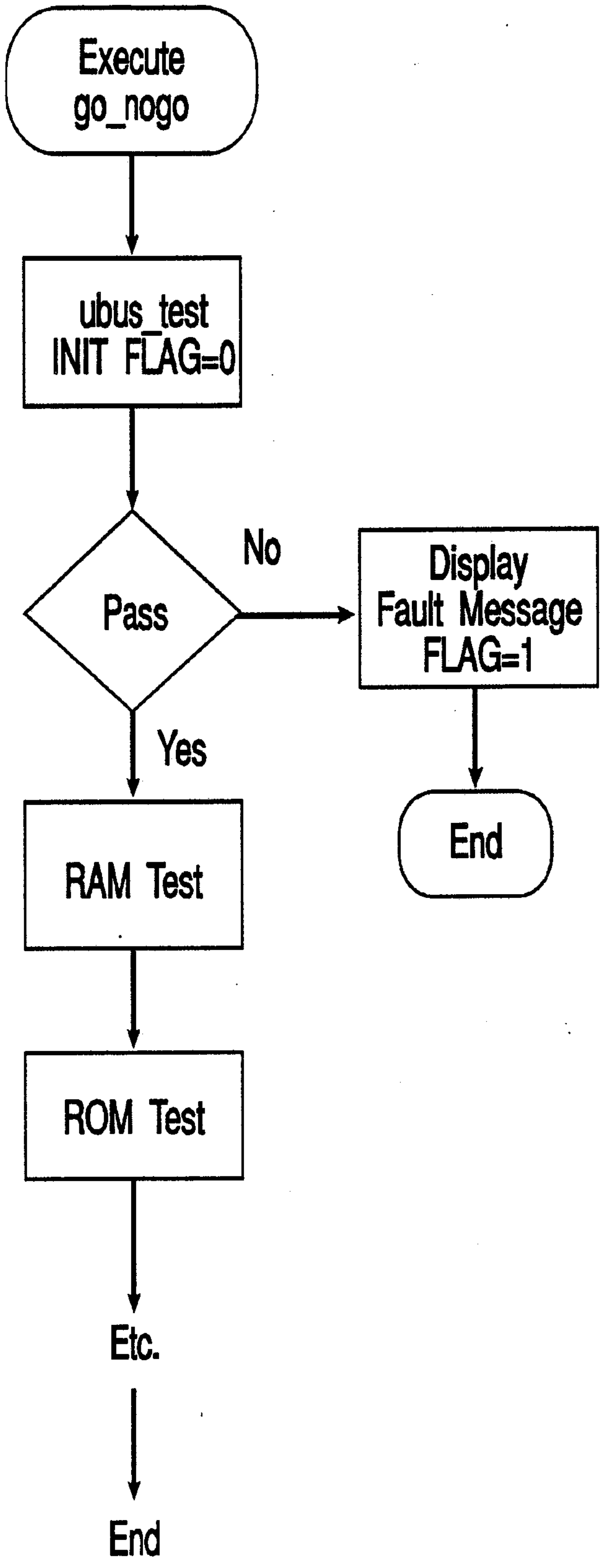

Execute go_nogo
ubus_test
INIT FLAG=0
Pass
No
Display Fault Message FLAG=1
End
Yes
RAM Test
ROM Test
Etc.
End

By now you should see there are some very important concepts that need to be taken into account when you develop UFI or GFI using a 9132 Memory Device Emulation Pod.

Remember, the UFI and GFI algorythms cause all stimulus programs that stimulate a node to be executed and are checked for proper response. This could cause some major problems if the Bus Test failed since any attempt to perform a read or write access at the UUT causes loss of microprocessor control.

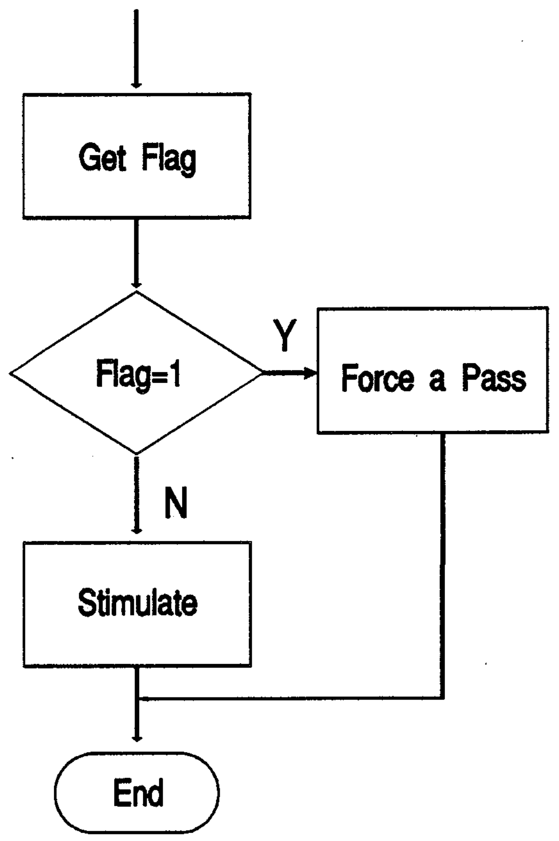
One Method of overcomming this situation is to use a "software switch". The software switch will turn off any programs that cause loss of microprocessor control.

The following programs are examples of how the software switch is used. These examples use a text file for storing the software switch contents but a persistent variable could also have been used.


Execute go_nogo
ubus_test INIT FLAG=0
Pass
No
Display Fault Message FLAG=1
End
Yes
RAM Test
ROM Test
Etc.
End

```
#INFO    /HDR/TRNR9132/UBUS_TEST (PROGRAM) 13:17 03/08/90
NAME: UBUS_TEST                                                DISK FREE:   331,008 BYTES
DESCRIPTION:                                                   SIZE: 1,473 BYTES
                                                               WRITE PROTECT: NO

program ubus_test
handle bus_data_low_tied(mask)
    declare
        string mask
        string cls = "\1B[2J"
        string curoff = "\1B[?25l"
        string line1 = "\1B[1;1H"
        string line3 = "\1B[3;1H"
        string beep = "\07"
    end declare
    b = val(mask,2)
    n = lsb b
    t1=open device "/term1/win", mode "unbuffered", as "output"
    print on t1, cls,beep,curoff
    print on t1,     "====== TROUBLESHOOT DATA BIT D",n," ======"
    print on t1, line3,"       ====PRESS STOP====          "
fault
end handle
handle
    fault
end handle
    declare
        string curoff = "\1B[?25l"
    end declare
    stat_fil_o = open device "/hdr/trnr9132/status", as "output"
    t1=open device "/term1/win", mode "unbuffered", as "output"
    print on stat_fil_o,"0"     close(stat_fil_o)
    print on t1,curoff, "TESTING BUS ... "
    setspace(getspace("memory","word"))
    ! PERFORM THE 9132 BUS TEST !
    b_stat = b_test()
    ! IF B_TEST FAILED, THEN SET FLAG IN STATUS FILE !
    if b_stat = "FAILED" then
        stat_fil_o = open device "/hdr/trnr9132/status", as "output"
        print on stat_fil_o,"1"
        close(stat_fil_o)
    ! ... AND LOCK UP
        loop
        end loop
    end if
end program
```


Get Flag
Flag=1
Y
Force a Pass
N
Stimulate
End

```
#INFO   /HDR/TRNR9132/ROM0_DATA (PROGRAM) 12:33 03/08/90
NAME: ROM0_DATA                                              DISK FREE:   327,936 BYTES
DESCRIPTION:                                                 SIZE:803 BYTES
                                                             WRITE PROTECT: NO

program rom0_data
    handle
        fault
    end handle
    declare
        string x
        stringrefer
        numeric stat_fil_i
    end declare
    setspace(getspace("memory","word"))
    if (gfi control) = "yes" then
        dev = gfi device
    else
        dev = "/probe"
    end if
    report_off()
    reset device dev
    sync device dev, mode "pod"
    sync device "pod", mode "data"
! OPEN THE STATUS FILE - CHECK IF TEST_BUS FAILED . . .
! IF IT DID, DO NOT CHECK ROM0_DATA RESPONSES
    stat_fil_i = open device "/hdr/trnr9132/status", as "input"
    input on stat_fil_i, x
    close(stat_fil_i)
    if x = "1" then
        arm device dev
        readout device dev
        refer = gfi ref
        gfi pass refer
    else
        arm device dev
            rampaddr addr $E0000, mask $1FE
        readout device dev
    end if
end program
```

```
#INFO   /HDR/TRNR9132/K_ADDR (PROGRAM) 12:46 03/08/90
NAME: K_ADDR                                        DISK FREE:   331,008 BYTES
DESCRIPTION:                                        SIZE:            706 BYTES
                                                    WRITE PROTECT: NO

program k_addr
    handle
        fault
    end handle
        t1 = open device "/term1/win", as "output", mode "unbuffered"
        print on t1,"ADDRESS STIMULUS"
        wait(2000)
        close (t1)
        if (gfi control) = "yes" then
            dev = gfi device
        else
            dev = "/probe"
        end if
    setspace(getspace("memory","word"))
    report_off()
    reset device dev
    sync device dev, mode "pod"
    sync device "pod", mode "addr"
    arm device dev
        stim_adr ($FFFF)
        stim_adr ($FF00)
        stim_adr ($F0F0)
        stim_adr ($CCCC)
        stim_adr ($AAAA)
        stim_adr ($5555)
        stim_adr ($3333)
        stim_adr ( $F0F)
        stim_adr (  $FF)
        stim_adr (   $0)
    readout device dev
end program
#INFO   /HDR/TRNR9132/K_DATA_IN (PROGRAM) 12:48 03/08/90
NAME: K_DATA_IN                                     DISK FREE:   331,008 BYTES
```

DESCRIPTION:

SIZE: 709 BYTES
WRITE PROTECT: NO

```
program k_data_in
handle
    fault
end handle
    t1 = open device "/term1/win", as "output", mode "unbuffered"
    print on t1,"DATA_IN STIMULUS"
    close (t1)
    if (gfi control) = "yes" then
        dev = gfi device
    else
        dev = "/probe"
    end if
    setspace(getspace("memory","word"))
        report_off()
    reset device dev
    sync device dev, mode "pod"
    sync device "pod", mode "data"
    arm device dev
        stim_dat ($FFFF)
        stim_dat ($FF00)
        stim_dat ($F0F0)
        stim_dat ($CCCC)
        stim_dat ($AAAA)
        stim_dat ($5555)
        stim_dat ($3333)
        stim_dat ( $F0F)
        stim_dat (  $FF)
        stim_dat (   $0)
    readout device dev
end program
```

Section 6
Using Diagnostics

Appendix A
Glossary

ADDRESS BUS

A number of lines used by the microprocessor for locating data in RAM, ROM, or I/O devices. DMA controllers may also use the address bus for transferring large blocks of data to or from memory. Each line is referred to individually as an address line.

ALGORITHM

A prescribed set of rules or procedures for solving a complex problem.

ASCII

American Standard Code for Information Interchange as defined by ANSI document X3.64-1077. ASCII is a standardized code set for 128 characters including full alphabet (upper and lower case), numerics, punctuation, and a set of control codes.

ASYNCHRONOUS DATA/CIRCUITS

Data or other signals that function independently of microprocessor bus cycles.

ASYNCHRONOUS MEASUREMENT

A technique used by the 9100A to measure digital signals not synchronized to the microprocessor bus timing.

BACKTRACING

A procedure for locating the source of a fault on a UUT by taking digital measurements along a signal path from bad outputs to bad inputs. Backtracing stops at the point where a bad output has all good inputs.

BLOCK ADDRESS

A set of contiguous addresses defined by a beginning and ending address.

BLOCK READ

TL/1 command that reads a specified block of UUT addresses to a text file using a specified format (Intel or Motorola).

BLOCK WRITE

TL/1 command that writes a text file using a specified format (Intel or Motorola) to a specified block of UUT addresses.

BREAKPOINT

Stops program execution when a previously defined condition occurs. The TL/1 debugger uses a line oriented breakpoint that stops program execution just prior to executing the specified program line. The new interface pods have a breakpoint feature that stops RUN UUT execution when a specified UUT address appears on the address bus.

BUFFER

1: A device used to isolate one circuit from another (i.e. a bus buffer prevents a bus failure in one circuit from interfering with the same bus in another circuit). **2:** Used to boost the drive capability of the circuit being buffered. **3:** Provides temporary data storage for data transfers, usually between memory and I/O devices.

BUS

A series of bi-directional lines connected from the microprocessor to each device that interchanges data with it. Only one device can transmit at a time while any number of other devices can receive or be placed in a tri-state condition.

CAD (Computer-Aided Design)

Electronic CAD systems let the user create, manipulate, and store designs on a computer. Used mainly for laying out complex, high density, multi-layered printed circuit boards.

CAE (Computer-Aided Engineering)

Very similar to CAD systems except they are used more for design simulation, verification, and optimization.

CAS (Column Address Strobe)

A line used by dynamic RAM to latch the column address into the RAM device.

CAPTURE SYNCHRONIZATION

Synchronizes the response gathering hardware with vector output timing generated by the Vector Output I/O Module.

CONTROL LINE

An output line from the microprocessor that controls a particular function such as Read or Write. Also an input to a device which enables, disables, or controls the device.

CRC (Cyclic Redundancy Check)

A CRC (often referred to as signature) is a four digit hexadecimal number representing a serial stream of binary data. The binary data is shifted through a special 16 bit register which when complete, results in a 16 bit signature remaining in the register.

DATA BUS

A bus used to move parallel data between the CPU, memory, and I/O devices.

DEBUG

Locating and removing hardware and software errors.

DEBUGGER (9100A)

A TL/1 Editor tool used to test the functionality of TL/1 programs and debug execution errors.

DIRECTORY

A collection of related data sets (i.e. program files, text files) on a disk.

DMA (Direct Memory Access)

A technique for quickly transferring large amounts of data directly between I/O devices and memory or between memory and memory. The DMA controller supplies address and control signals required to accomplish the transfer directly rather than going through the CPU.

EDITOR

The programming environment that provides accessibility to files and automated test development.

EXTERNAL SYNCHRONIZATION

This technique uses the external Start, Stop, Clock, and Enable lines on either the I/O or Probe clock modules to provide measurement timing.

FAULT

A defect in a UUT that causes the circuitry to operate incorrectly.

FAULT COVERAGE

Normally expressed as a percentage of faults that a board test can detect of all possible faults.

FAULT DETECTION

The process of determining if a UUT has faults. See FUNCTIONAL TEST.

FAULT ISOLATION

The process of locating the component or defect causing the failure.

FUNCTIONAL TEST

A test that provides a pass/fail status on a specific section of the UUT circuitry.

GFI (Guided Fault Isolation)

An automated backtracing algorithm used to locate the source of a failure.

HANDSHAKING

Incorporates the use of special control lines or control codes to coordinate the transfer of data between two or more devices.

ICT (In-Circuit Test)

A test performed on one component at a time to verify component functionality; does not check for dynamic interaction between components.

IMMEDIATE MODE

The operating mode in which the operator interacts with the 9100A via the Applications keypad and display. Actions specified are performed as the proper keys are pressed.

INTERFACE POD

Translates generic mainframe commands (read,write,bus test, etc.) into microprocessor machine code to accomplish the desired action at the UUT.

INTERRUPT

An input to the microprocessor that is activated by an I/O device when it is ready to perform a function.

INTERNAL SYNCHRONIZATION

Used when the I/O Module is the source of the stimulus. The clock edge used by the measurement device is generated internally by the program.

I/O (Input/Output)

Generally refers to external input/output devices (keyboard, modem, or display) and the interface (PIAs, UARTs, DUARTs, etc.) required to communicate with the microprocessor.

I/O MODULE

A part of the 9100A system that can both stimulate and measure digital circuitry with up to 40 separate lines.

KERNEL

The heart of a microprocessor-based system which includes the microprocessor bus, RAM, ROM.

MASK

A value where each logic 1 represents a bit that is to be acted on.

NODE

A set of component pins on a UUT that are connected together.

PIA (Parallel Peripheral Interface)

A popular I/O interface device that has three, 8 bit I/O registers, or ports that can be configured as input, output, or bi-directional.

POD SYNCHRONIZATION

Synchronizes the response gathering hardware with an internal pod signal called podsync. Podsync generation can be made to depend on valid address, data, or other (pod dependent) timing cycles.

RAM (Random Access Memory)

Semi-conductor memory that can be read or written to much faster than tape or disk drive memories. There are two basic forms of RAM; static and dynamic. Dynamic RAM requires a periodic refresh cycle and special circuitry to perform, but uses considerably less power and space than static RAM and generally comes in higher density packages.

RAS (Row Address Strobe)

A line used by dynamic RAM to latch the row address into the RAM device.

REGISTER

A temporary storage device for holding data.

RESPONSE

The measurement of a node characterized by the stimulus applied.

ROM (Read Only Memory)

Used mostly for storing programs not intended to be altered. Often referred to as firmware. Some ROMs can only be programmed once (PROMs), but due to the need to make software changes, reprogrammable ROMs (EPROMs, EEPROMs) are also available.

ROM SIGNATURE

A four digit hexadecimal number (CRC) which represents the data stored in a specified section of ROM. The signature is obtained by shifting all the binary bits in the specified ROM space through a special 16 bit register.

RUN UUT

A 9100A command that causes the microprocessor in the pod to fetch instructions from the UUT memory.

SERIAL DATA

Binary data transferred one bit at a time on a single line or channel.

SIGNATURE

See CRC and ROM Signature.

SOFTKEY

A key whose function is determined by software and is changeable.

STATUS LINE

Input lines to the microprocessor used for reporting UUT conditions.

STIMULUS

Signals generated by the interface pod, I/O module, probe or externally such as an operator, and is used to exercise a node in a predictable and repeatable manner.

STIMULUS ROUTINE

A sequence of commands that begins a measurement, provides a stimulus, then ends the measurement.

SYNCHRONOUS DATA

Concurrent with microprocessor bus timing cycles.

SUB-ROUTINE

A set of software instructions that may be used over again in a program.

TL/1 (Test Language One)

The programming language used by the 9100A to perform tests on a UUT.

TRI-STATE

Being in a high impedence state.

UUT (Unit Under Test)

The circuit board or system being tested by the 9100A.

USERDISK

In the 9100A, Userdisk is the highest level in the disk structure. Userdisk can be either the hard disk (HDR or /hdr) or the floppy disk drive (DR1 or /dr1).

VARIABLE

Provides the means for storing and changing data values in a program. For example, if x is the variable name, it's value would be stored and then changed as follows:

```
x = 5          ! assign the value of 5 to x

print x        ! print the value of x

x = x + 3      ! add 3 to the value of x (x = 8)

print x        ! print the new value of x
```

VECTOR OUTPUT I/O MODULE

A 9100A option that allows test vectors to be applied to a UUT at speeds up to 25 MHz.

VIRTUAL ADDRESS

An address that extends beyond 32 bits. Used when addressing special addresses in some pods.

WAIT STATE

This causes a delay in the microprocessor bus cycle to give slow devices time to be accessed, or a RAM refresh circuit time to complete it's refresh cycle.

WINDOW

An area of the programmer's display that is reserved for certain information. The programmer can make the window appear or go away by pressing the proper key, depending on whether the information is needed.