



COPY REQUEST FORM



Copy Services available from 7am to 4:30pm, Monday through Friday. Call extension 5389 or interoffice mail at mail stop 119.

document description	number of originals	number of copies
9100 Part II manual	165	5

HIGHLIGHTED options will default when form is not completely filled out

Paper Size/Type

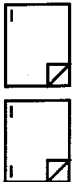
Paper Color

Finishing

- 8 1/2 x 11
- 8 1/2 x 14 (white)
- 11 x 17 (white)
- 3 - Hole Drilled
- Coverstock
- Tabs
- COLOR COPIES
- Letterhead
- Int. Ltrhd
- 2nd page Ltrhd
- Interoffice Memo
- Press Release
- Transparencies
- Labels

- WHITE**
- Recycled White
- Blue
- Buff
- Goldenrod
- Grey
- Ivory
- Pink
- Yellow
- Green - (for Company confidential only)

- COLLATED**
- Single Sided Copy
- DOUBLE SIDED COPY**
- STAPLE**
- Book Staple
- Tape Bind 15 - 125 total sheets only
- Slipsheet between sets



Special Instructions:



COPYRIGHTED MATERIAL (Release has been obtained in writing from an Officer of the Company holding copyright.)

* Not to be confused with Department of Defense Confidential — **DEPARTMENT OF DEFENSE CLASSIFIED MATTER** (Confidential, Secret, Top Secret) **WILL NOT BE REPRODUCED UNDER ANY CIRCUMSTANCE.**

AUTHORIZED SIGNATURE for Copyright release

Xerox Facilities Management Use Only

Total Black Impressions

Total Color Impressions

NAME Maui Barrett

TODAY'S DATE 3-12-96 DUE DATE 3-13-96

COST CTR / WORK ORDER #: 0406 EXT: 6330

Call When Ready

Interoffice Mail

MAIL STOP: _____

White - Copy Center Yellow - Customer upon completion Pink - Customer reference

9100-SERIES SOLUTIONS

9100-SERIES SYSTEMS TRAINING - PART 2

FLUKE®

P/N 870568

©1990, Fluke Corporation. All rights reserved. Printed in U.S.A.

Fluke Corporation

**P.O. Box 9090
Everett, WA 98206
(206) 347-6100 or
(800) 44-FLUKE**

Rev. June 1994

Table of Contents

SECTION 1

9100FT Part I Review

SECTION 2

UFI: Microprocessor Emulation

Section Page

Executing a Functional Test.....	3
Exercise 2-1	3
Troubleshooting From a Functional Test	5
Exercise 2-2	5
Creating a UFI Database.....	7
Creating a UFI Database.....	8
Part Description	9
Reference List	9
Compile to Learn Responses	9
Stimulus Routine.....	9
Stimulus Programs.....	9
Response File.....	10
Compile to Troubleshoot	10
Perform a Summary	10
Test.....	10
Building A UFI Database	11
Step 1: Part Descriptions.....	11
Exercise 2-3	11
Step 2: Reference Designator File	13

Exercise 2-4	13
Step 3: Compile to Learn Responses	15
Exercise 2-5	15
Step 4: Develop the Stimulus Routine	17
Data Stimulus Commands	17
Address Stimulus Commands	17
Step 5: Stimulus Programs	19
Stimulus Program Structure	19
Define	19
Setup	19
Stimulus Routine	21
Exercise 2-6	21
Step 6: Stimulus Response Files	23
Exercise 2-7	23
Merging Responses	27
Response File OFFSET Window	29
Upper Left Table	29
Upper Right Table	29
Waveforms	30
Legend	30
Softkeys	31
Changing the Calibration Delay	34
Adding Information to an Existing Response File	37
Step 7: Compile To Troubleshoot	39
Exercise 2-8	39
Step 8: Summary File	39
Exercise 2-9	39
Step 9: Testing	41
Exercise 2-10	41
Lab Exercise	43
Address Bus	43

Lab Exercise.....	45
ROM Address Decode Circuit.....	45
Lab Exercise.....	47
ROM 1 Data Bus.....	47
Lab Exercise.....	49
RAM ROW Addressing.....	49

SECTION 3

GFI: Microprocessor Emulation

*Section
Page*

Creating a GFI Database	3
Exercise 3-1	3

SECTION 4

Memory Emulation

*Section
Page*

Overview	3
Description of the 9132FT Pod.....	5
Installation of the Pod and Modules	9
Exercise 4-1	9
Master User Disk	13
Pod Self Test.....	15
Exercise 4-2.....	15
Sync Module Connection to the UUT	16
Reset Connection to UUT.....	23

Test Conditions that Cause a Reset	25
ROM Module Connection to UUT	27
Exercise 4-3	27
Pod Setup.....	29
Exercise 4-4	29
Exercise 4-5	41
SETUP Program	41
Calibration of the Probe.....	43
Exercise 4-6	43
Bus Test.....	45
B_TEST	45
Exercise 4-7	45
TEST_BUS.....	47
B_DIAG.....	49
Exercise 4-8	49
Exercise 4-9	51
Unbuffered Faults.....	51
Buffered Faults	51
Fault Isolation	53
STIM_DAT.....	53
STIM_ADR	55
QWK_RD	55
QWK_WR	57
Exercise 4-10	59
Testing the UUT RAM.....	63
RAM Fast Test	63
HyperRAM Test.....	63
RAM Full Test	63
Exercise 4-11	63
Exercise 4-12	65
Gathering Signatures.....	69

Exercise 4-13	69
Using the self test socket	69
Using the ROM Module plugged into the UUT	69
From other UUT ROMs	71
UUT Boot ROMs soldered to the UUT	71
ROM Test	71
Exercise 4-14	71
Exercise 4-15	73
Troubleshooting Hints	75
Problems Due to a Marginal UUT	77
GO_NOGO Program	79
Exercise 4-16	79

SECTION 5

UFI: Memory Emulation

*Section
Page*

The Software Switch	3
program bus_test	5
program rom0_data	7
program k_addr	8
program k_data_in	9

SECTION 6

Using Diagnostics

APPENDIX A

Glossary

APPENDIX B

Technical Data Sheets

APPENDIX C

Application Notes

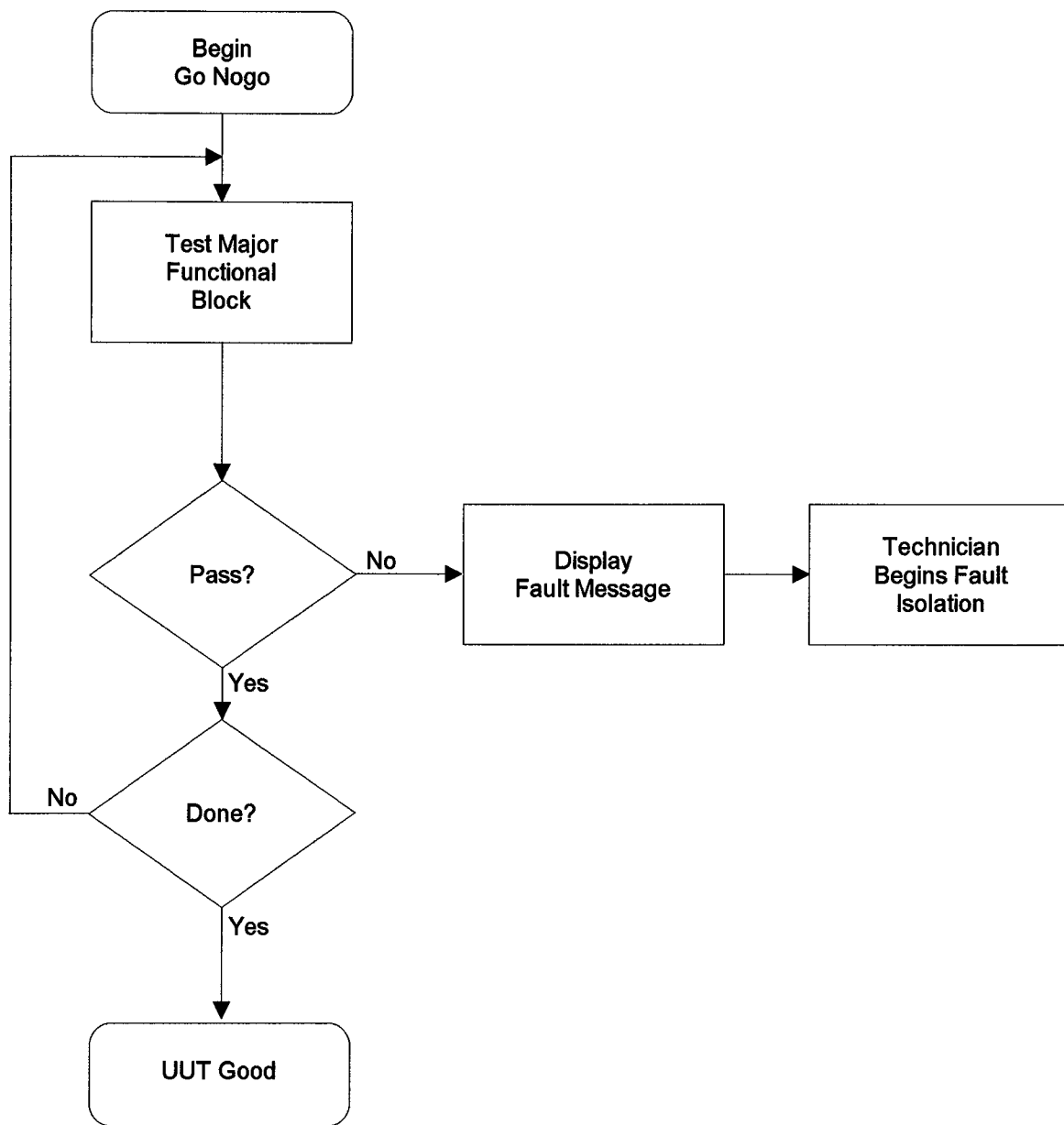
Section 1

9100FT Review

Section 2

UFI: Microprocessor Emulation

The exercises you perform in this section illustrate the capabilities and ease of use of the 9100FT 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.




EXECUTING A FUNCTIONAL TEST

The software you are about to execute follows the flow illustrated on the previous page. Exercise 1 performs a major functional test on the UUT. If the test passes the next major functional block is tested. If a major functional block fails, the error is displayed. The technician can then begin fault isolation. This method is commonly referred to as "go-nogo" testing.

EXERCISE 2-1


With no fault switches set on the Demo-Trainer UUT (Switch pack 1, switches 1 through 8 ON, all other switches OFF.)

press...  on the 9100FT Applications keypad.

The Mainframe displays the following menu:

EXEC UUT PROGRAM


press...  to select the UUT name CD.

press... 

press...  to select program GO_NOGO.

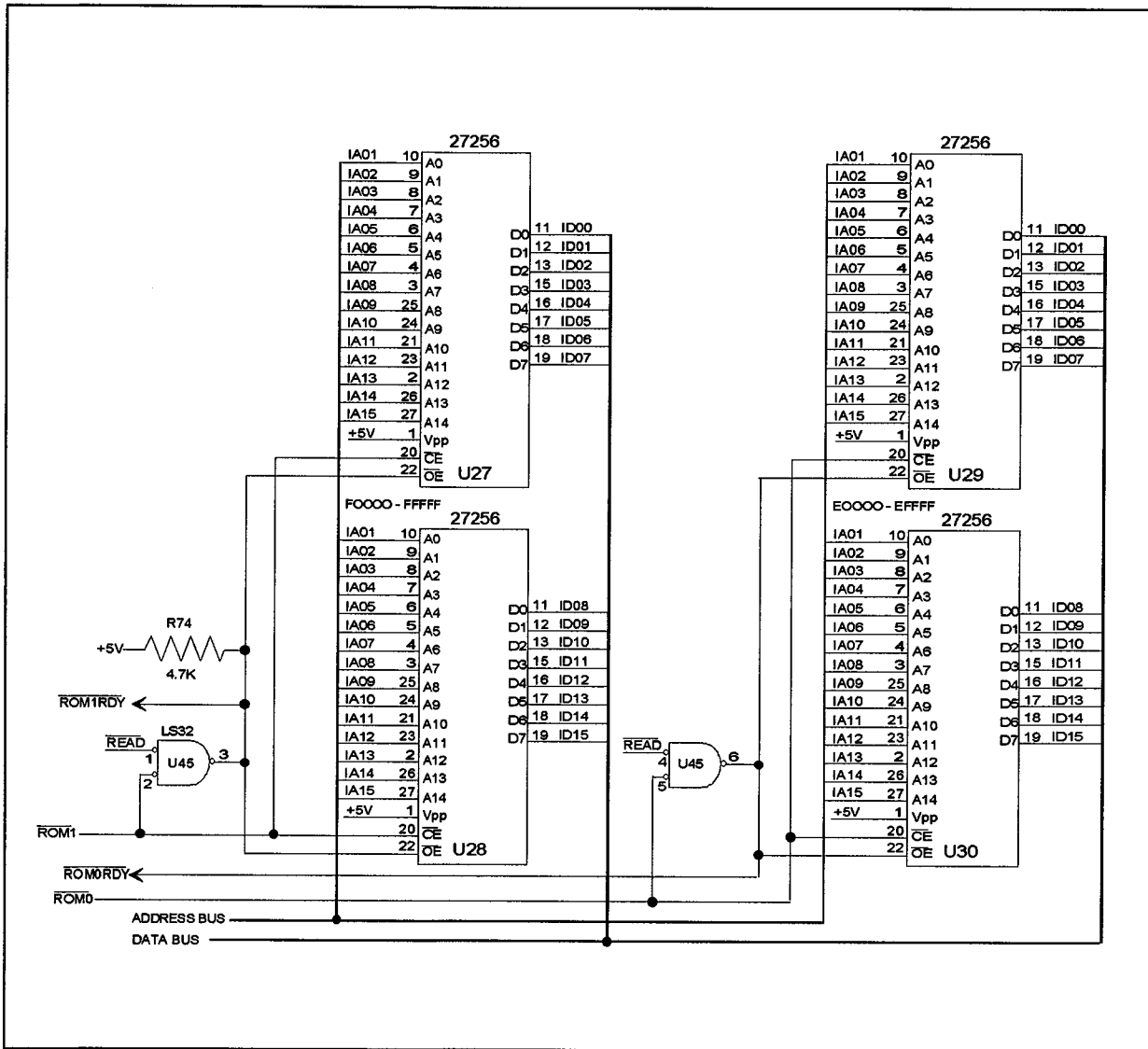
The following should now be displayed on the Mainframe:

EXEC UUT CD PROGRAM GO_NOGO

press...  on the 9100FT Applications keypad to begin the program execution.

During execution of the program, a go-nogo test is 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!)



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 exercise 1.

EXERCISE 2-2

During program execution, a fault is detected while testing the U27 functional block. The following error message appears on the Mainframe display:

Testing from F0000 to FFFFE
all ROM data bits stuck high

press... on the 9100FT Applications keypad to continue the test. 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 let's start at the source of the problem by performing tests at the ROM device itself.

All of the UFI programs and responses are done for us. So begin by performing a UFI test of control lines at U27.

press... on the Applications keypad. The following prompt should be displayed:

RUN GFI UUT CD REF PIN

Move the cursor to the REF field, Type-U27.

Move the cursor to the PIN field, Type-22.

The display should read:

RUN GFI UUT CD REF U27 PIN 22

press... *What were the results?*

Analyzing the result... what would be the next logical step?

Find the fault by backtracking.

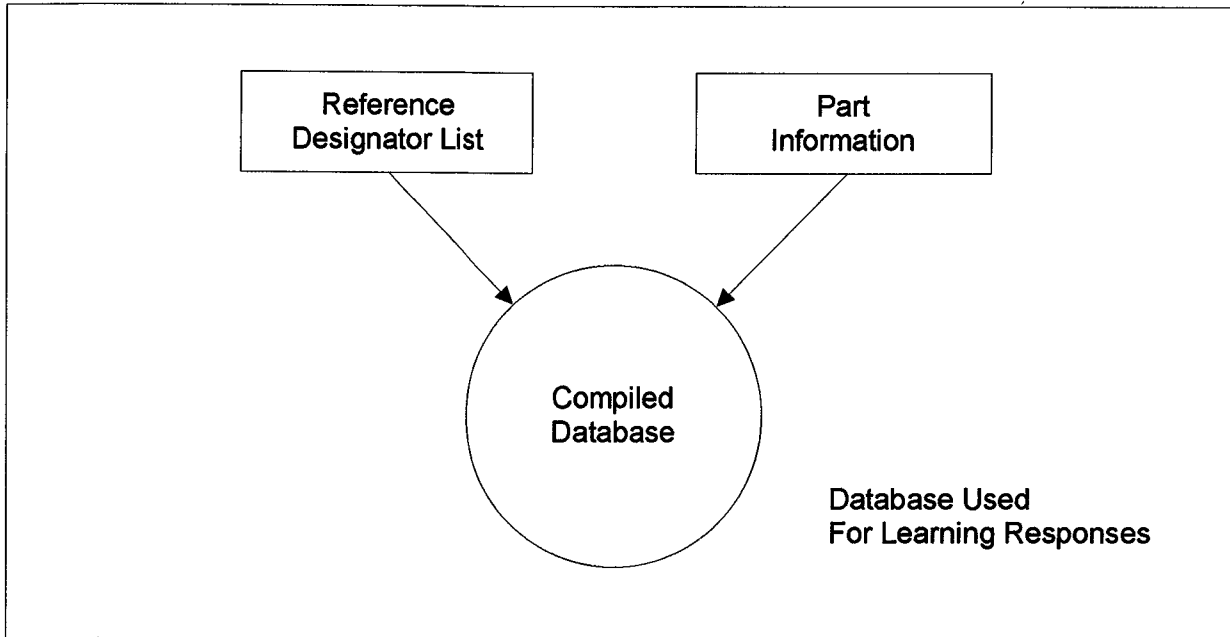


Figure 1.

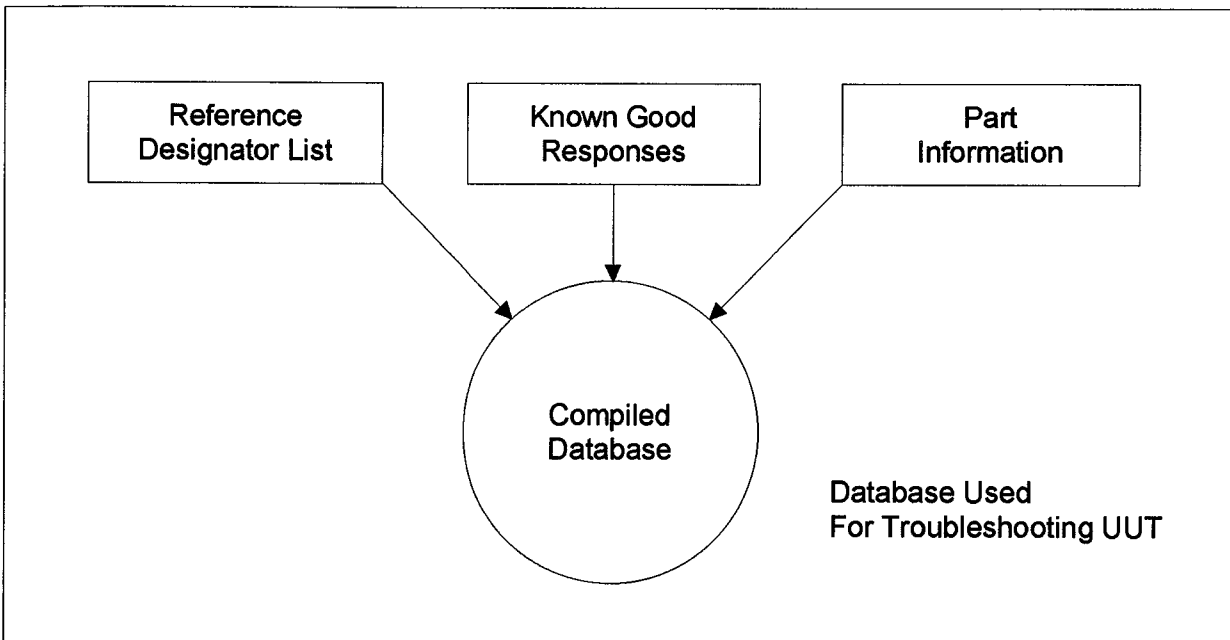


Figure 2.

CREATING A UFI DATABASE

UFI is designed so that an experienced technician can use the 9100FT'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 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 shows the structure of the UFI database used for troubleshooting. It contains the following types of information:

- ✓ **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**, the **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.

Stimulus Routine

The Stimulus routine provides the starting point for the stimulus, the stimulus itself, and the ending point of the stimulus.

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.

Response File

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

Compile to Troubleshoot

Stimulus programs, stimulus responses, 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 page 2-8 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.

Perform a Summary

Optional for UFI

Test

A test should be performed with and without a fault inserted into the UUT. Doing so will ensure the test is finding a fault only when there is one.

BUILDING A UFI DATABASE

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.

EXERCISE 2-3

Edit the following file:

Edit: /hdr/partlib Type: LIBRARY

A directory of part descriptions already in the parts library (PARTLIB) are shown. Because the pin configuration is the same, the types are generic in that a 74245 is used for 74LS245, 74ALS245, and 74HCT245, etc. Check the directory for all parts that you have listed. *NOTE: The parts are shown in hierarchical order.*

Are the parts in the library?

To illustrate how to create a part description, edit a part called *your name*.

Edit: Your name Type: PART

How is the pin configuration selected?

How is the function of each pin selected?

To QUIT editing the part:

press...

When asked if you wish to save your changes, select NO.

Where did you return to?

This page intentionally left blank.

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.

EXERCISE 2-4

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?

To QUIT editing the REFLIST:

press...

Where did you return to?

UUT COMPILER (UFI)

Processing UUT refilest 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, the (Compile) key is used to perform TL/1, GFI, or UFI compilations.

EXERCISE 2-5

press... The following prompt will appear:

COMPILER TYPE: TL/1

press... to select UFI.

press...

You are prompted for the database type...

press... to select LEARN RESPONSES.

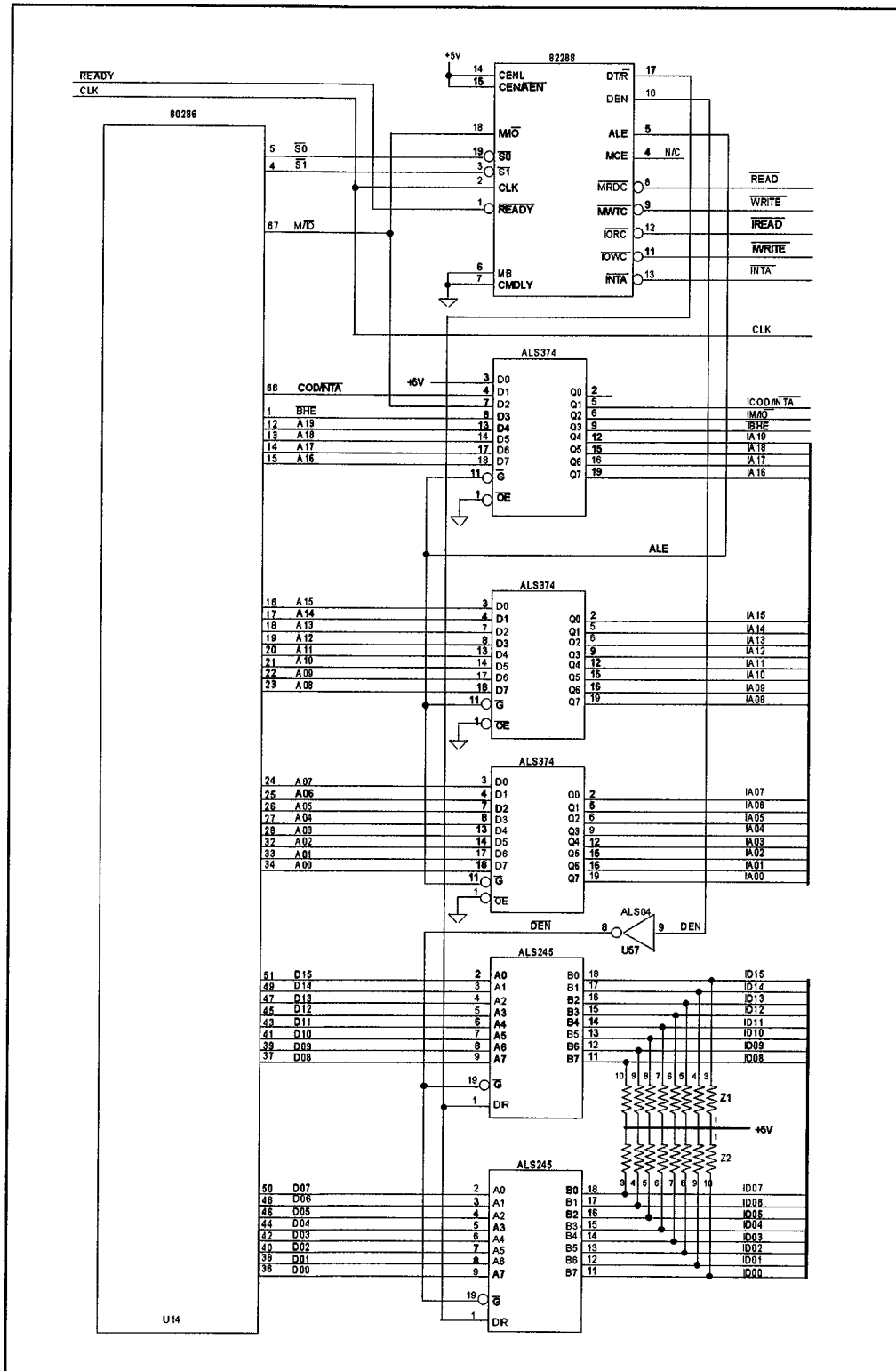
press... to begin compiling the database.

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

press... 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.



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 when 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 may be performed at any time prior to step 6.

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

ARM

SHOW

```
Program micro_data
```

```
    if (gfi control) = "yes" then
```

```
        devname = gfi device
```

```
    else
```

```
        devname = "/probe"
```

```
    endif
```

```
end program
```

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 than 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 were all the same for each RAM output, the test may still pass.

IMPORTANT:

A stimulus program must not simultaneously exercise multiple sources on a net. Each source capable of driving any given net must have its 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.

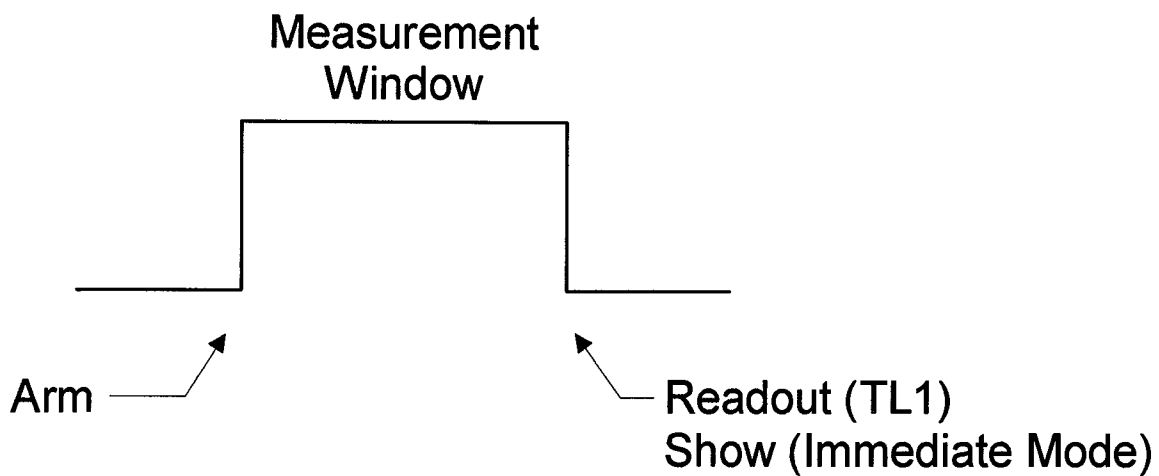
- ✓ **Define**
- ✓ **Setup**
- ✓ **Stimulus routine**

DEFINE

The **Define** portion of a stimulus program describes the measurement device to be used. Note the program 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*.

SETUP

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



STIMULUS ROUTINE

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 and repeatable response.

A very important part of writing a stimulus program is naming it. Especially when 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 determining where to test next. The following stimulus program is named **micro_data**. The name implies that the stimulus exercises data lines from the microprocessor.

EXERCISE 2-6

Write the stimulus program:

Edit: /hdr/trainer/micro-data Type: PROGRAM

Enter the routine that determines if the program has been called by GFI/UFI. *If so...* 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 (CHECK) and selecting the current TL/1 options.

After any check errors are corrected, "DEBUG" the program, using the debugger.

press... (SAVE) to save the program.

press... to quit the program.

At UUT level, perform a TL/1 compile using the current TL/1 options.

HDR/TRAINER/MICRO_DATA (RESPONSE) Line 5

----- Response Data -----
Node Learned Async CLK Counter
Signal Src With SIG LVL LVL Mode Counter Range

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
GOTO	SAVE	LEARN	OFFSET	SELECT	DELETE	INSERT	MORE		FAULT

Screen 2.

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.

EXERCISE 2-7


Edit the stimulus response file:

Edit: /hdr/trainer/micro_data Type: RESPONSE

The display should resemble Screen 2, shown on the opposite page.

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

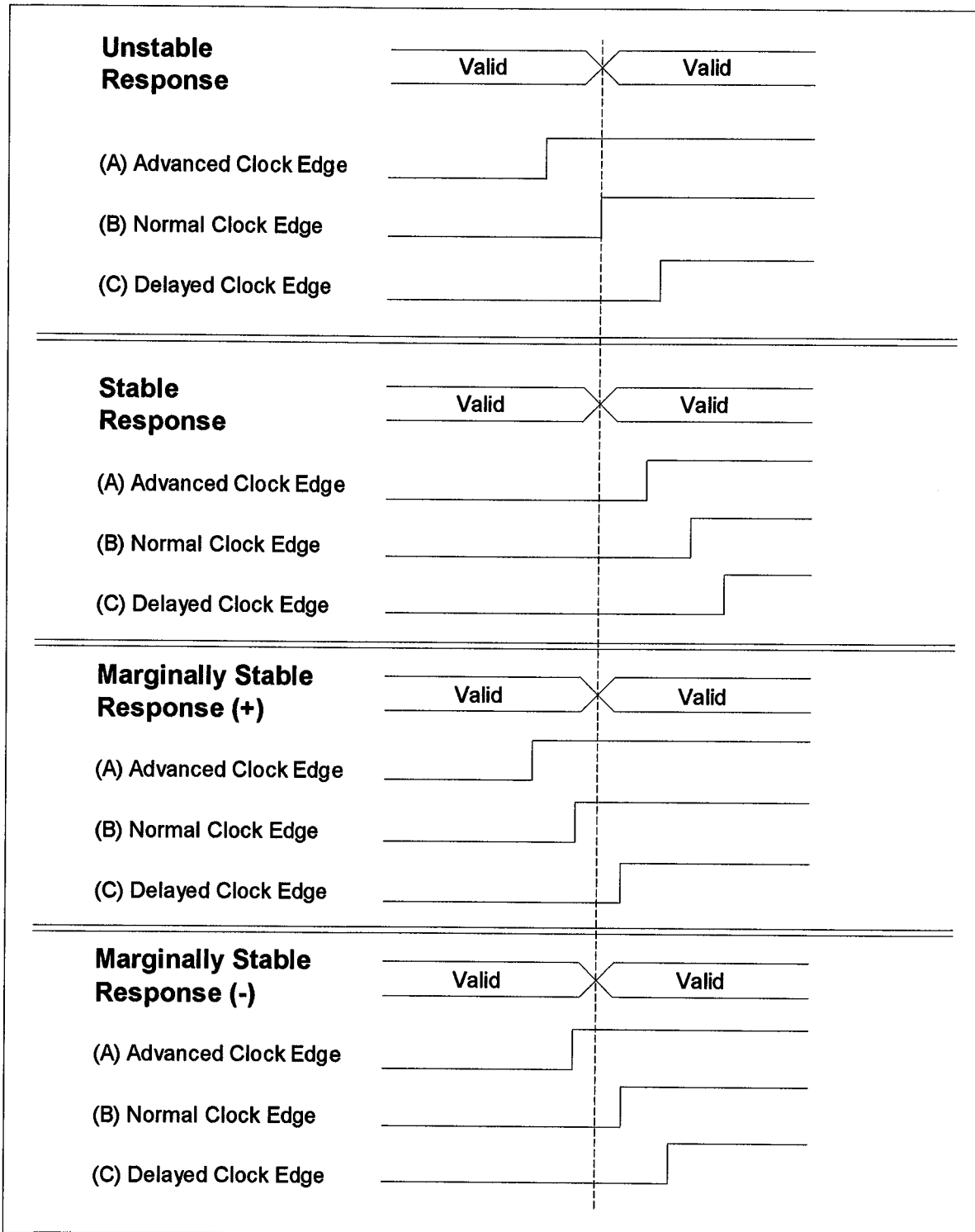
U3-11
U3-12
U3-13
U3-14
U3-15
U3-16
U3-17
U3-18

NOTE: Use  to move to 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 (TRANS or FREQ) upon completion of the Learn. *This column cannot be edited.*



press... (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... again to return to the response screen.

press... to place the cursor at the first node U3-11.

press... (LEARN) on the programmer's keyboard.

When prompted...

USE CURRENT LEARN OPTIONS YES

select... NO using the key.

press...

- **Learn Using:** Indicates whether 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 measurement devices if required.

press... key to select UFI.

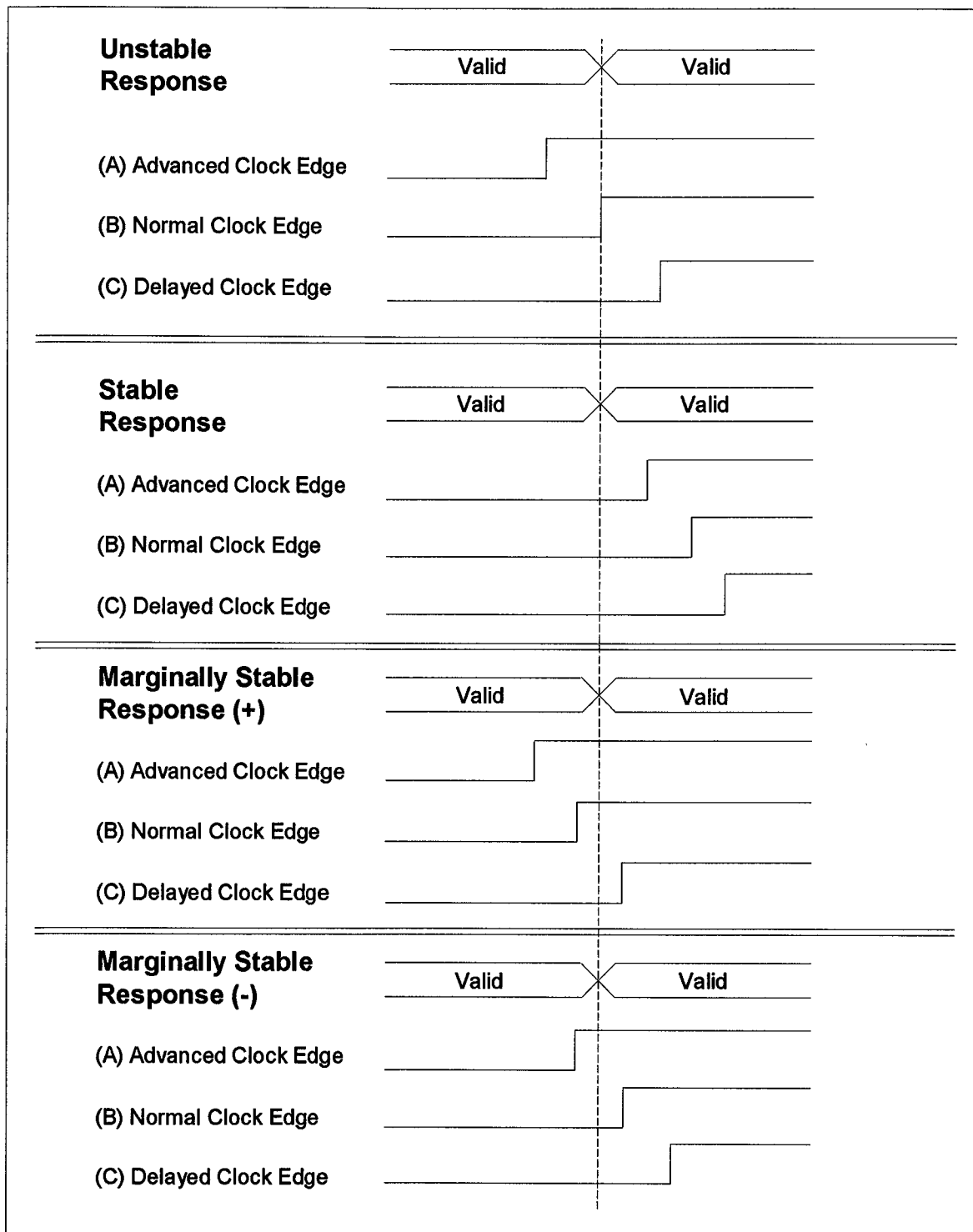
- **Learn for:** Indicates the number of pins covered by LEARN. The following three options may be selected using the key:

ONE NODE The line of the edit window where the cursor is located 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.

press... key to select ONE NODE.



- **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 is repeated. A number between 1 and 99 may be entered. Multiple executions of the LEARN utility are used to tell if the responses are:

- ✓ Stable
- ✓ Unstable
- ✓ Marginally Stable

type... 1 .

press... (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. *NOTE: Leave one space between U3 and U23 entries.*

press... (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.
- ✓ An * indicates all three responses were different.

/HDR/TRAINER/MICRO_DATA (RESPONSE)
Line 6

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

Node	Learned	Async	Clk	Counter	Counter Range
Signal Src	With	SIG	LVL	LVL	Mode
U3-11					

OFFSET WINDOW

Offset: SIG: Async LVL: Clk LVL: Count: Clk LVL SIG LEGEND: <div style="display: flex; align-items: center; gap: 5px;"> <div style="width: 10px; height: 10px; background-color: #cccccc; border: 1px solid black;"></div> means the Clk LVL was 1X, XD, or 1XD </div> <div style="display: flex; align-items: center; gap: 5px;"> <div style="width: 10px; height: 10px; border: 1px solid black; margin-top: 2px;"></div> means the SIG or Clk LVL for one offset changed while looping </div>	Offset Range: Sample Resolution: Number of loops through Range: Node Signal Src: U3-11
--	---

F1

F2

F3

F4

F5

F6

F7

F8

F9

F10

OFFSET
EXEC
LOOP
FAULT

Screen 3

Response File OFFSET Window

press... (OFFSET) to enable the Offset feature.

Notice the Offset Window (Screen 3) 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 times the offset range has been swept.
Node Signal Src	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 that the sampled signature is different than 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 occurred at the particular sample.

The * means the SIG or Clk LVL for one offset changed while looping.

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... (OFFSET)

press... 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... 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)**Line 5**

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

Node	Learned	Async	Clk	Counter	
Signal Src	With	SIG	LVL	LVL	Mode
U3-11					Counter Range

OFFSET WINDOW

Offset: 1000056	Offset Range: 999876 TO 1000056
SIG: 00ED	Sample Resolution: 1
Async LVL: 1X0	Number of loops through Range: 1
Clk LVL: 1	Node Signal Src: U3-11
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 4

press... 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... The following prompt line appears:

SAMPLE RESOLUTION _____

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... A prompt to probe a pin or clip a device appears.

Follow the instructions given on the screen.

When the return key is pressed, the status message EXECUTING... appears at the bottom of the screen and sampling begins (example shown in screen 4).

DO NOT remove the measurement device until the entire offset range has been sampled.

When execution is done, the status message EXECUTING at the bottom of the screen will be replaced. The new status message... COMPLETE ... will appear.

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 ns for sync to pod data, do the following:

```
sync device "/probe", mode "pod"  
sync device "pod", mode "data"  
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 3 to 4 ns taps for the probe.

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 Clk LVL column.

press... (SELECT)

What happened?

press... SHIFT key with (SELECT) several times.

What happened?

Only intensified responses are saved. Select the signature and clocked level columns.

QUIT and SAVE all response information.

press...

The 9100FT will prompt you with:

QUIT TO SAVE CHANGES ? YES

YES should be the default. If it is not...

press... to select YES

press...

NOTES:

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 When complete, Quit and Save, then 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 to the response file.

UUT COMPILER (UFI)

Processing UUT refilest and parts

0 errors

Writing string table...

Writing part table...

Writing ref 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.

EXERCISE 2-8

At the UUT directory level:

press... (COMPILE)

select... UFI

press... (RETURN)

select... TROUBLESHOOT UUT

press... (RETURN)

Display should resemble Screen 6 on facing page.

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.

EXERCISE 2-9

You must be at the UUT directory level to create a summary file.

press... (SUMMARY)

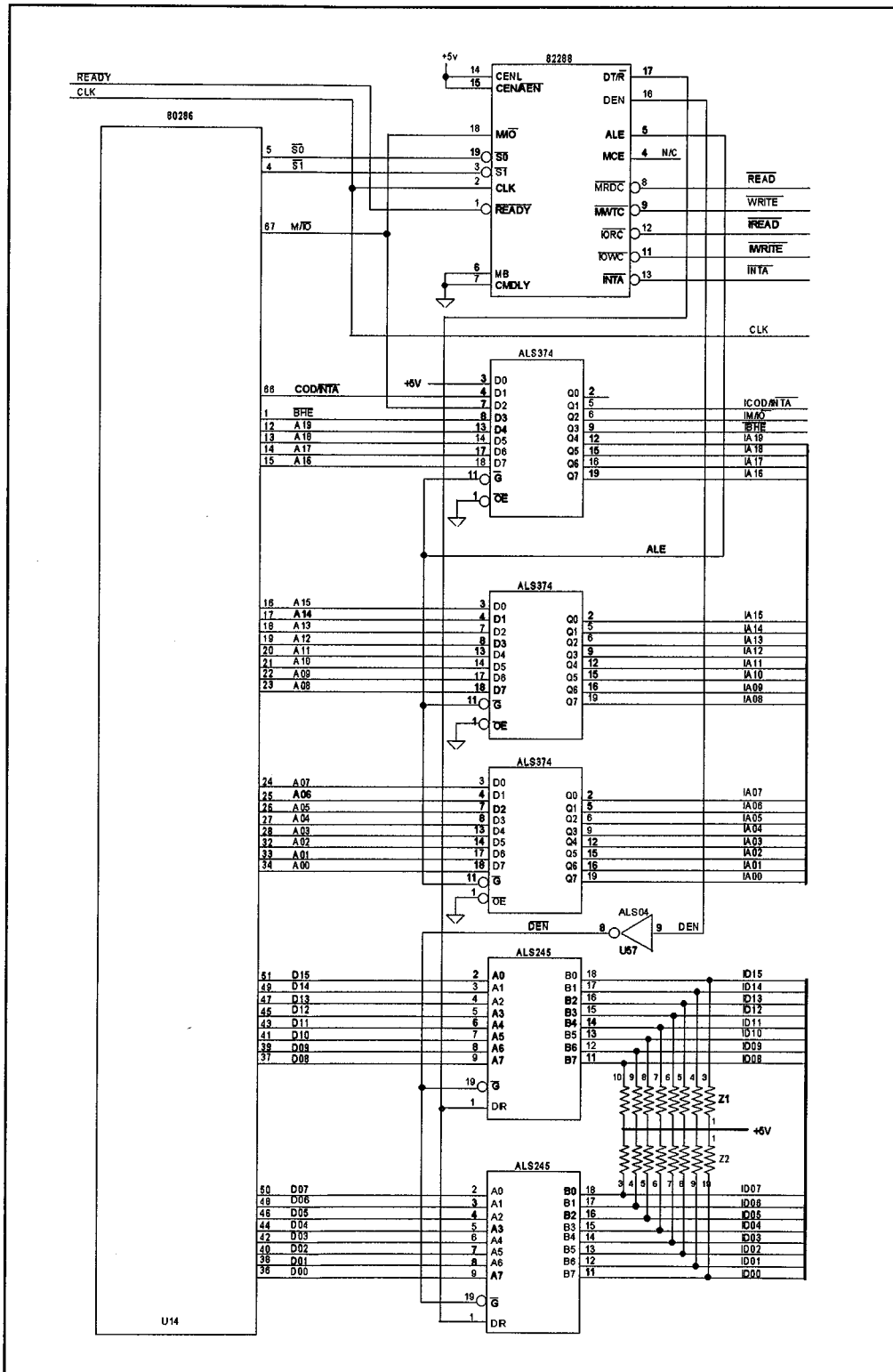
The following prompt will appear at the bottom of the screen:

Generate GFI Summary to TEXT file _____

A 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 9100FT display.

Now review the summary file by editing the text file. If you choose not to generate a text file, you may press with (SUMMARY). The summary will only be displayed, not written to disk.



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.

EXERCISE 2-10

Leave the editor by pressing along with .

press... ^L *The following prompt will appear:*

RUN GFI UUT CD REF PIN

enter... **TRAINER** (using key) in the UUT field.

enter... **U3** in the REF field.

enter... **16** in the PIN field.

press...

The following prompt should be displayed:

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

probe... U3 pin 16.

press... ^H

You should see the following response:

**UNGUIDED MODE
PIN GOOD AS OUTPUT**

set... SW4-2 fault switch to the fault position.

test... U3 pin 16 again.

What happened?

LAB EXERCISE

Address Bus

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

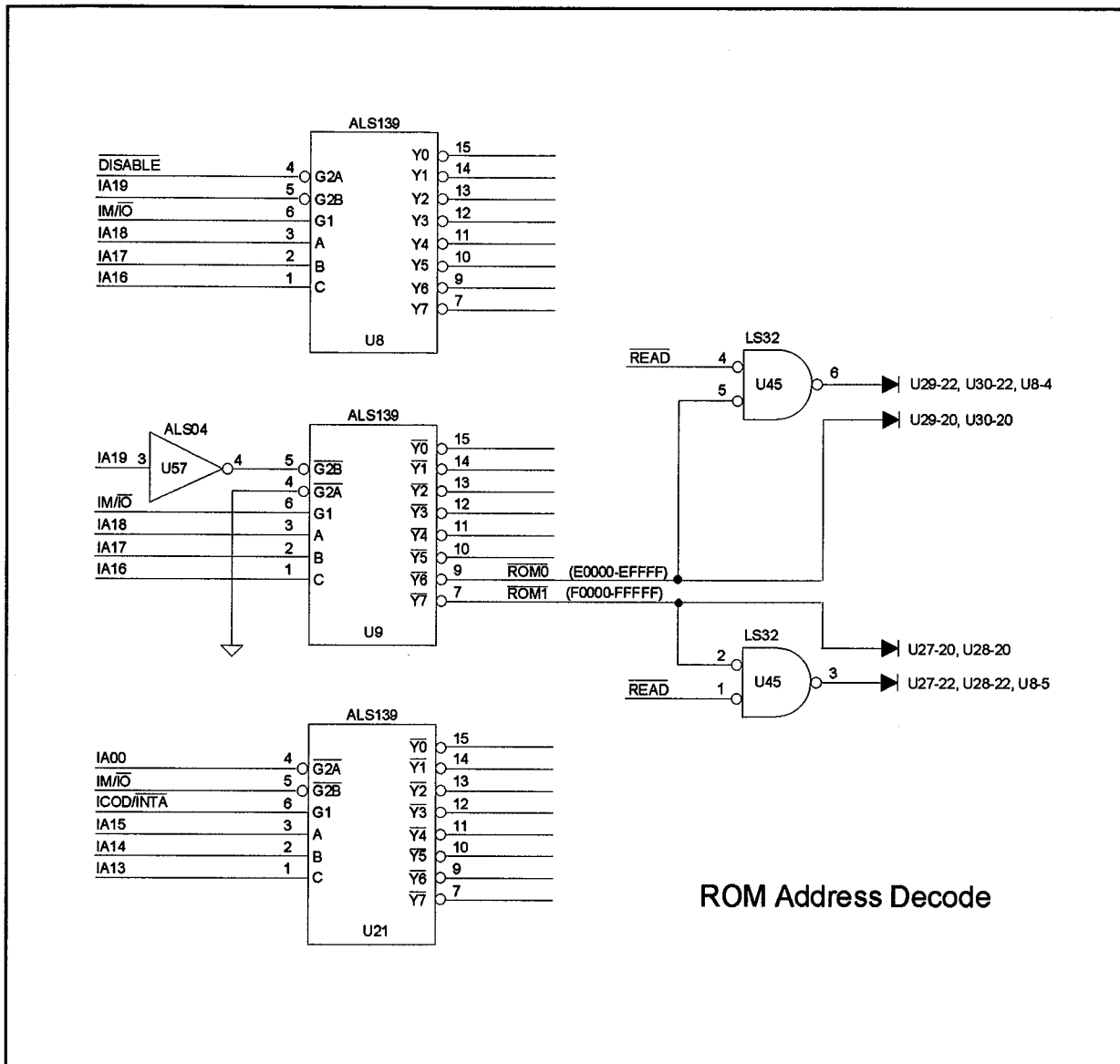
When complete:

perform... GFI test on U2-2 in the immediate mode.

confirm... That the pin is good.

set... Switch 1-1 open.

test... U2-2 again.



LAB EXERCISE

ROM Address Decode Circuit

perform... All steps necessary to test:

ROM address decoder outputs U9 pins 7 and 9.

ROM1 select U45 pins 2 and 3.

ROM0 select U45 pins 5 and 6.

ROMS U27, U28, U29 ,U30 decode inputs.

add... The decoder address inputs to the "Address Stimulus Response file".

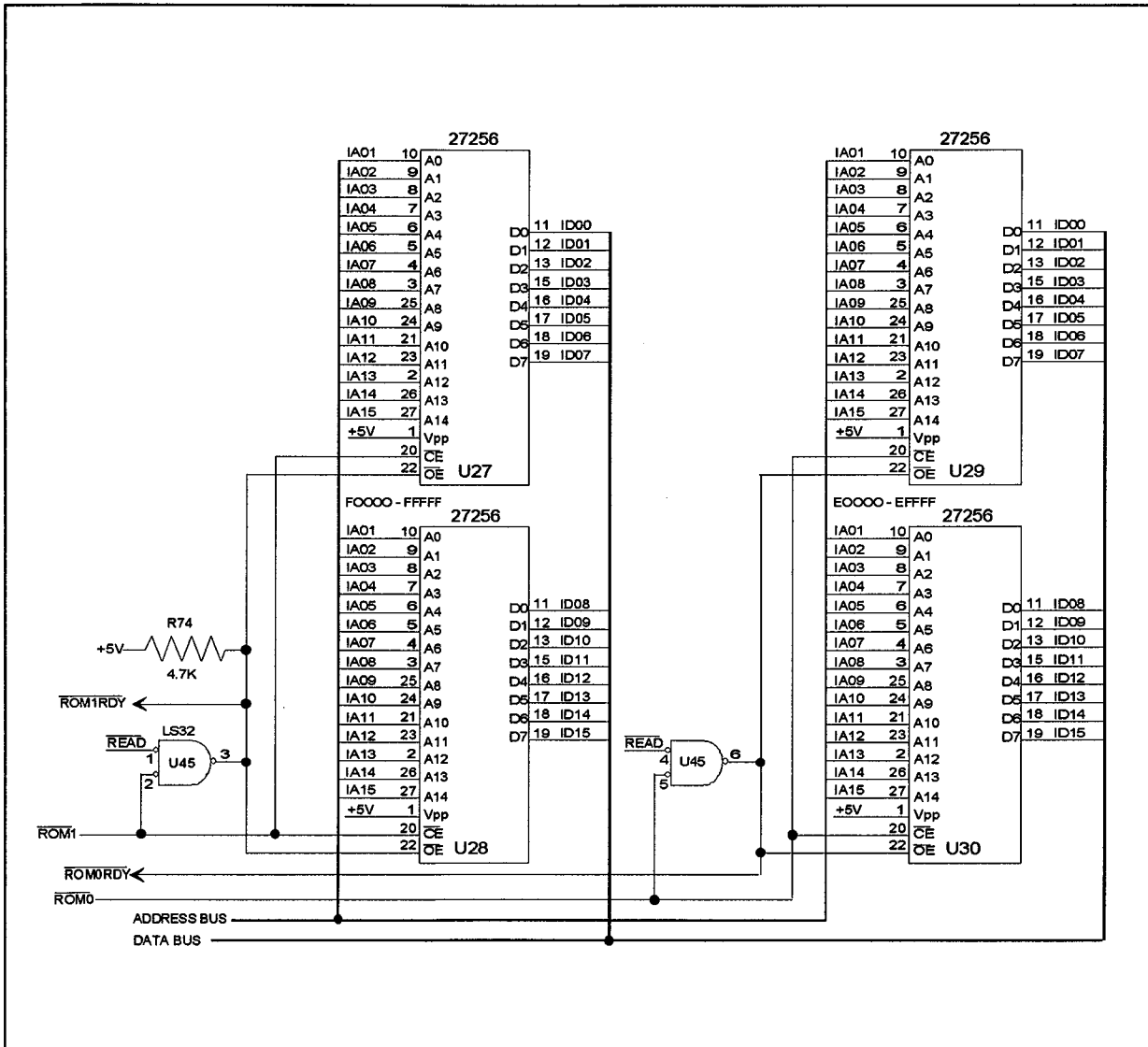
When complete:

perform... GFI test on U27-22 in immediate mode.

confirm... That the pin is good.

set... Fault switch 1-2.

test... U27-22 again.



LAB EXERCISE

ROM 1 Data Bus

perform... All steps necessary to test the data bus
from ROM1.

add... ROM1 address and data inputs to the
appropriate response files.

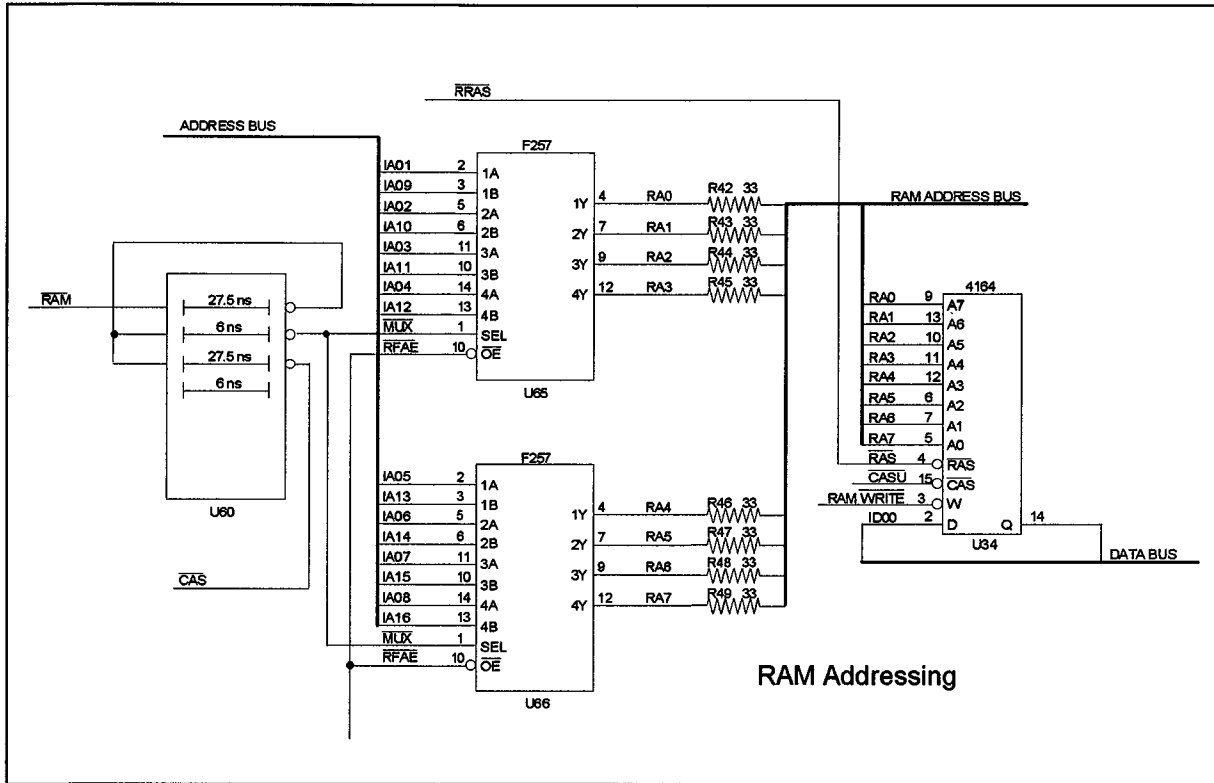
When complete:

perform... GFI test on U27-17 in immediate mode.

confirm... That the pin is good.

set... Fault switch 4-2.

test... U27-17 again.



RAM Addressing

LAB EXERCISE

RAM ROW Addressing

perform... All steps necessary to test the RAM address bus (RA0 - RA7) from the outputs of the RAM address multiplexers (U65, U66), to the inputs of U34 during ROW address time only.

add... The multiplexer address inputs to the appropriate response file.

When complete:

perform... GFI test on U34-11 in immediate mode.

confirm... That the pin is good.

set... Fault switch 4-8.

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

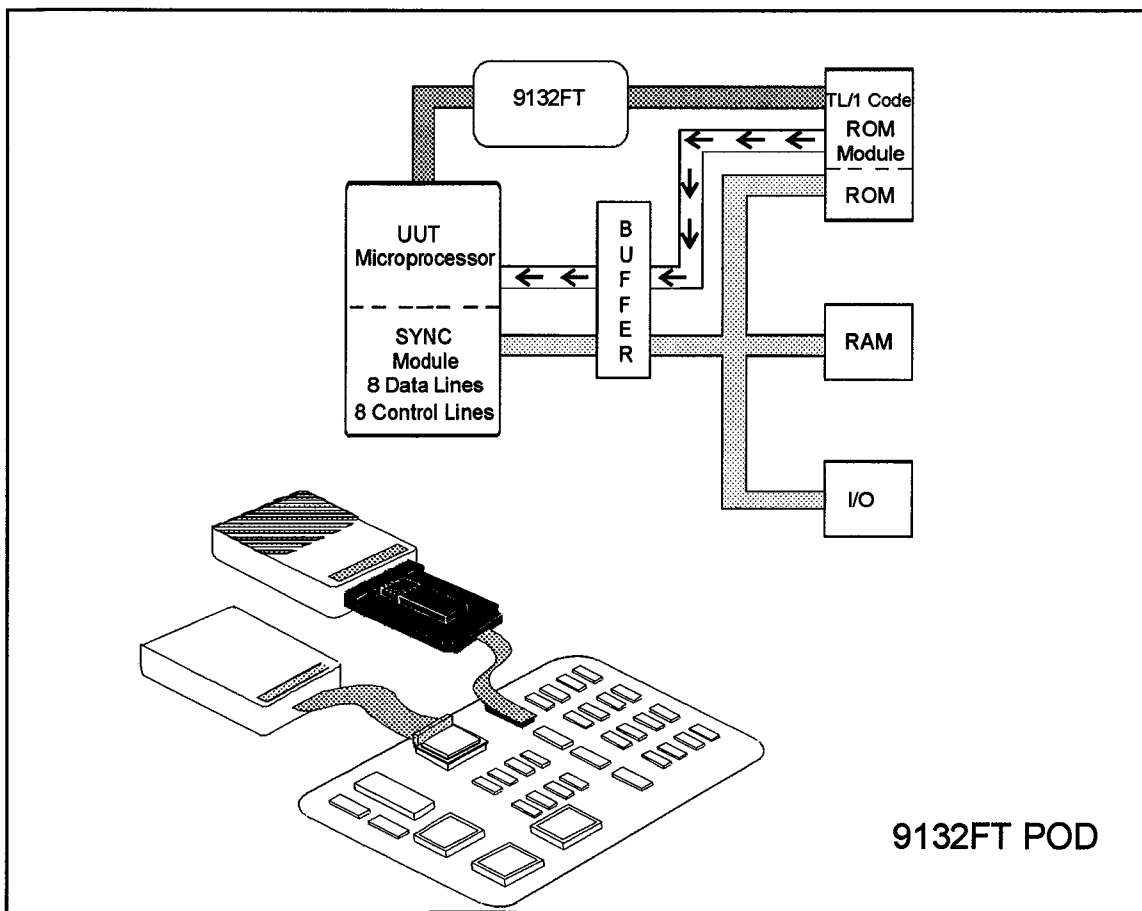
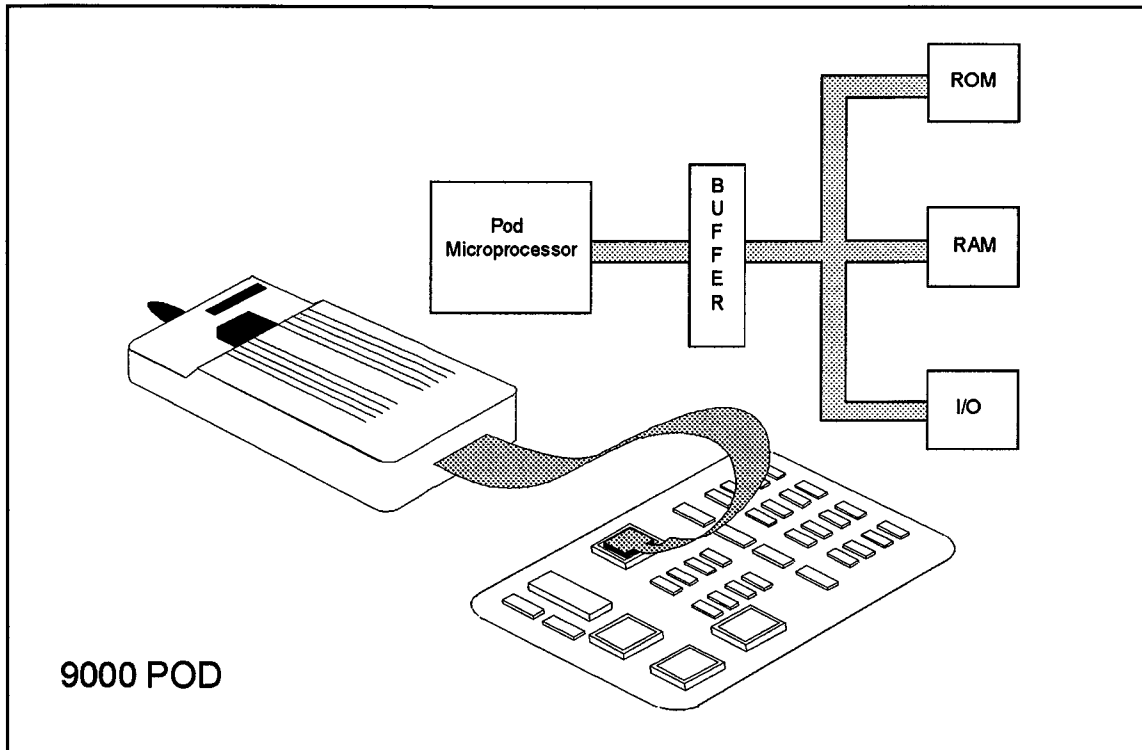
CREATING A GFI DATABASE

EXERCISE 3-1

- ① Verify NODELIST using TRAINERB (UUT).
- ② Verify 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

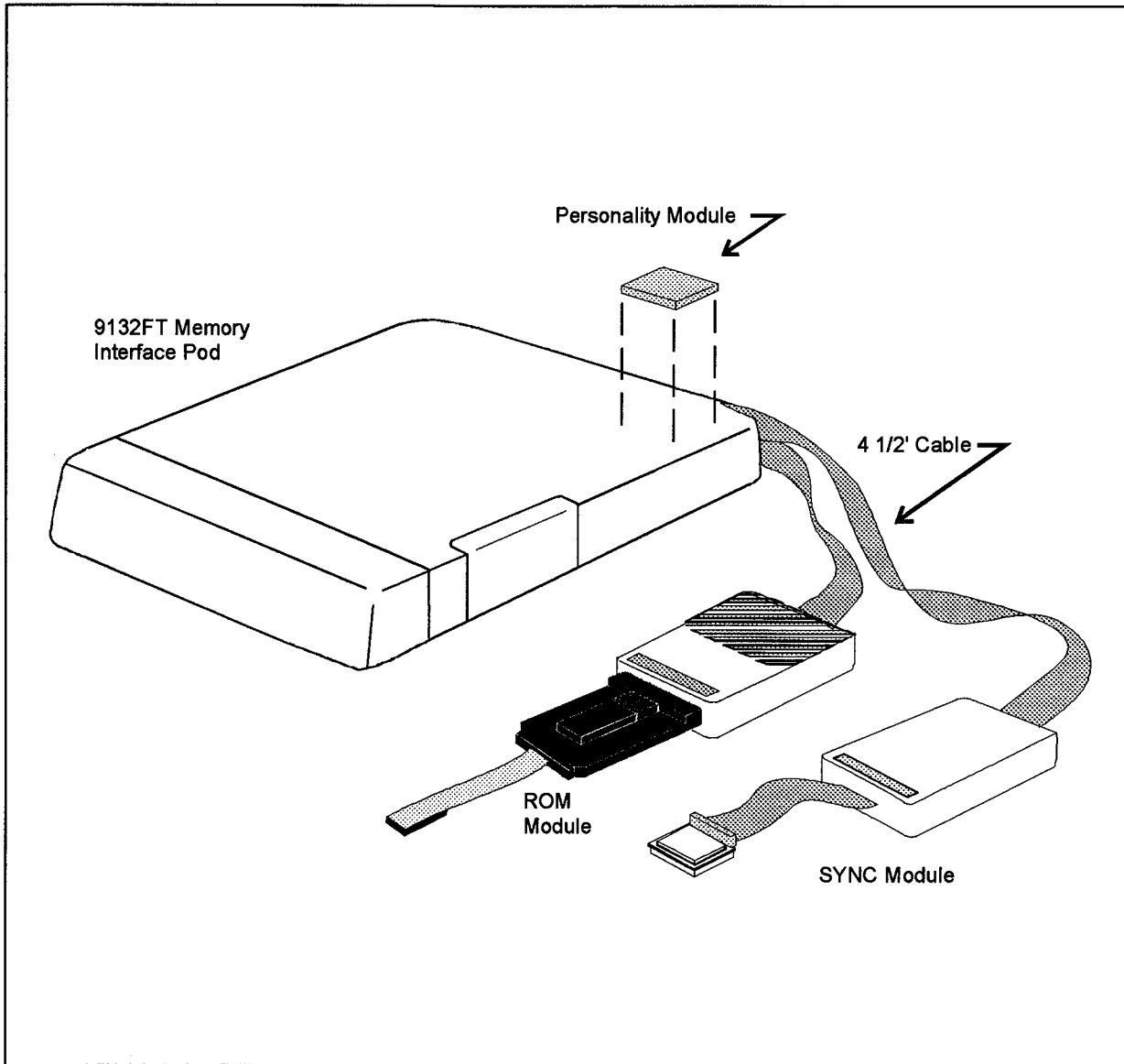


OVERVIEW

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

The Memory Emulation Pod allows the 9100FT to gain access to the UUT by replacing the Boot Memory with ROM Modules and connecting seventeen (17) lines of logic analysis onto the μ P. Sixteen (16) of the lines are used for logic analysis while one (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 μ P to the reset address (typically the boot address). Once this has been done, all of the μ P overhead is handled by the 9132FT passing code to the μ P through the ROM modules inserted into the boot memory.

In the case of a kernel fault which will not allow the μ P to run due to bad address, data, or control line, the 9132FT will reset the UUT and feed patterns to the μ P. This allows isolation of faults through a diagnostic algorithm within the 9100FT.

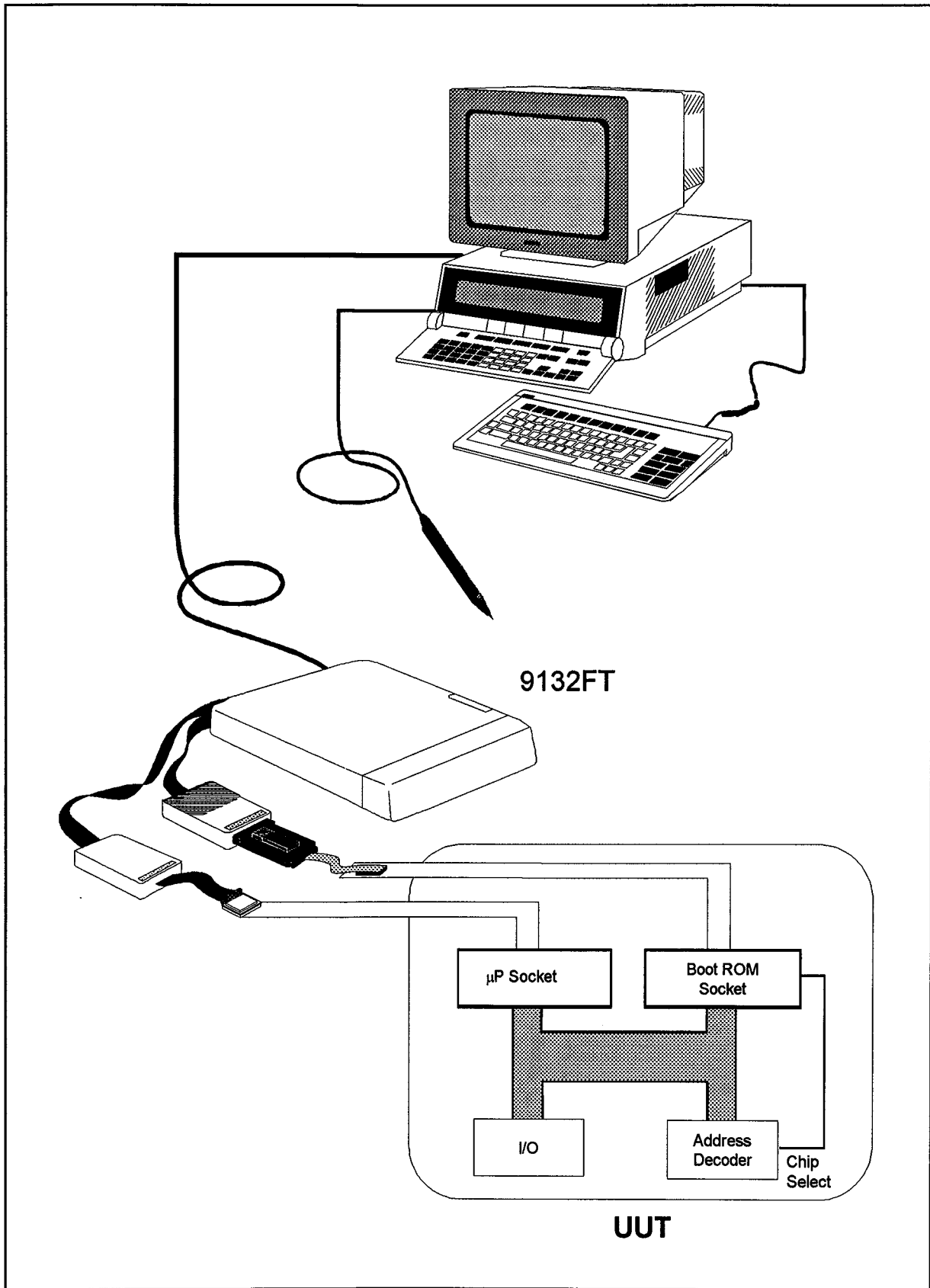


DESCRIPTION OF THE 9132FT POD

The 9132FT 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 module adapters. 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.



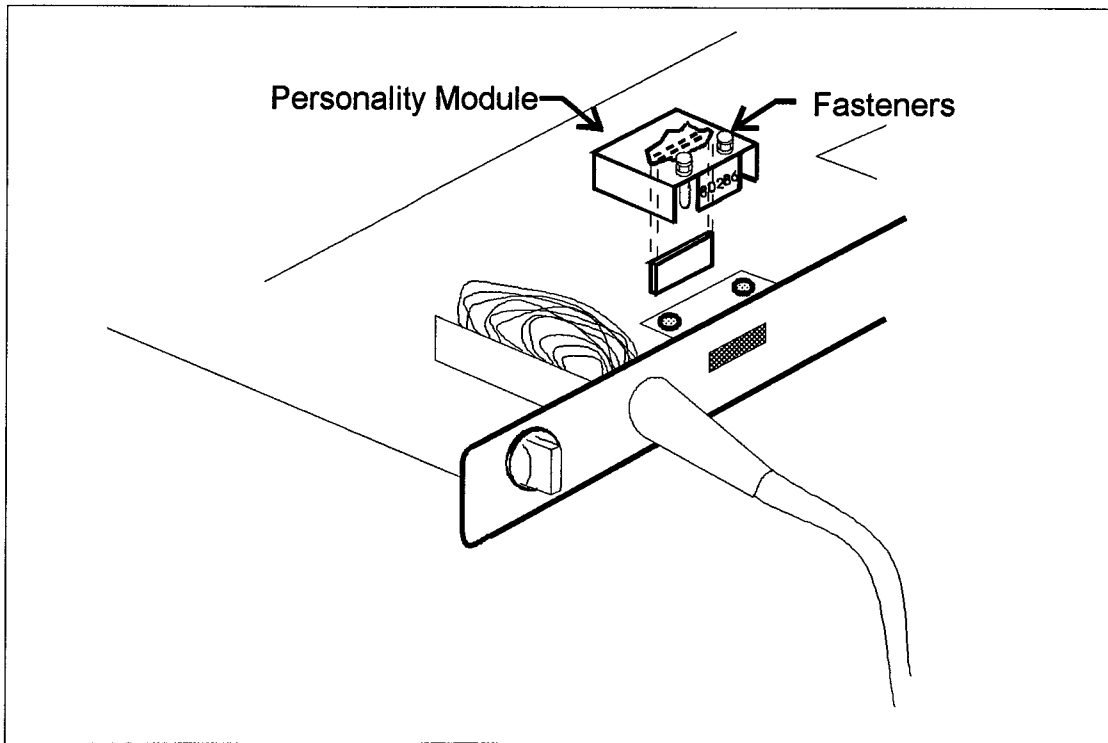
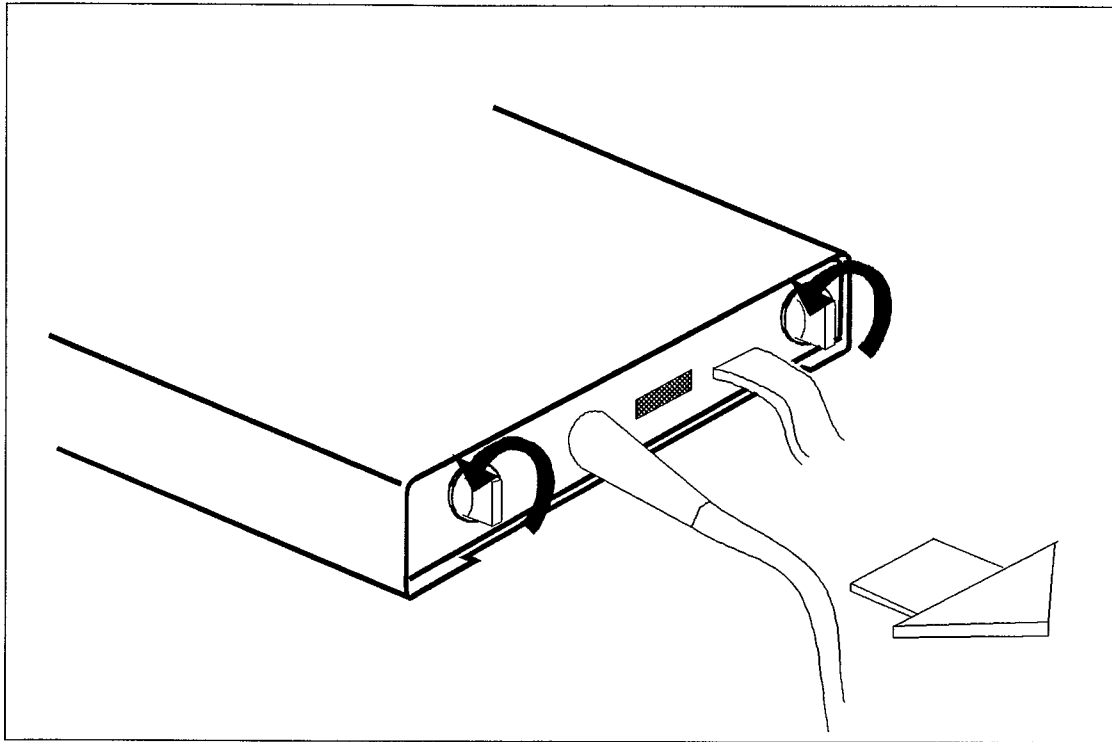
Description of the 9132FT Pod Cont...

The Mainframe supplies operating power for the 9132FT 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.

Over voltage protection circuits, or fuses on each line to the UUT, guard against damage to the Pod. The over voltage 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.

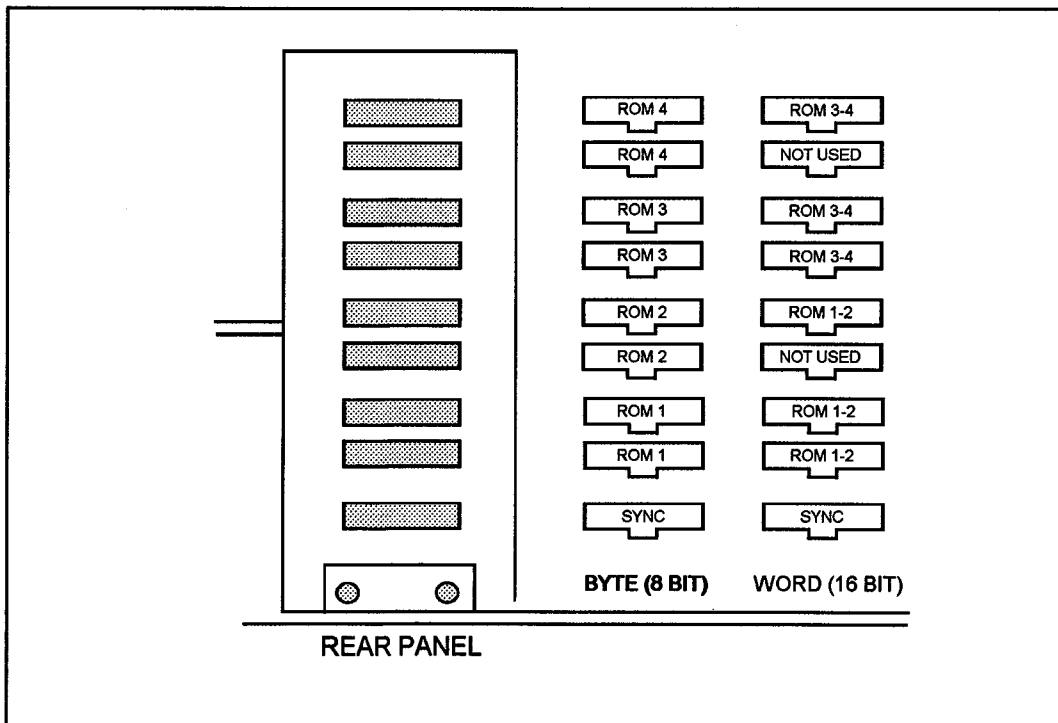
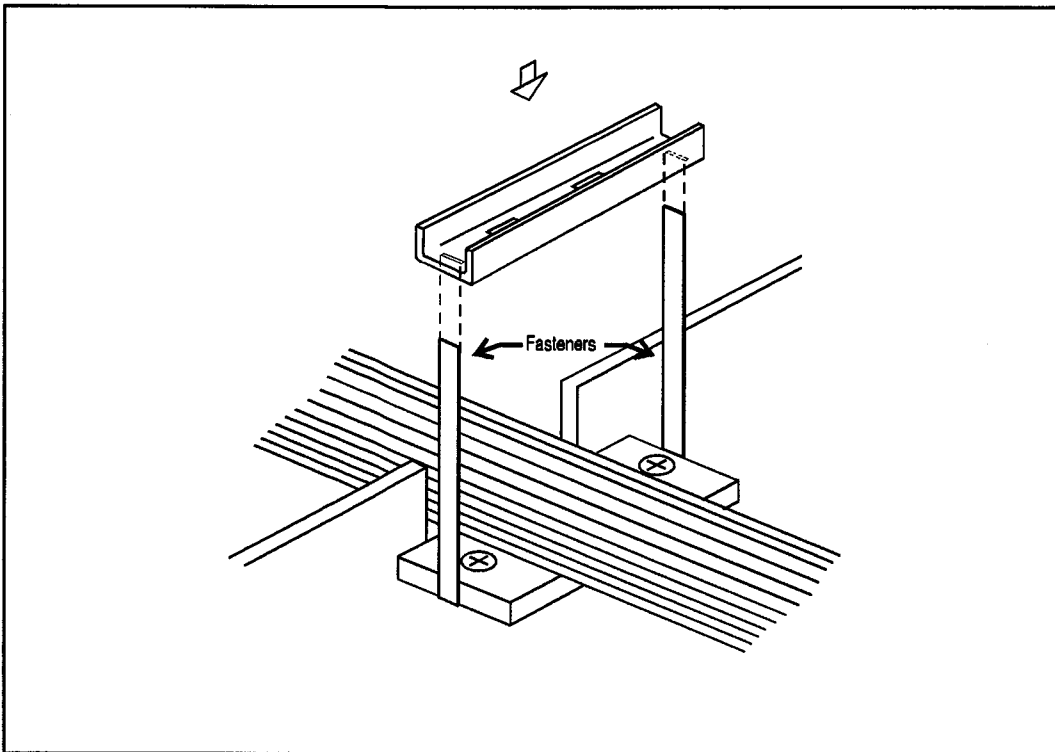
A self-test socket on the Pod enables the mainframe to check Pod operation. The ROM Module flex cable and the Sync Module adapter cable are connected to the self test socket during self test operation. This allows the mainframe to investigate the Pod's internal functions.



INSTALLATION OF THE POD AND MODULES

EXERCISE 4-1

- 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 pulling the panel out and away from the case.
- ③ To configure the Pod for the 80286 processor, an **80286FT Personality Module** must be installed in the Pod.
- ④ Verify that the correct Personality Module is installed.



Exercise 4-1, Installation of the 9132FT Pod and Modules Cont...

- ⑤ Install the sync and ROM Modules next. Plug each module into their corresponding sockets.
- ⑥ Close back of pod panel.
- ⑦ Connect the Pod to the Mainframe.
- ⑧ Turn power to the Mainframe ON.


MAIN: COPY DISK FROM DR1 TO HDR


Master User Disk

The 80286FT database for the 9100-SERIES Mainframe is contained on one 3.5 inch floppy diskette supplied with the 9132FT-80286.


Installing the database

To install the database on a Mainframe with a hard disk drive, insert the diskette into the Mainframe floppy drive.

press... 

press... 

select... COPY DISK FROM DR1 TO HDR

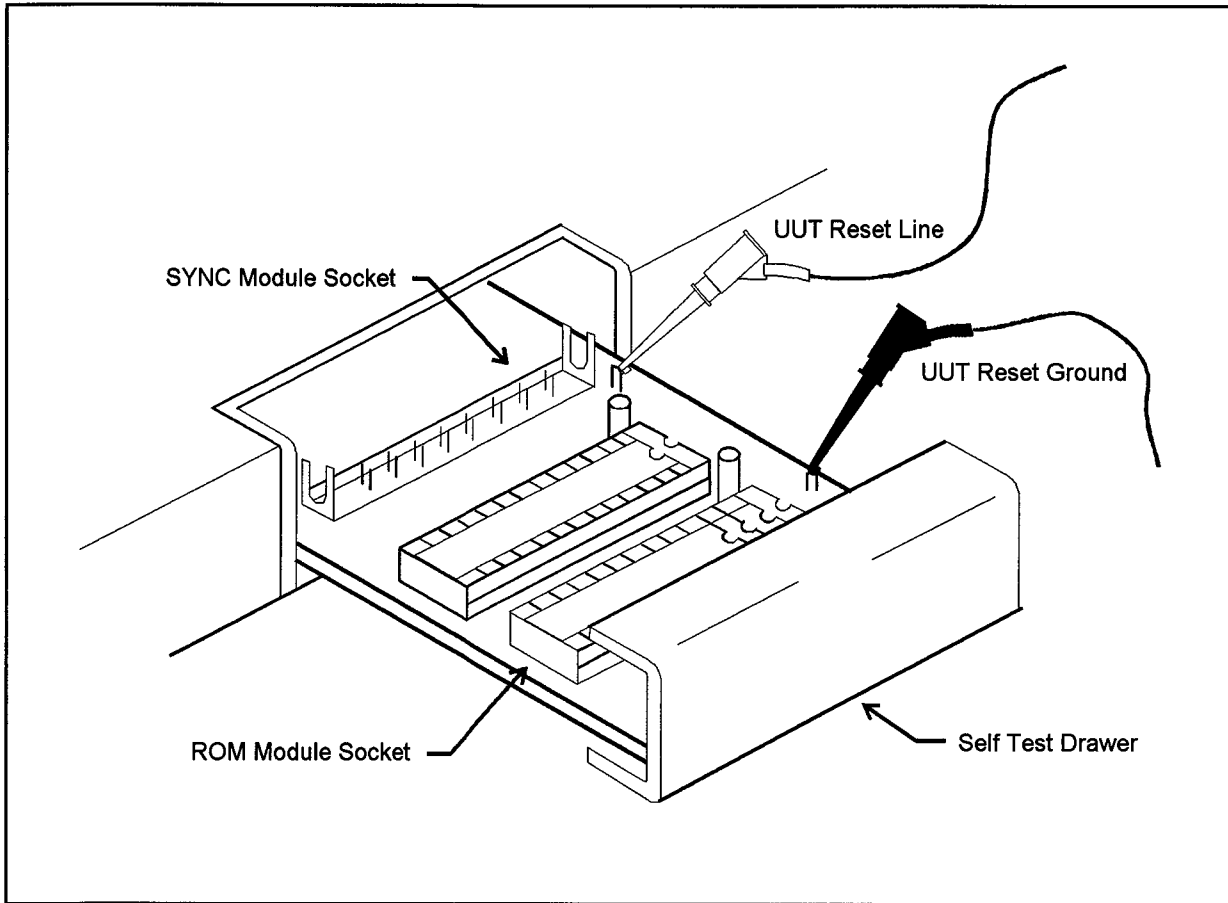
press... 

(The database only needs to be installed once.)

After the database has been installed on the Mainframe, store the original floppy diskette in a secure and accessible place in the event it is needed in the future.


copy... 80286 Master User Disk from DR1 to HDR.

press...  (RESET) on the 9100 keypad.




POD SELF TEST

EXERCISE 4-2

- ① Make sure the Pod is properly connected 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 flying lead to the reset line test point and the UUT Reset ground flying lead to the ground line test point on the self test pca.
- ⑤ Insert the "ROM Module 1 flex cable" into the "ZIF self-test socket". (ROM Module adapter sockets must be empty.)
- ⑥ *press...*  key on the Mainframe to obtain the display:

MAIN: SELFTEST POD

press... 

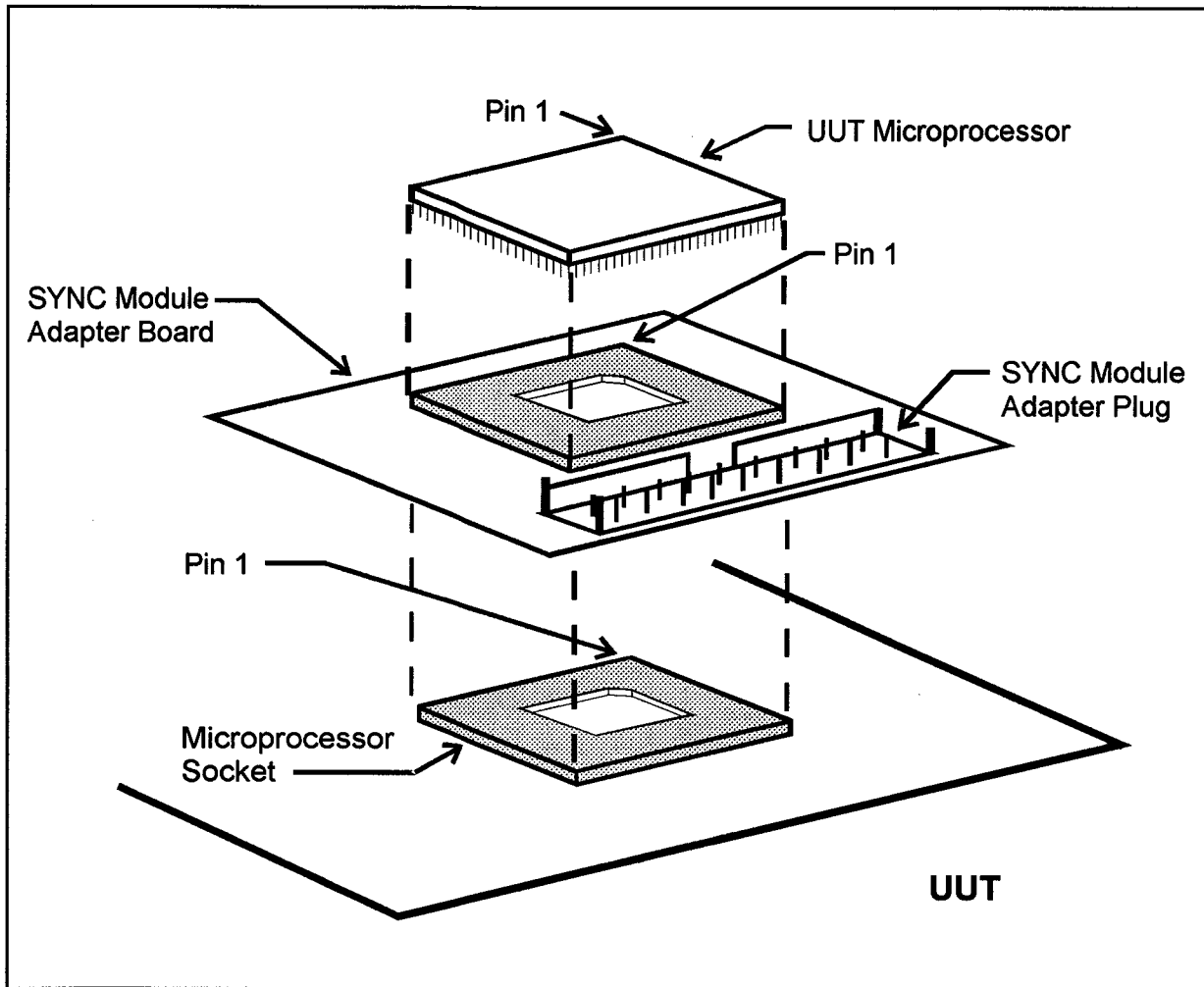
The following will be displayed if Pod passes self-test:

MAIN: SELFTEST POD COMPLETE

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

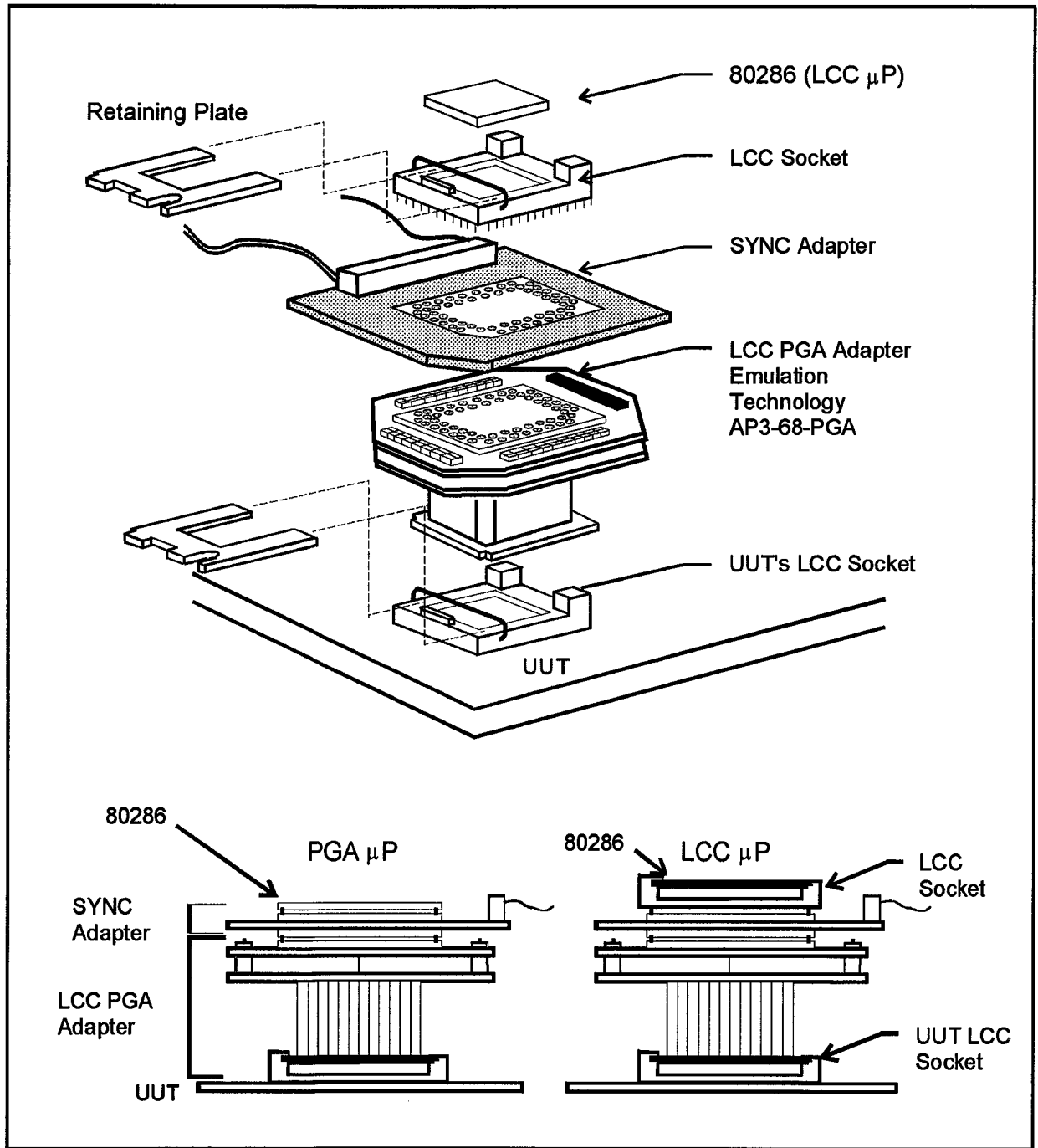
If the self test fails refer to 9132FT-80286 Instruction Manual - Appendix F.

Sync Module Connection to the UUT



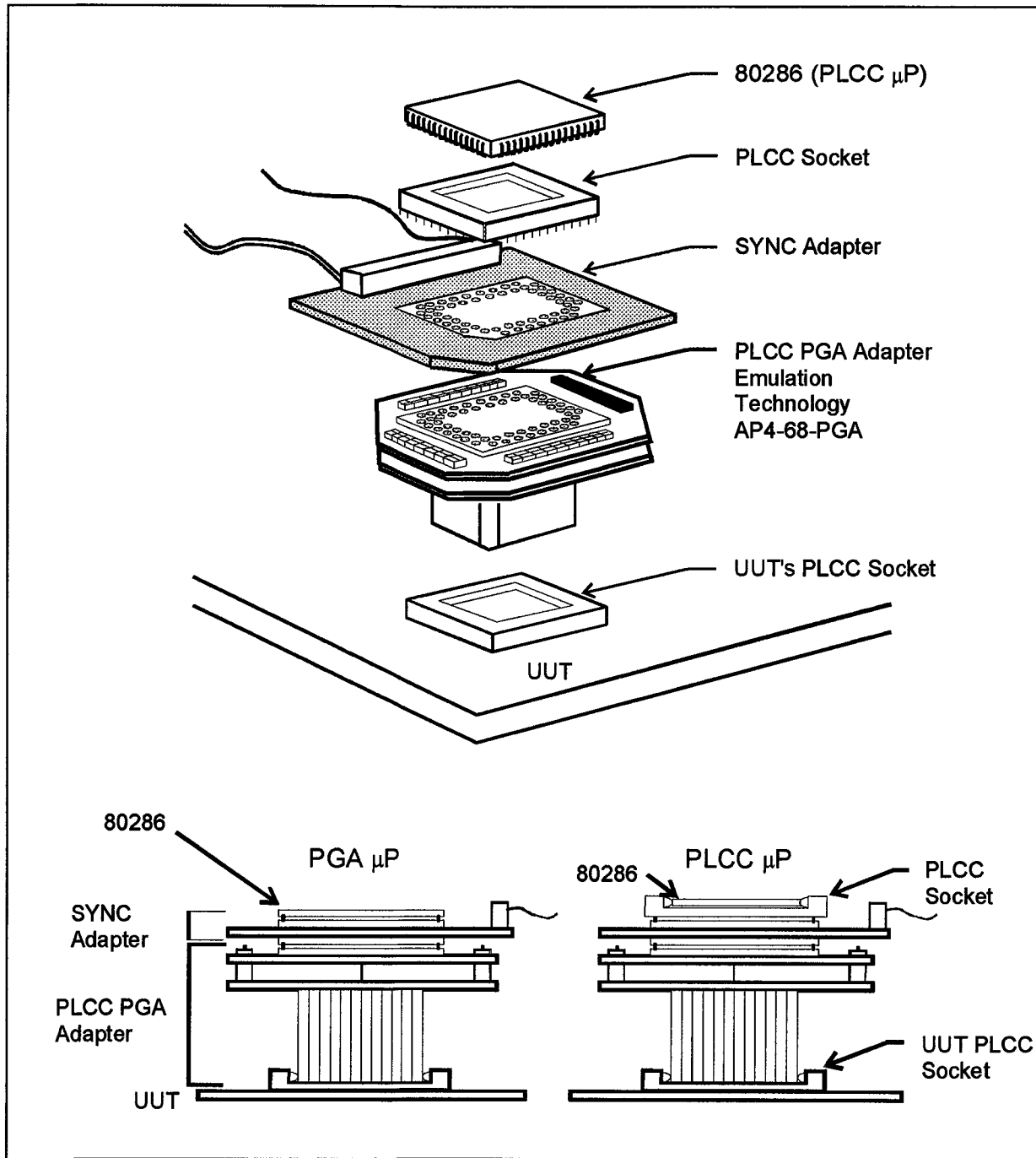
- ① PGA (Pin Grid Array) up to PGA socket adapter.

Sync Module Connection to the UUT Cont..



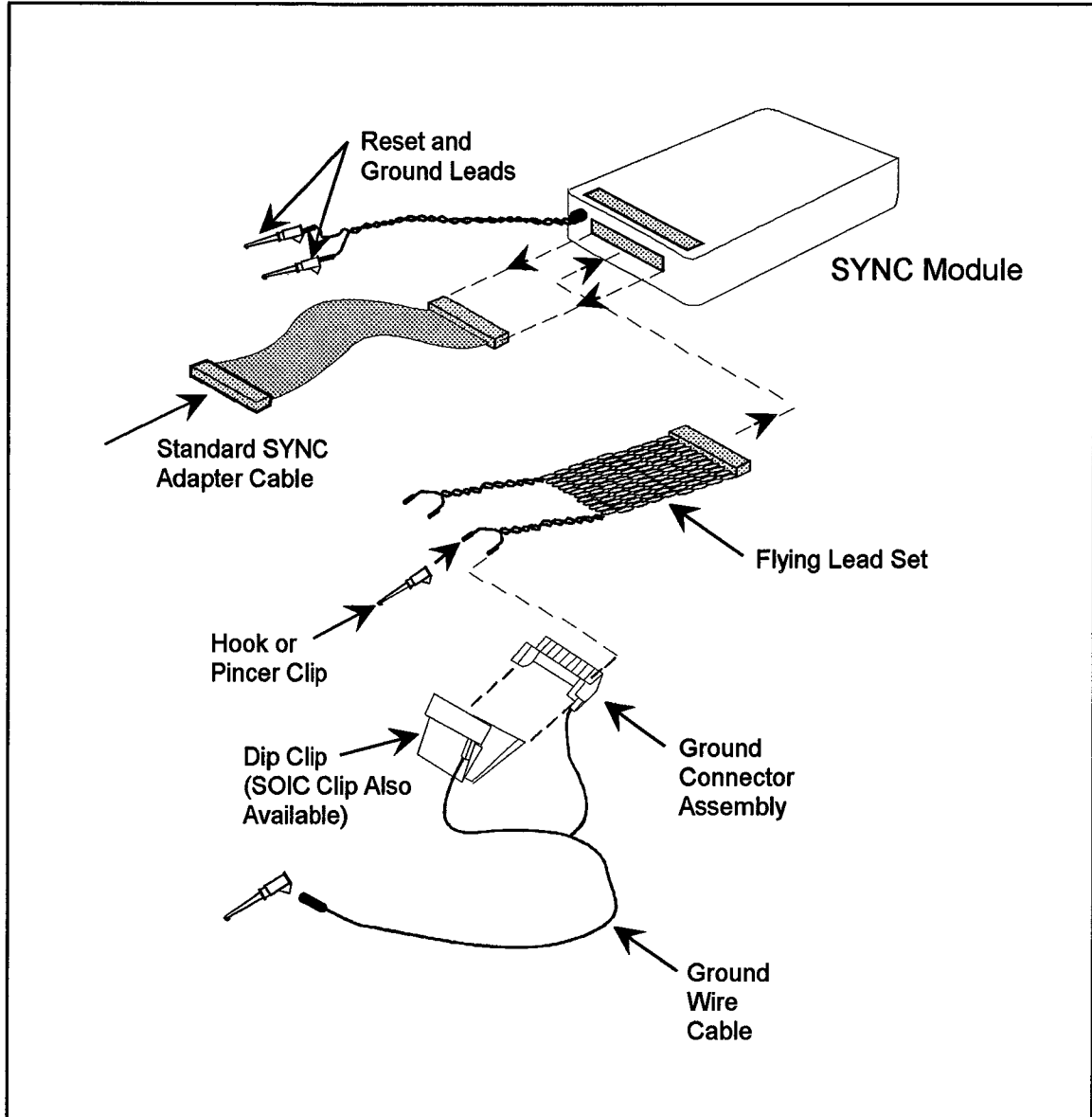
② LCC up to LCC (Leadless Chip Carrier) socket adapter.

Sync Module Connection to the UUT Cont...



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

Sync Module Connection to the UUT Cont...



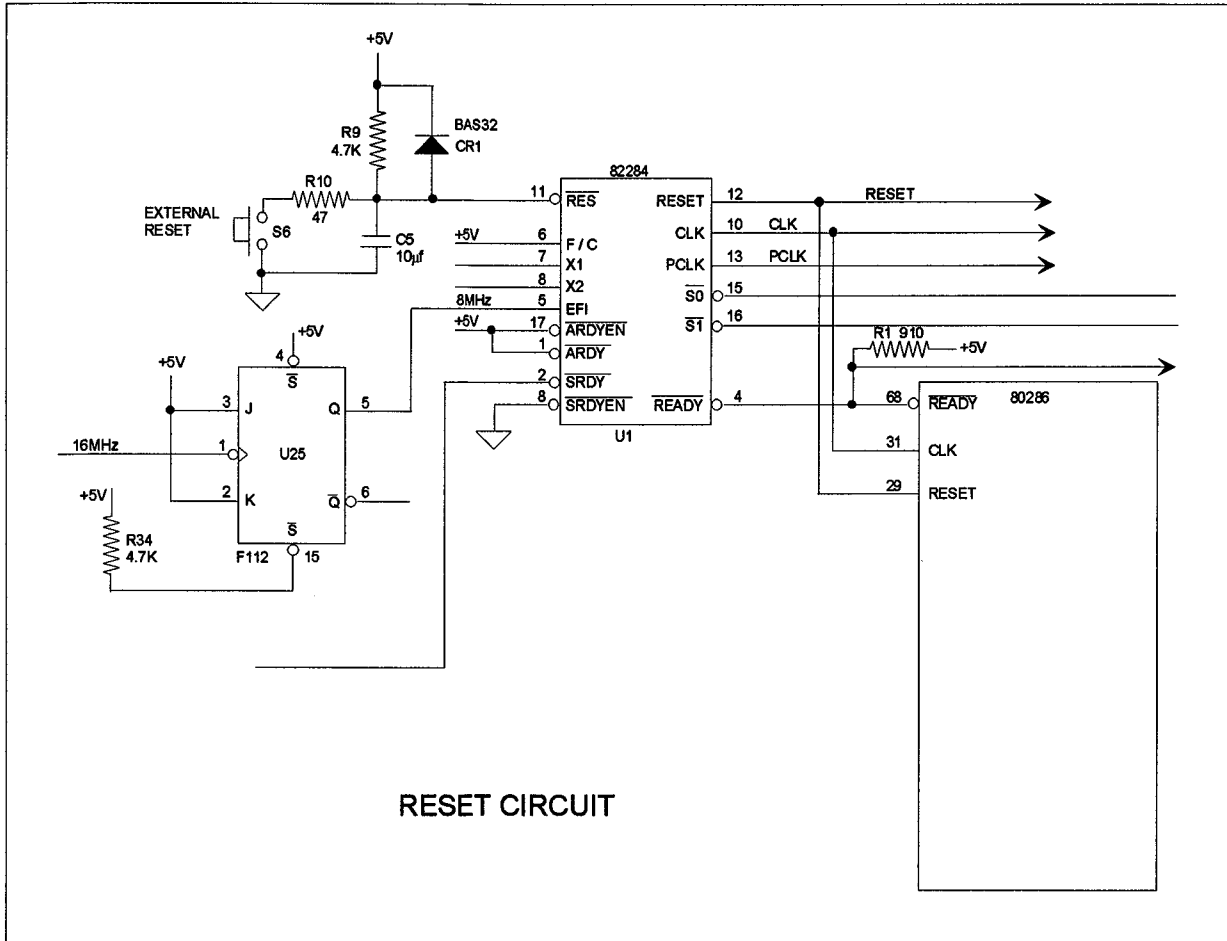
④ Flying lead set.

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	<u>READY</u>
23	GND	24	HOLD
25	GND	26	HLDA
27	GND	28	<u>S0</u>
29	GND	30	<u>S1</u>
31	GND	32	PEREQ
33	GND	34	(KEY)

Sync Module Connection to the UUT Cont..

- ⑧ Connect the 68 pin "PGA socket saver" to the ZIF socket on the UUT.

- ⑨ 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.)



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.

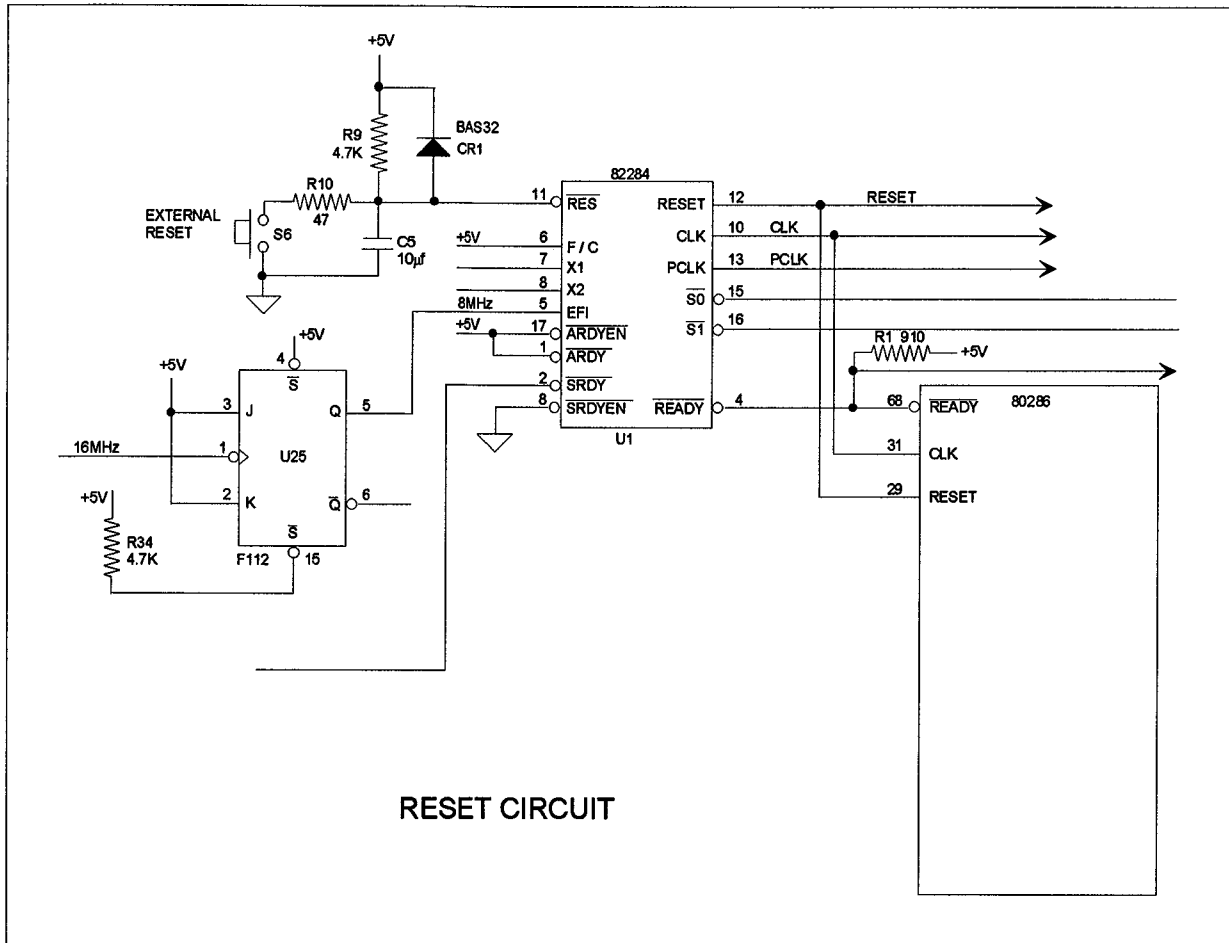
For a full functional test, the best connection position is one that resets as much of the UUT as possible, like a full system reset. This connection allows the test to start as closely as possible to actual UUT reset start-up conditions. Since this connection usually allows UUT hardware to clear a fault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

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

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

Caution!

Be sure that power is removed from the UUT



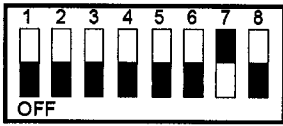
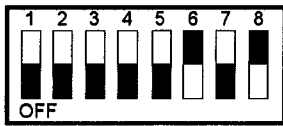
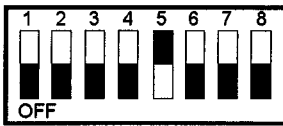
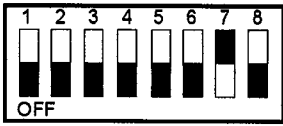
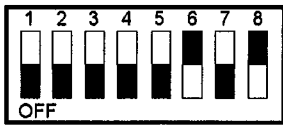
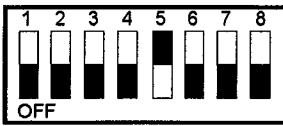
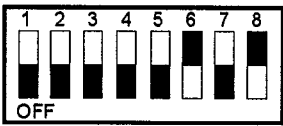
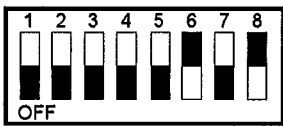
RESET CIRCUIT


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.
- ✓ After certain types of UUT faults.

ROM Module Adapter Switch Settings for Standard ROM Types

ROM TYPE	SWITCH SETTING	ROM TYPE	SWITCH SETTING	ROM TYPE	SWITCH SETTING
2716		27128		27010	
2732		27256		27020	
2764		27512		271024 272048	N/A N/A

NOTE:  = PRESSED

ROM Types Similar to Standard ROMs

NUMBER	ORGANIZATION	SELECT ROM_TYPE
2516	2K X 8	2716
27C16	2K X 8	2716
27C32	4K X 8	2732
27C64	8K X 8	2764
27C128	16K X 8	27128
27C256	32K X 8	27256
27C512	64K X 8	27512
27C010	128K X 8	27010
27C1001	128K X 8	27010
27C020	256K X 8	27020
27C1024	64K X 16	271024
27C010	64K X 16	271024
27C2048	128K X 16	272048
27C220	128K X 16	272048

ROM Module Connection to UUT

EXERCISE 4-3

- 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".

CAUTION!

Make sure the ROMS are inserted in the socket with proper orientation. Damage will occur to the ROM, ROM Module or both.

- ④ Plug "ROM Module #2" into the socket in which 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 in which the low-byte boot ROM had resided.

Refer to 9132FT-80286 Instruction Manual - Appendix C if your boot ROMs are soldered-in.

- ⑧ Verify all connections then turn ON power to the UUT. (Be sure that power to the Mainframe is already ON.)

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

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

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

NOTES:

POD SETUP


EXERCISE 4-4

Using the Interactive Setup and Calibration routine to setup and calibrate the Pod.

To begin the setup procedure:

press...  key on the 9100 keypad.

select...  (SETUP)

press... 

(Wait for the software to be accessed.)

Select desired action:

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 FFFFFFFF

Default: FFFFFFF0

Specifies the address 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.

BOOT_ADR

Values: FFXXXX, 0FXXXX

Default: FFXXXX

Initial fetch address upon reset.

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.

Exercise 4-4, Pod Set Up, Cont...

select... (INFO) Read the presented information. (Use the key to display the next screen.)

select... (SETUP) The following steps will follow the text on the 9100 screen:

1. Verify UUT communications address.
XFER_ADR is presently set to FFFFF0.
OK CHANGE EXPLAIN
2. Verify the boot ROM base address.
BOOT_ADR is presently set to FFXXXX.
OK CHANGE EXPLAIN
3. Verify UUT boot ROM type.
ROM_TYPE is presently set to 27256.
OK CHANGE EXPLAIN
4. Verify number of ROM modules connected.
ROM_MODS is presently set to 2.
OK CHANGE EXPLAIN

Please turn off UUT power if the next step requires changing DIP switches on the ROM module flex cable. Switching the DIP switches with the power on can cause 5V to be shorted to signal lines.

-Press ENTER to continue-

5. Verify the flex cable DIP switches are set as shown.

	1	2	3	4	5	6	7	8	
	■	■	■	■	■	■	■	■	

 on
CONT EXPLAIN
6. Verify the type of ROM module connection used.
CLIP_OVR is presently set to NO.
OK CHANGE EXPLAIN
7. Verify that the ROM modules are connected to the UUT's boot ROM sockets.
OK EXPLAIN
8. Verify that the SYNC module is connected to the UUT.
OK EXPLAIN

RST_POL

Values: LOW, HIGH
Default: LOW

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

RST_LEN

Values: 0 to 99999999
Default: 1000

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

CLP_OVR

Values: YES, NO
Default: NO

CLP_OVR is set to YES if clip over connections are used to test boot ROM's that are not removed from the UUT.

ROM_SWP

Values: AUTO, YES, NO
Default: AUTO

Swaps identities of ROM module bytes 1 and 2, and 3 and 4.

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 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.

Exercise 4-4, Pod Setup, Cont...

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

10. Verify RESET pulse polarity.
RST_POL is presently set to LOW.
OK CHANGE EXPLAIN

11. Verify RESET pulse length (micro-seconds).
RST_LEN is presently set to 1000.
OK CHANGE EXPLAIN

12. Verify UUT power is on.
OK

UUT power sensed OK at all ROM Module connections.
CONT

13. To verify the RESET connection requires use of the probe.
OK SKIP

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

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 EXPLAIN

Probe D8 at the microprocessor and press the button on the probe or any key.

ROM module #2 connection is OK.
CONT

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.

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.

Exercise 4-4, Pod Setup, Cont...

16. The basic pod setups are completed. The following items have been automatically set:

Setup INTERFACE BCYCLCLK is SYNC_MOD.
Setup INTERFACE DATAPRB IS NO.

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

-Press ENTER to continue-

17. *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.

18. *select...* (INFO) *and read the presented information.*

19. *select...* (BUS_TEST)

Calibrating BCYCLCLK and BURST_SZ
(will be displayed briefly)

Both BCYCLCLK and ROM_CE appear reliable. You may choose which bus cycle clock source you want to use.

-Press ENTER to continue-

Verify bus cycle clock source.

BCYCLCLK is presently set to SYNC_MOD.

OK CHANGE EXPLAIN

Completed.

Setup INTERFACE BCYCLCLK is now SYNC_MOD.

Setup INTERFACE BURST_SZ is now 1.

-Press ENTER to continue-

RD_CAL

Values: 2, 127
 Default: 1

Specifies the number of microprocessor bus cycles expected between a trigger event and the appearance of the read cycle of interest on the UUT bus.

PODSYNC is active during the bus cycle. UUT wait states and data bus width at the boot ROMs may have an effect on this value. The value is most easily and accurately set by executing the Interactive Setup and calibration routine. For more information, see Section 2 of 9132FT-80286 Instruction Manual. A separate calibration exists for each address space.

MEM_W	memory word	Default: 2	I/O_B	I/O byte	Default: 2
MEM_B	memory byte	Default: 2	LOCK	locked memory	Default: 1 *
I/O_W	I/O word	Default: 2		byte	

WR_CAL

Values: 2, 127
 Default: 1

Specifies the number of microprocessor bus cycles expected between a trigger event and the appearance of the write cycle of interest on the UUT bus.

PODSYNC is active during this bus cycle. UUT wait states and data bus width at the boot ROMs may have an effect on this value. The value is most easily and accurately set by executing the Interactive Setup and Calibration routine. For more information, see Section 2 of 9132FT-80286 Instruction Manual. There is a separate calibration for each address space:

MEM_W	memory word	Default: 2	I/O_B	I/O byte	Default: 2
MEM_B	memory byte	Default: 2	LOCK	locked memory	Default: 1 *
I/O_W	I/O word	Default: 2		byte	

* LOCK defaults are set to 1 to force the SETUP program to actively re calibrate this space (because of the special characteristics of the 80286 -modify-write cycle).

Exercise 4-4, Pod Setup, Cont...

19. Cont.

Checking ADR_STIM and CY_SPLIT
(will be displayed briefly)

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

Setup CALIBRTN ADR_STIM is now 5.
Setup INTRFACE CY_SPLIT is now 1.
-Press ENTER to continue-

The bus test calibrations have been completed. The pod
is now ready to perform read/write sync pulse calibrations.
(Select the SYNC function)
-Press ENTER to continue-

20. *select...* (SYNC)

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

Calibrating read sync pulses . . . (will be displayed briefly)

Calibrating write sync pulses . . . (will be displayed briefly)

The sync pulse calibrations are completed.
CONT

21. *select...* (RUN_UUT)

Calibrating RUN_UUT. . .
(will be displayed briefly)

Completed.
Setup CALIBRTN RUN_UUT is now 5.
CONT

22. *select...* (MAINMENU)

SAVE SYSTEM SETTINGS IN USERDISK

Exercise 4-4, Pod Setup, Cont...

23. select... (CHECK)

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

(The following will be 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

24. select... (QUIT)

(Read message provided)

25. Save pod setups.

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

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

Refer to 9132FT-80286 Instruction Manual - Appendix D and observe the parameters for setup and calibration that can be set using the front panel key .

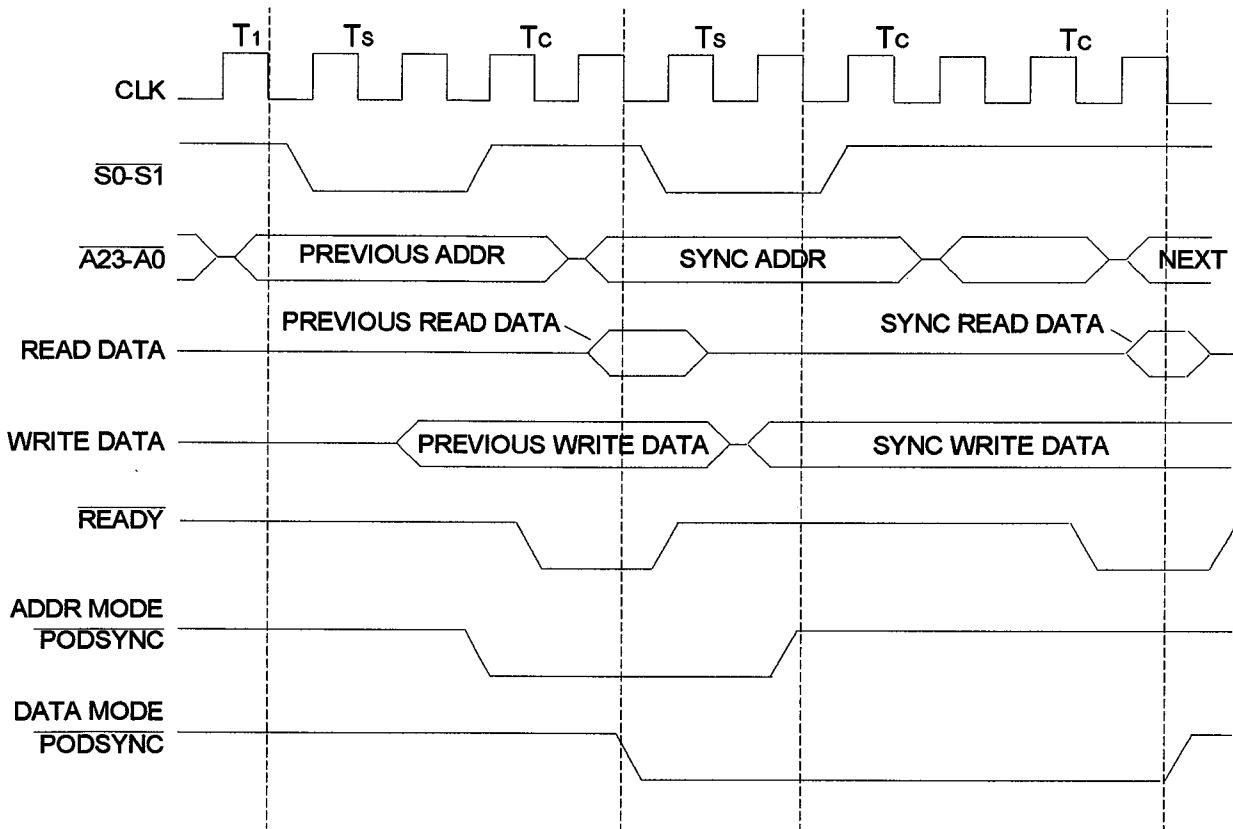
Program setup

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

end setup

EXERCISE 4-5**SETUP Program**

With the help of the 9132FT-80286 Instruction Manual - Appendix E write a TL/1 program to setup the Demo board.



CALIBRATION OF THE PROBE

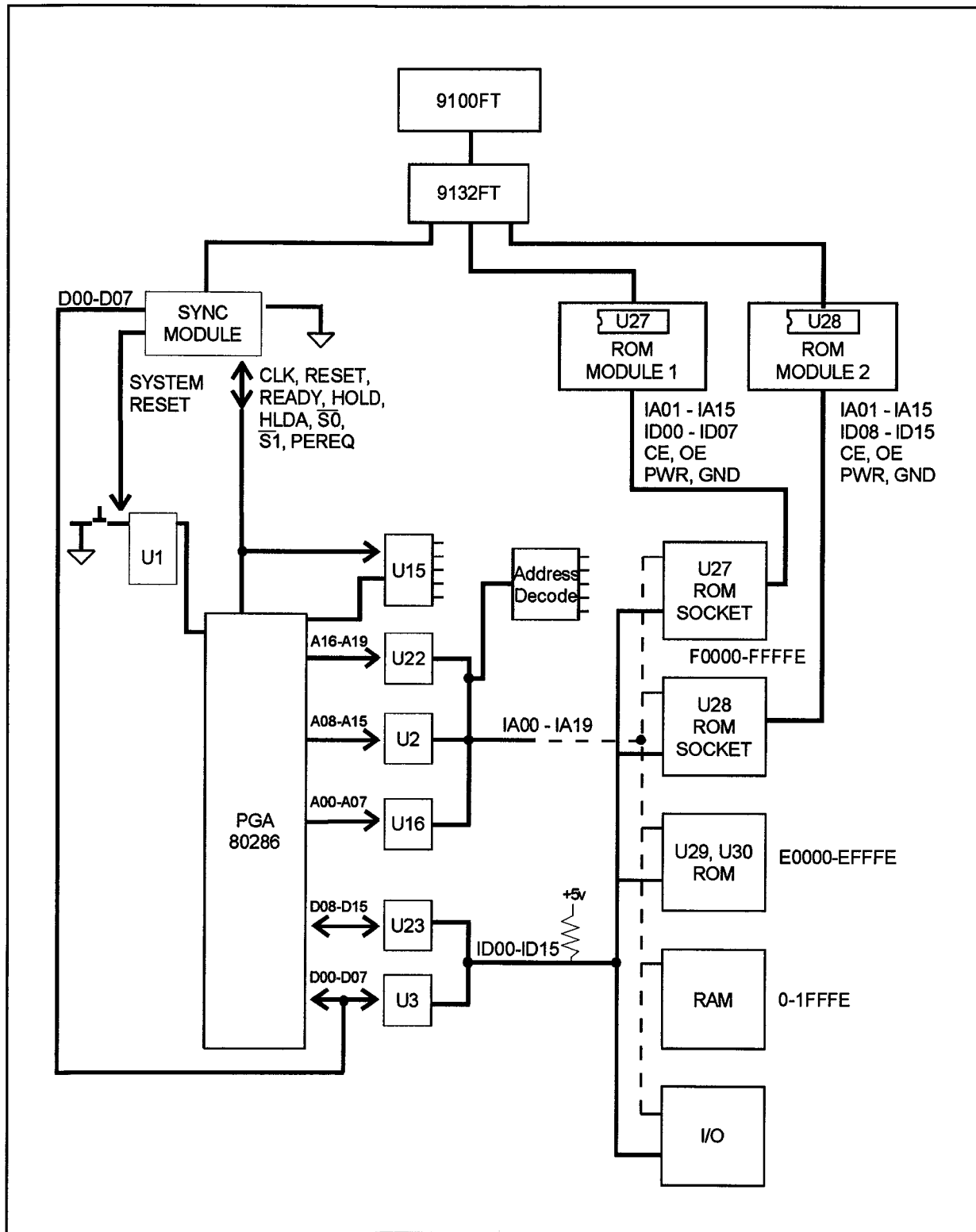
The purpose of calibration is to ensure that the probe samples signals on the UUT at a known point in time with respect to the UUT timing signals such as CLK and READY.

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 9132FT 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.

EXERCISE 4-6

press...  on the keypad and calibrate the probe.



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.

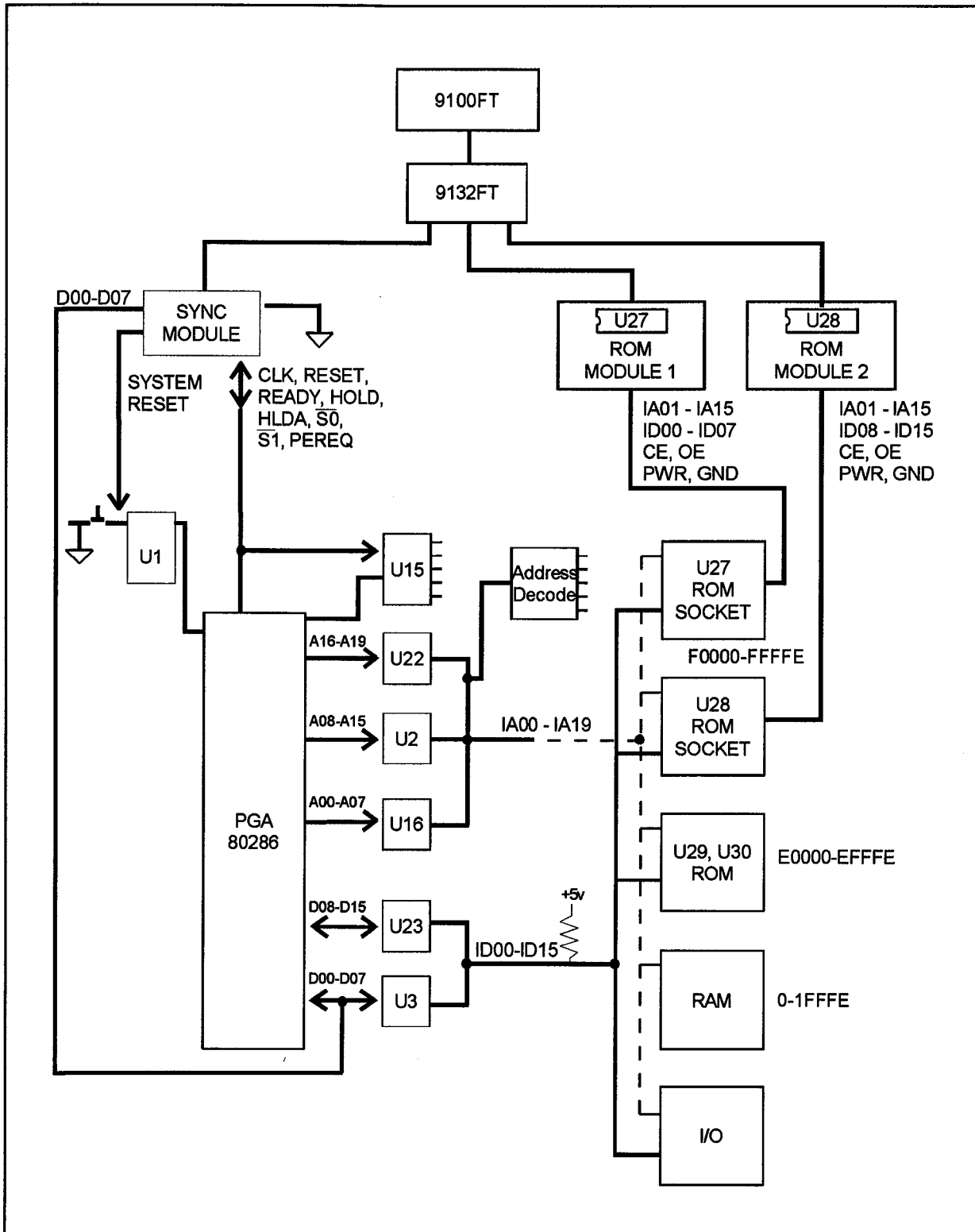
If a fault *is* found, the program prints a message on the Mainframe display and transfers execution to the program TEST_BUS for execution of the fault diagnostics.

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.

EXERCISE 4-7

Run BUS test in the immediate mode with no faults.

Refer to 9132FT-80286 Instruction Manual - Appendix E-6.



TEST BUS

TEST_BUS is 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.

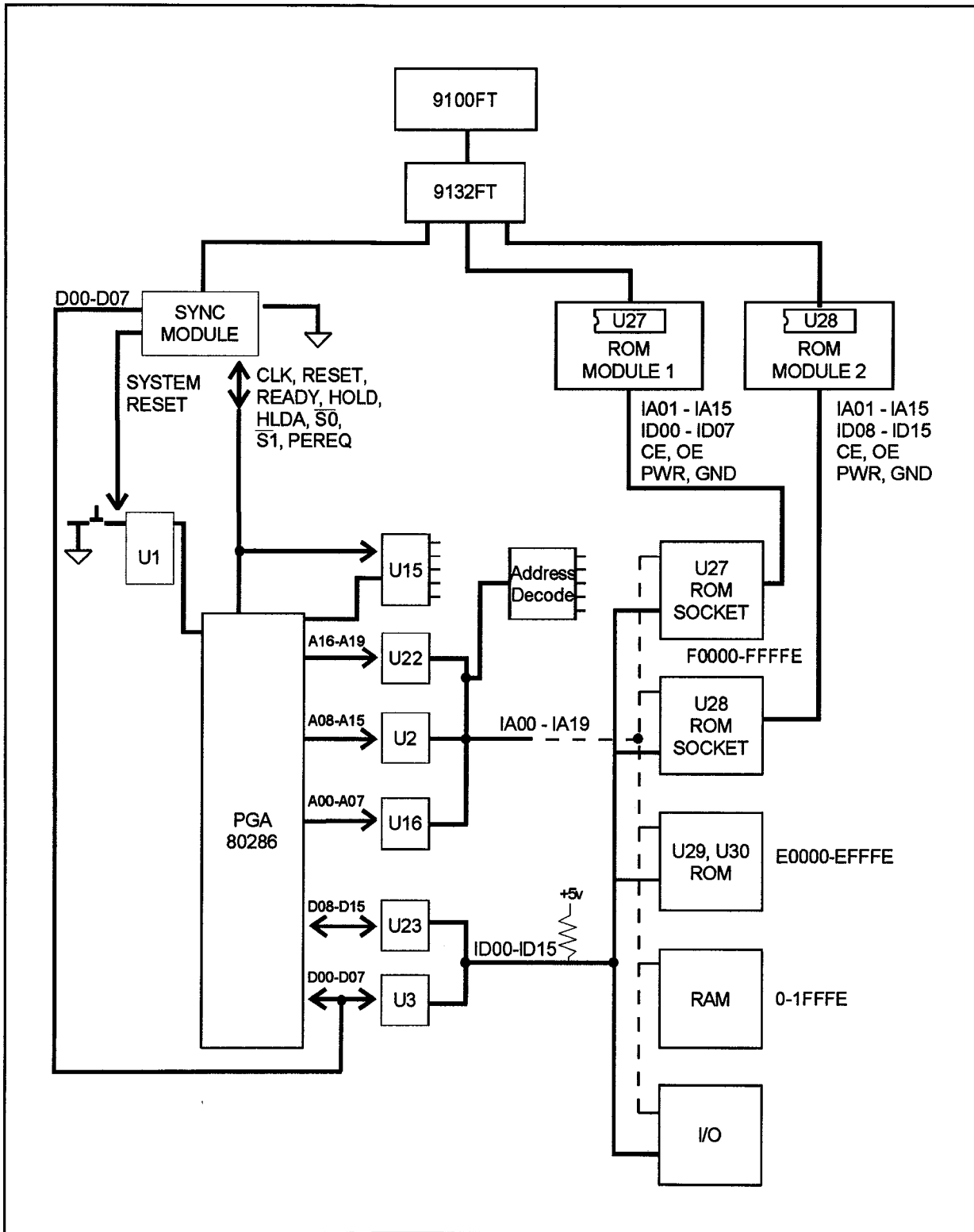
TEST_BUS performs as many diagnostics as possible without any user intervention. The program checks for:

- ✓ 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.
- ✓ 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 key on the Mainframe keypad is pressed, the program, if possible, continues to diagnose further.

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

Refer to 9132FT-80286 Instruction Manual - Appendix E-7.



B DIAG

B_DIAG is a TL/1 program that is executed when the front panel key sequence, DIAGNOSE BUS, is pressed.

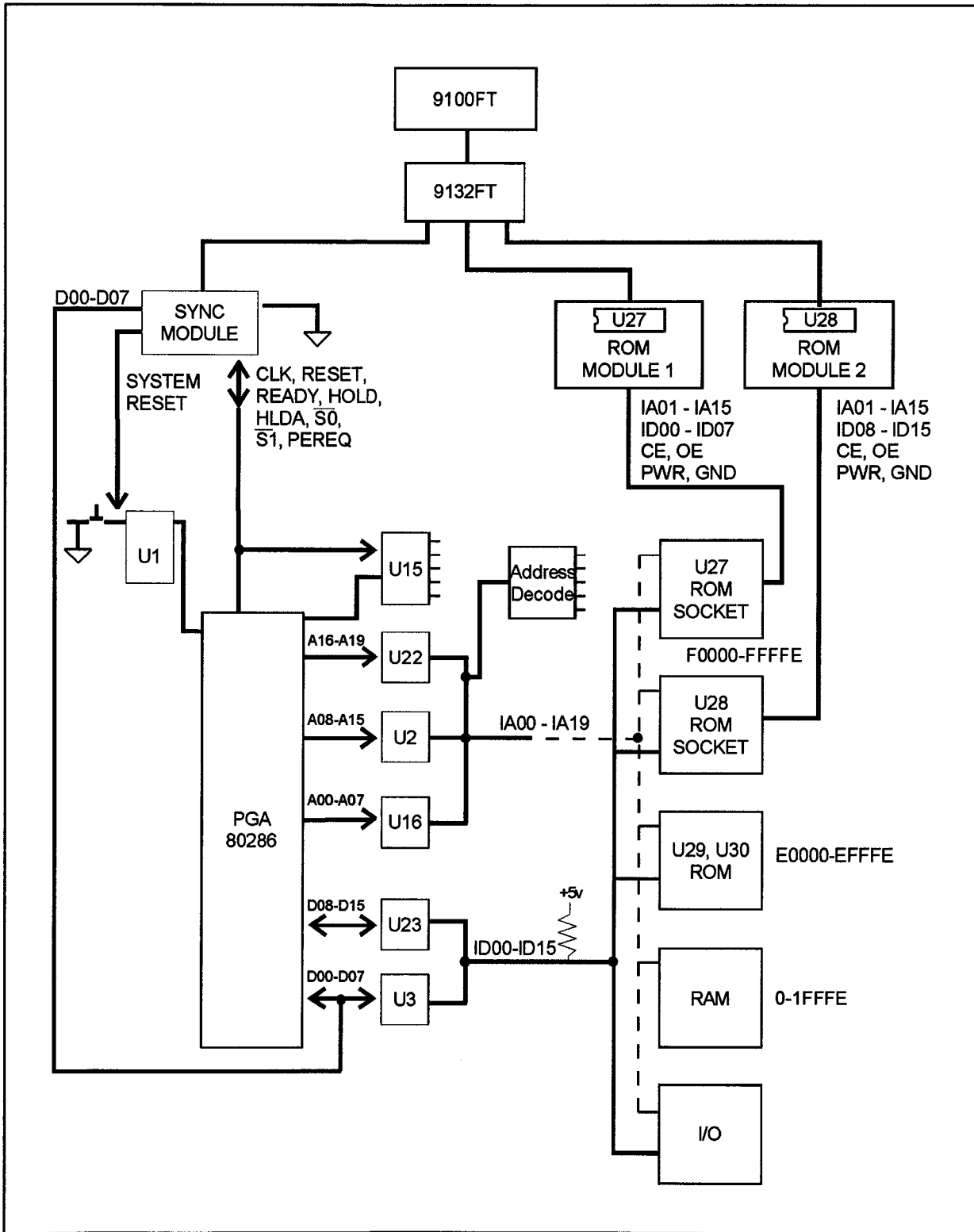
B_DIAG starts by prompting the user to probe the UUT microprocessor's CLK line (and reports the clock frequency measured by the probe), then prompts the user to probe the status and control lines, reporting in each case whether or not a problem was detected.

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

EXERCISE 4-8

Run DIAGNOSE BUS in the immediate mode with no faults.

Refer to 9132FT-80286 Instruction Manual - Appendix E-9.



EXERCISE 4-9

Set the following faults in the UUT one at a time and run TEST BUS in the immediate mode. Write the error message for each fault below.

ERROR MESSAGE**UNBUFFERED FAULTS**

Data	Fault Message
Fault Switch 3-1	
Fault Switch 2-7	

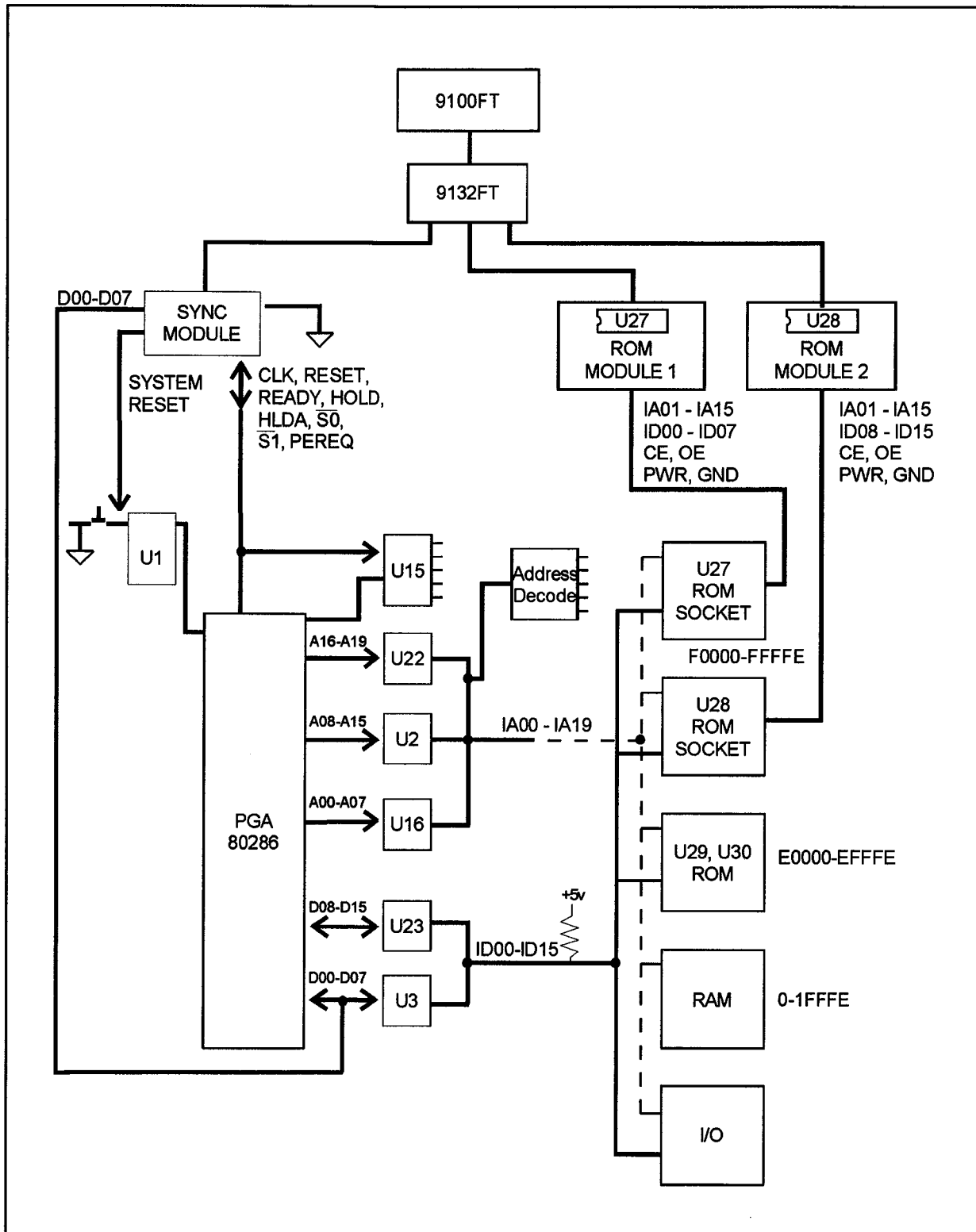
Address	Fault Message
Fault Switch 2-5	
Fault Switch 2-3	

Control	Fault Message
Fault Switch 2-1	
Fault Switch S5 to Run	


BUFFERED FAULTS

Data	Fault Message
Fault Switch 4-2	

Address	Fault Message
Fault Switch 4-1	



Fault Isolation

press...  on the keypad. The following should appear on your screen:

```
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.

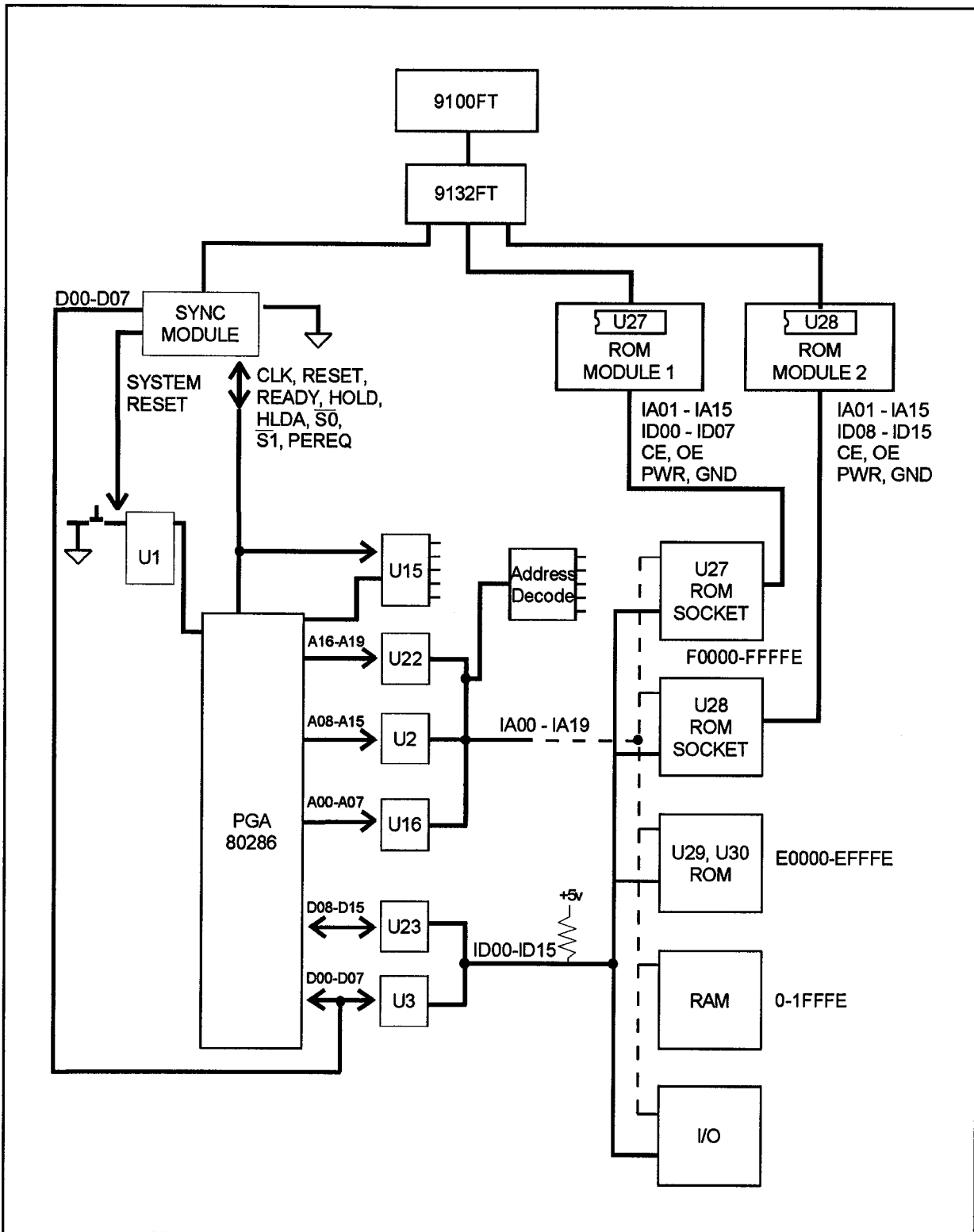
STIM_DAT is the most basic stimulus routine used for diagnosing UUT kernel faults by **TEST_BUS**. This program is useful in stimulus programs for testing inoperative kernels.

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 9132FT-80286 Instruction Manual - Appendix E-11.



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. The microprocessor must be able to successfully fetch several words of data from the ROM Modules for this program to return with "no faults detected". This routine is useful for troubleshooting address bus faults (As long as they were not RESET ADDRESS or ROM 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.

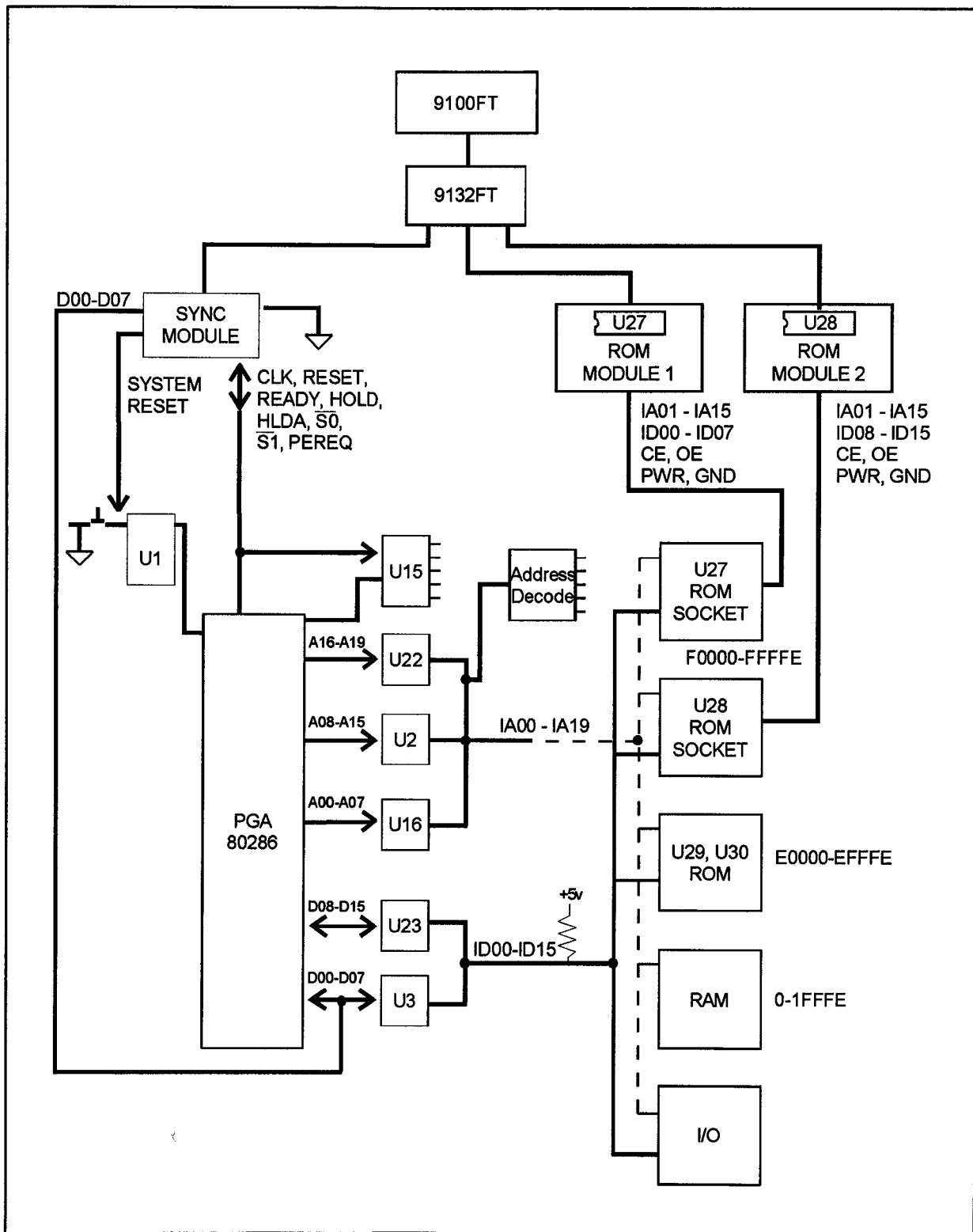
Refer to 9132FT-80286 Instruction Manual - Appendix E-12.

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.



QWK_WR

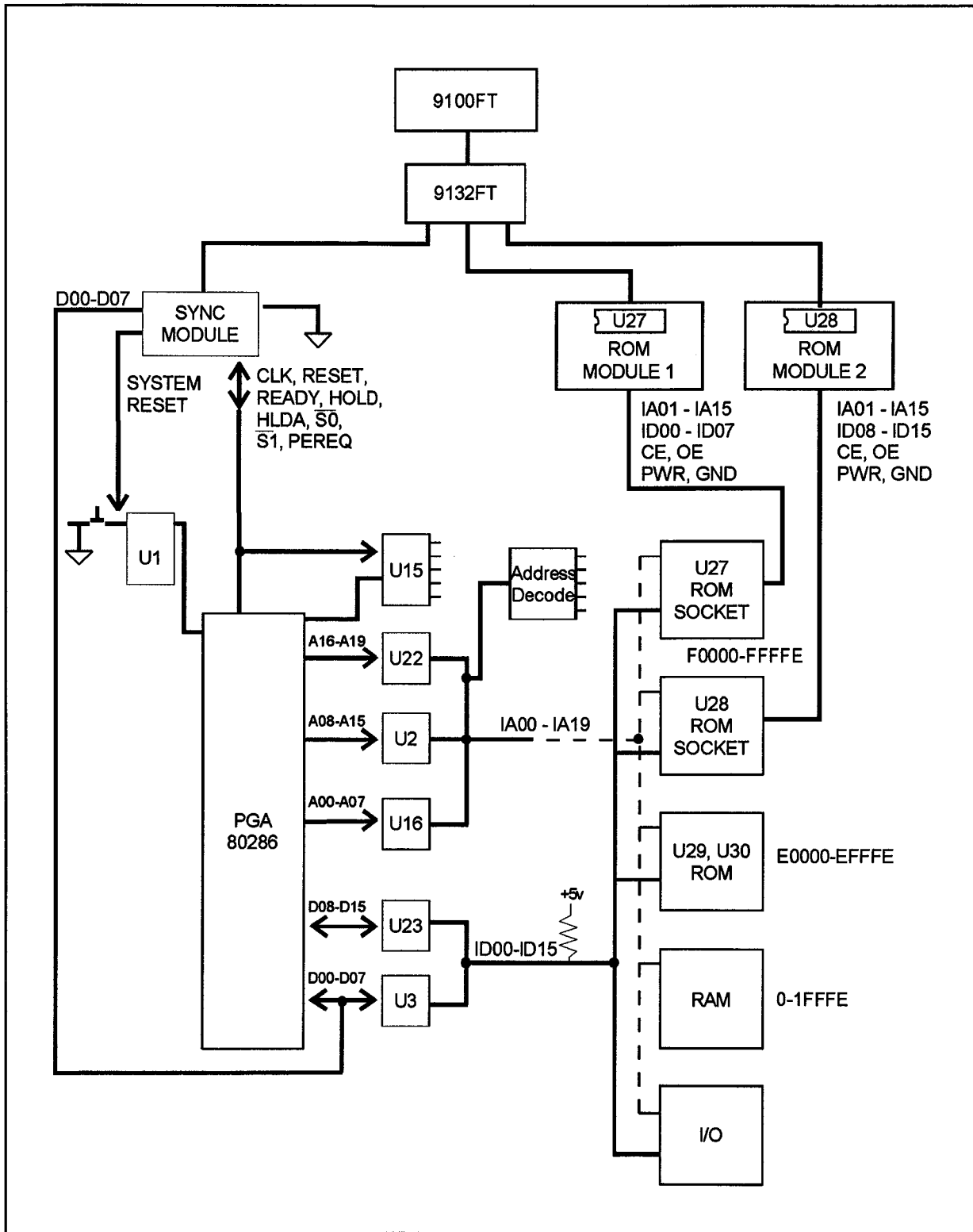
The QWK_WR program causes the Pod to implement a quick looping write function. Once QWK_WR is entered, the program returns at once. Though the program returns immediately, the Pod continues to perform writes at the specified address. A sync pulse is generated for each write, as specified by the current sync mode.

Writing continues until a Pod access of any kind is initiated.

NOTE: Because the UUT is reset each time QWK_WR is exited, looping on this quick function is not recommended.

Arguments: ADDR
DATA

Returns: Nothing



EXERCISE 4-10

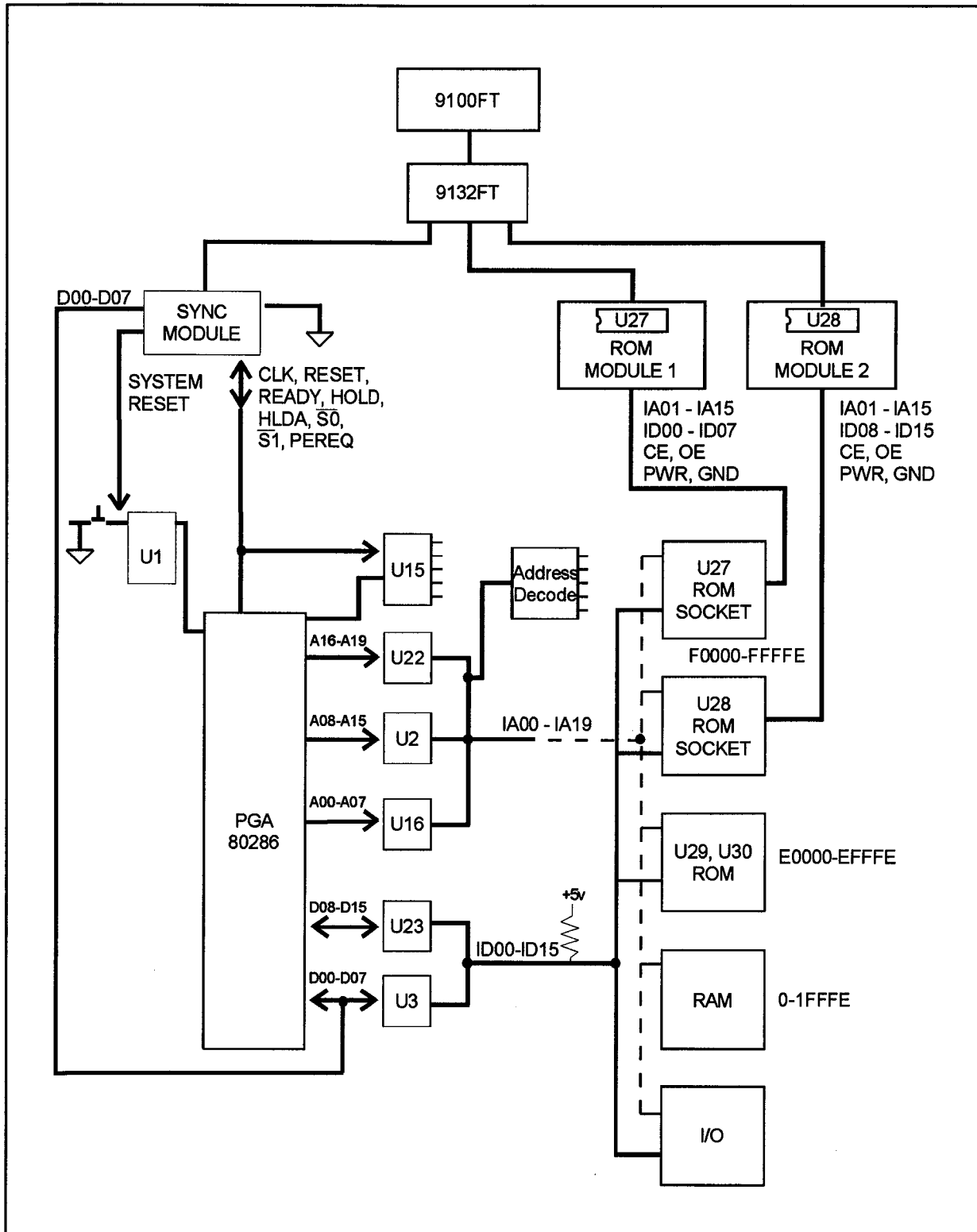
Set the following faults in the UUT one at a time and run TEST BUS in the immediate mode. Try each of the stimulus routines just discussed to find the bad node.

HINT: Reference Exercise 6 on page 51.

Fault Switch 3-1	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	

Fault Switch 2-7	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	

Fault Switch 2-5	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	

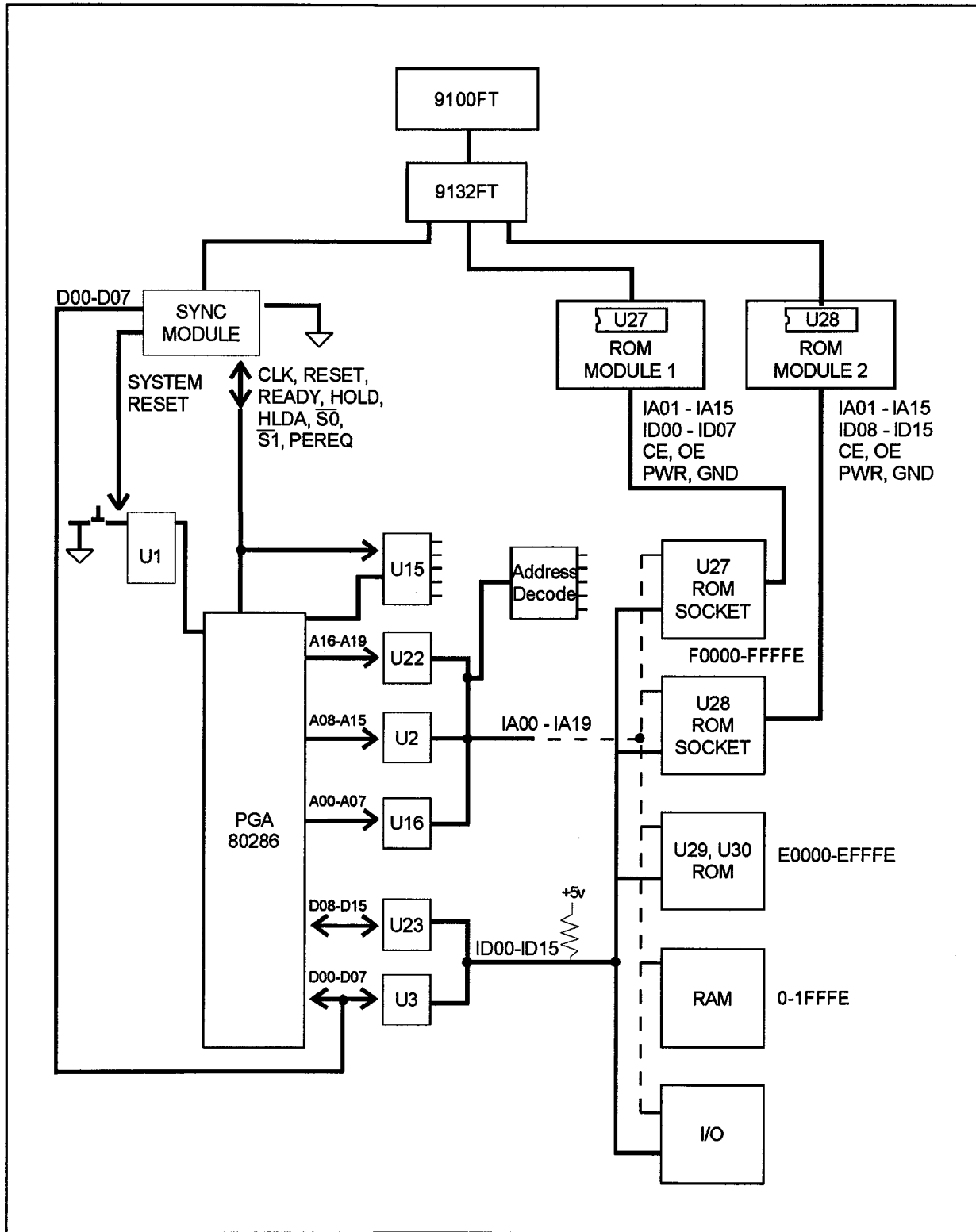


Exercise 4-10 Cont...

Fault Switch 2-3	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	

Fault Switch 4-2	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	

Fault Switch 4-1	
Fault Message:	
Stimulus:	
Sync Mode:	
Bad Node:	



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/writeable.

HyperRAM Test

RAM fault coverage is essentially the same as that of the RAM Fast Test. The difference between them is that the HyperRAM Test is completed in much less time.

RAM Full Test

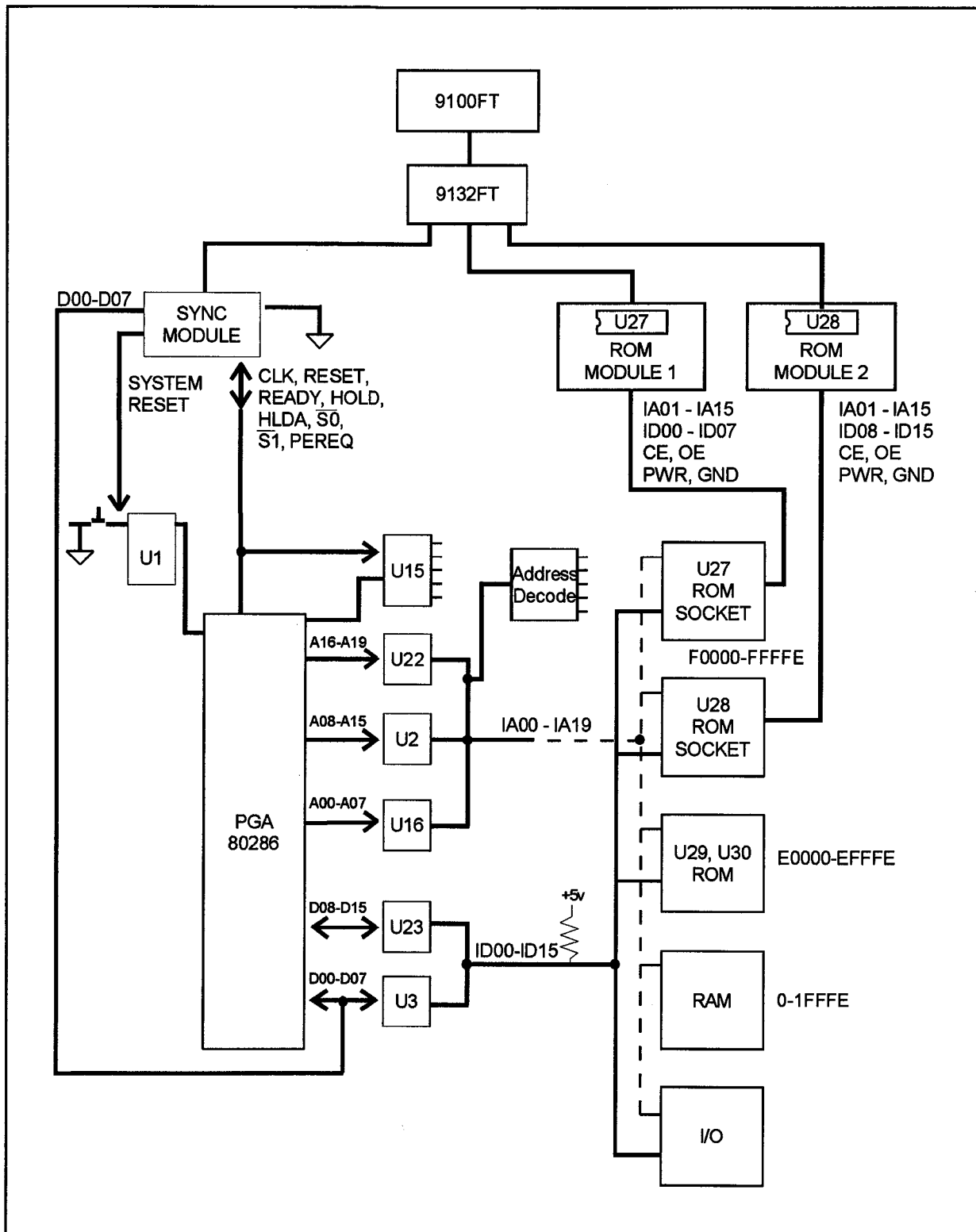
The RAM Full Test is the most comprehensive RAM test algorithm available in the 80286 Pod. Besides the fault types detected by other algorithms available in the Pod, the RAM Full Test makes several additional passes through memory to help find coupling faults.

EXERCISE 4-11

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

Refer to 9132FT-80286 Instruction Manual - Appendix E-12.

Refer to 9132FT-80286 Instruction Manual - Appendix E-16.

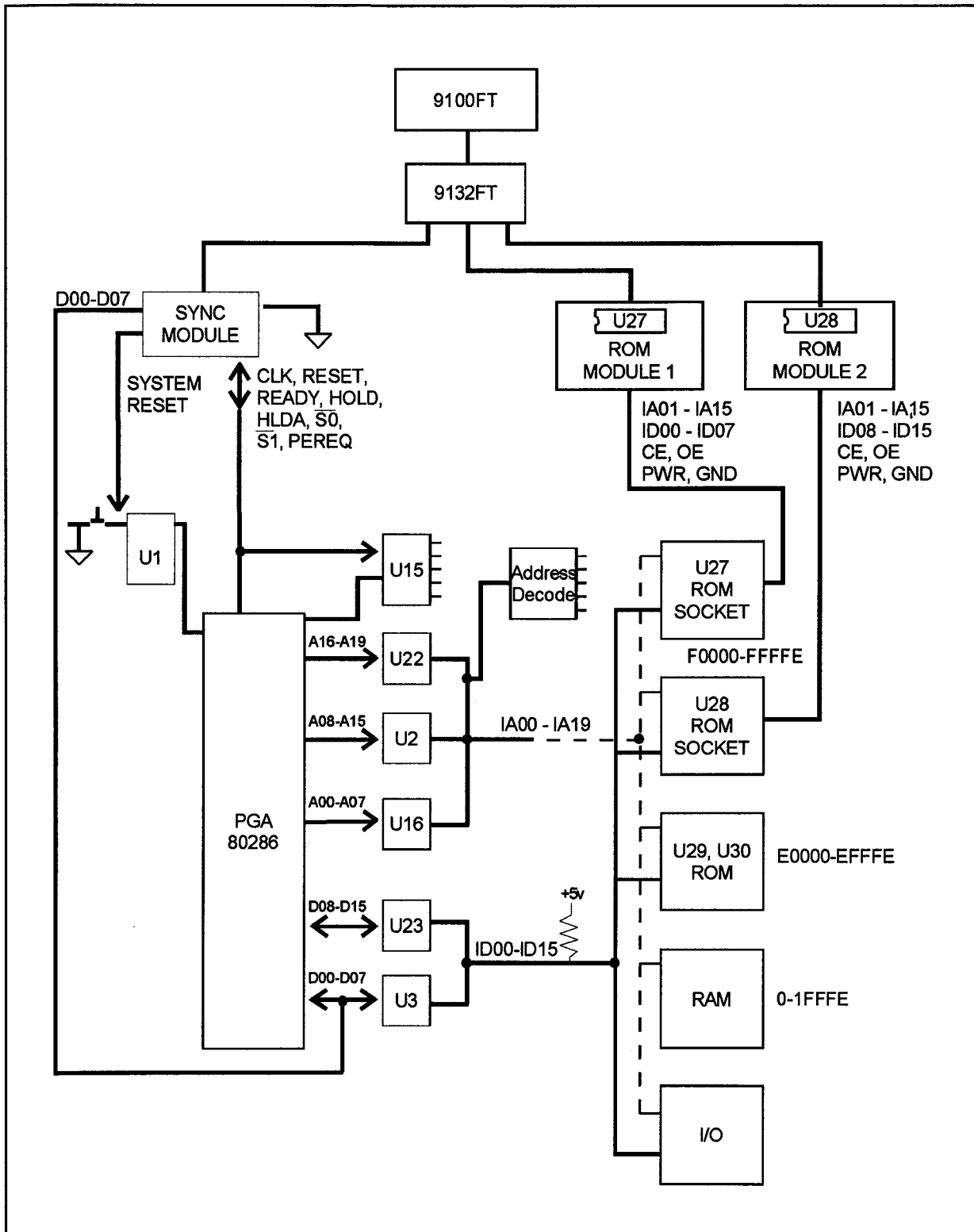


EXERCISE 4-12

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

Fault 4-2	Fault Messages
Functional Test 1:	
Functional Test 2:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	

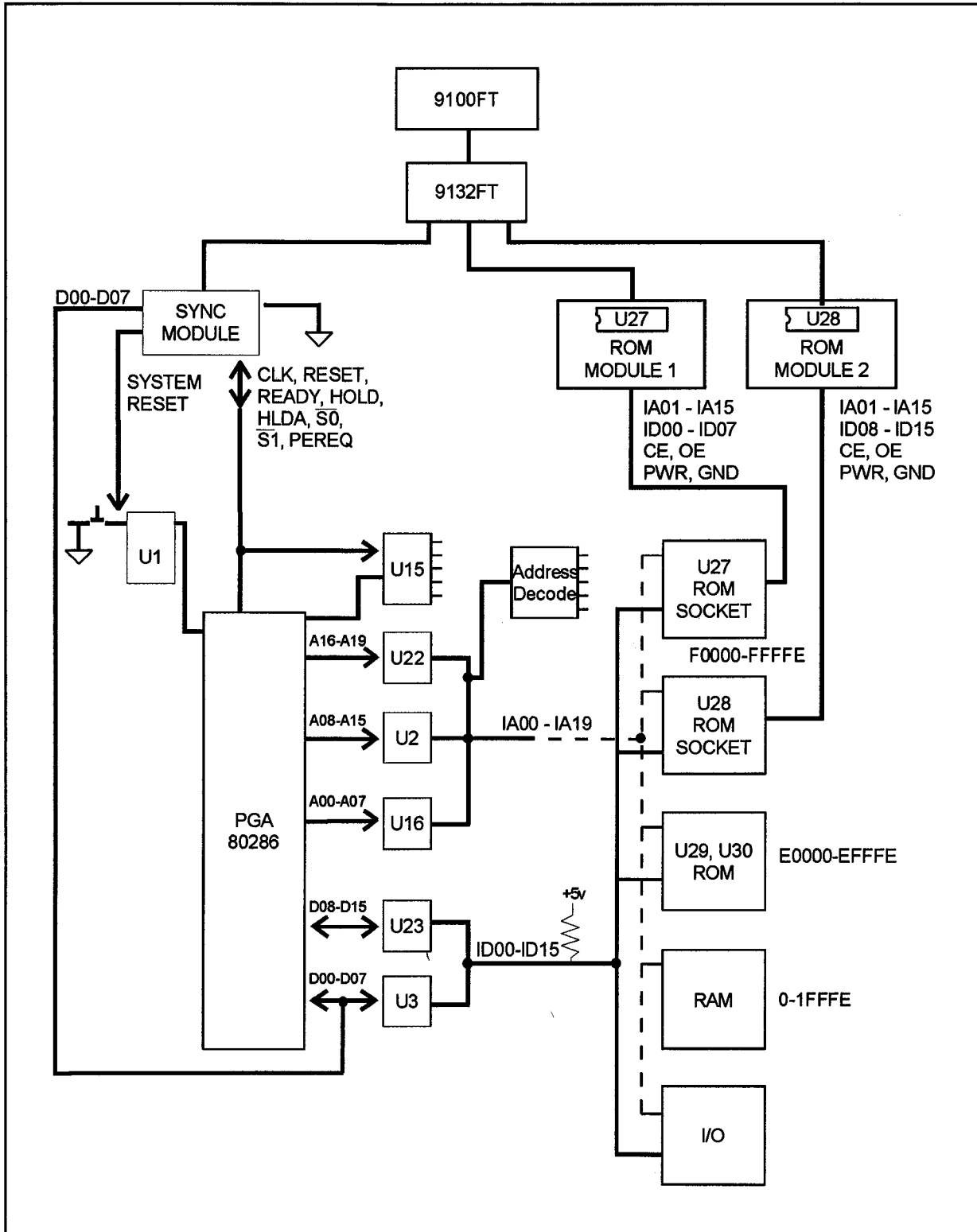
Fault 1-1	Fault Messages
Functional Test 1:	
Functional Test 2:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	



Exercise 4-12 Cont...

Fault 1-5	Fault Messages
Functional Test 1:	
Functional Test 2:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	

Fault 4-8	Fault Messages
Functional Test 1:	
Functional Test 2:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	



Gathering Signatures

EXERCISE 4-13


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...*  on the keypad.
select... GET SIG

6. The following should appear on your screen:

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

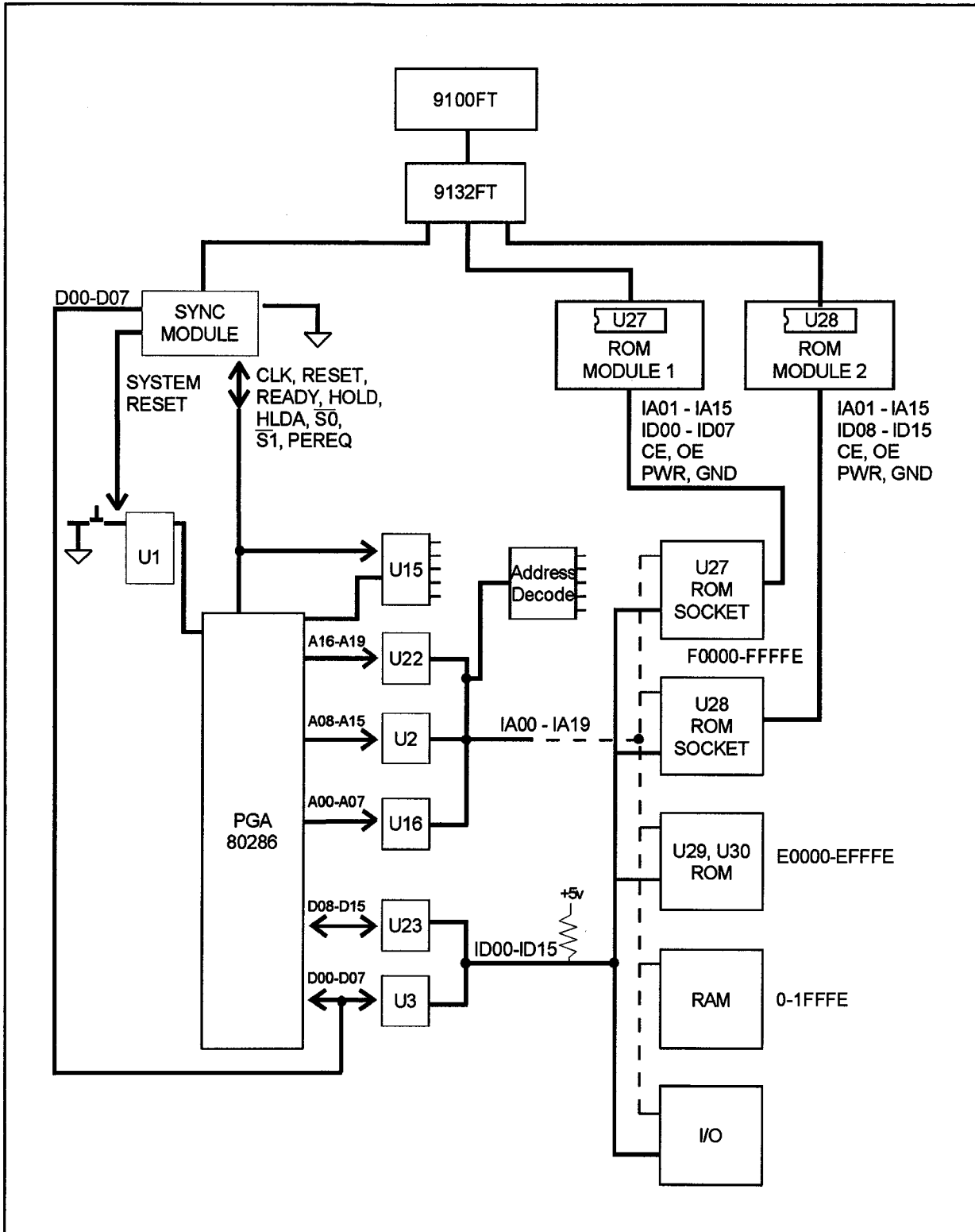
7. *press...*  and wait for the signature. SIG = F387
8. Repeat steps for ROM Module 2. SIG = 8E6E
9. Install ROM Modules back to the UUT.

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...*  on the keypad.
select... GET SIG

5. The following should appear on your screen:

```
GET SIG ROM REF U27 ADDR F000 UPTO FFFF MASK FF STEP 2  
ADDR OPTION: UUT_ROM
```

Exercise 4-13, Gathering Signatures, Cont...

6. *press...* and wait for the signature. SIG = F387
7. Repeat steps for ROM Module 2.

GET SIG ROM REF U28 ADDR F0001 UPTO FFFFF MASK FF STEP 2

8. *press...* and wait for the signature. SIG = 8E6E

FROM OTHER UUT ROMS

1. Use the 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.

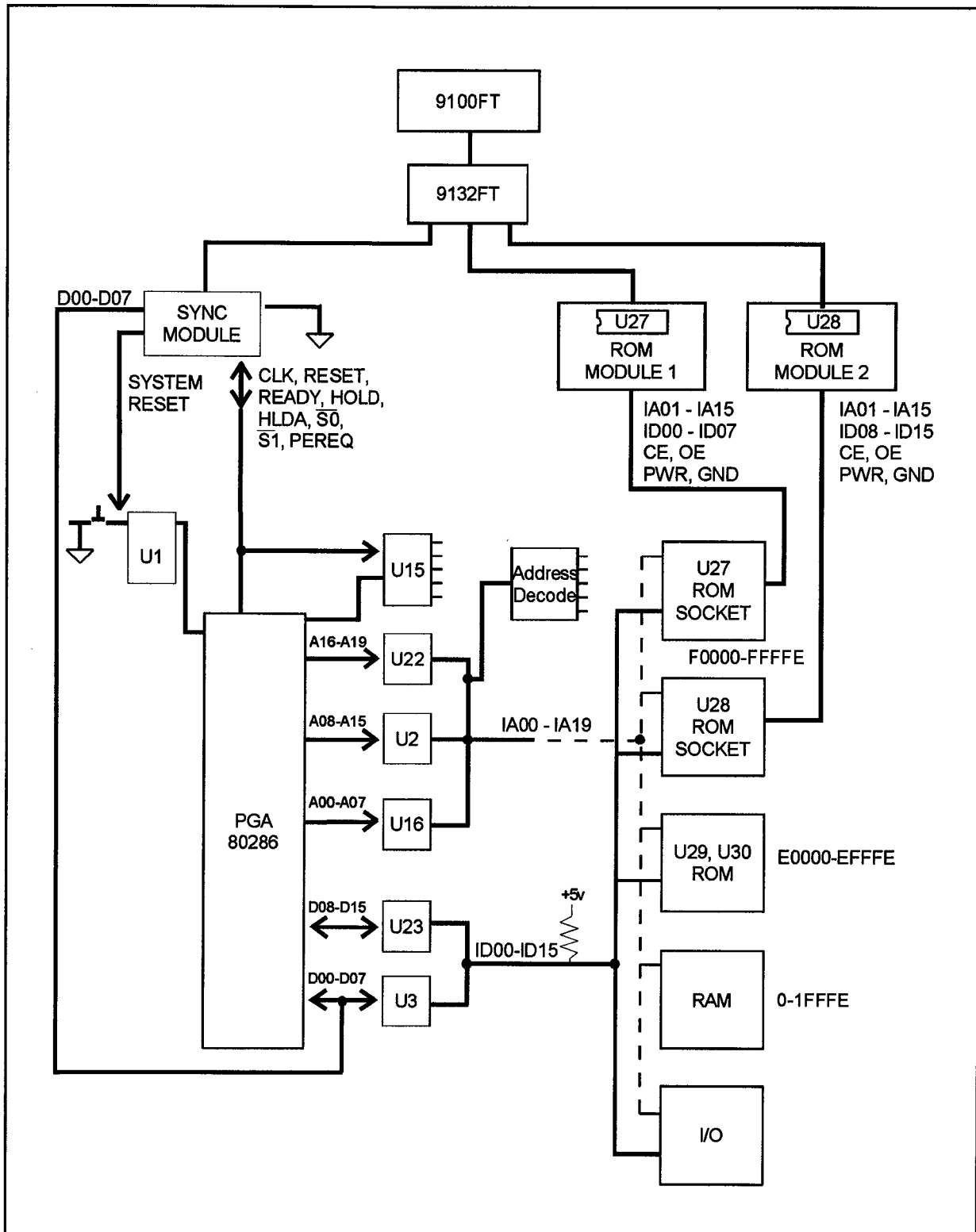
Refer to 9132FT-80286 Instruction Manual - Appendix C-6.

ROM TEST

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

EXERCISE 4-14

Use TEST ROM FULL ALL REF to test the ROMs.

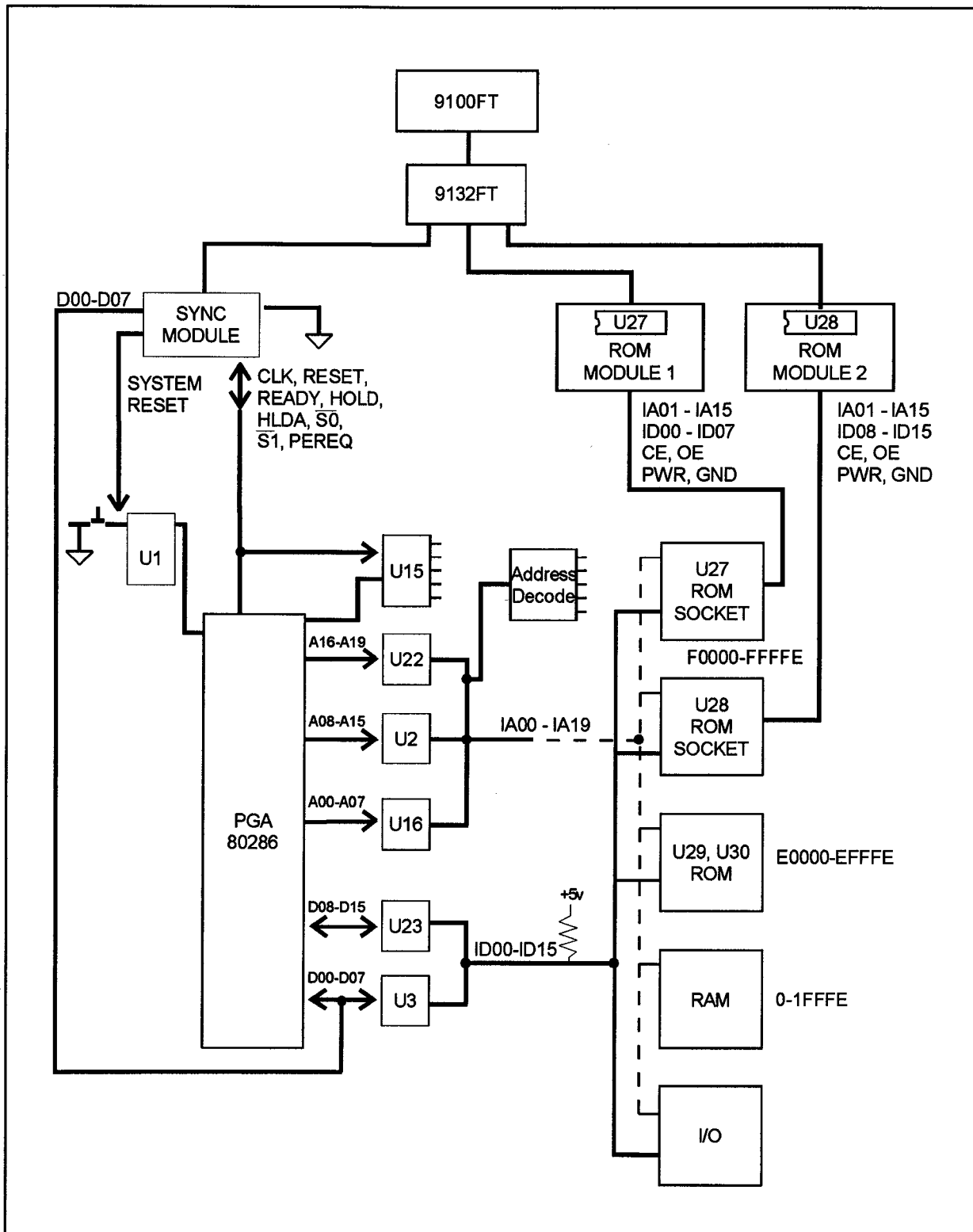


EXERCISE 4-15

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

Fault 1-2	Fault Messages
Functional Test 1:	
Functional Test 2:	
Functional Test 3:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	

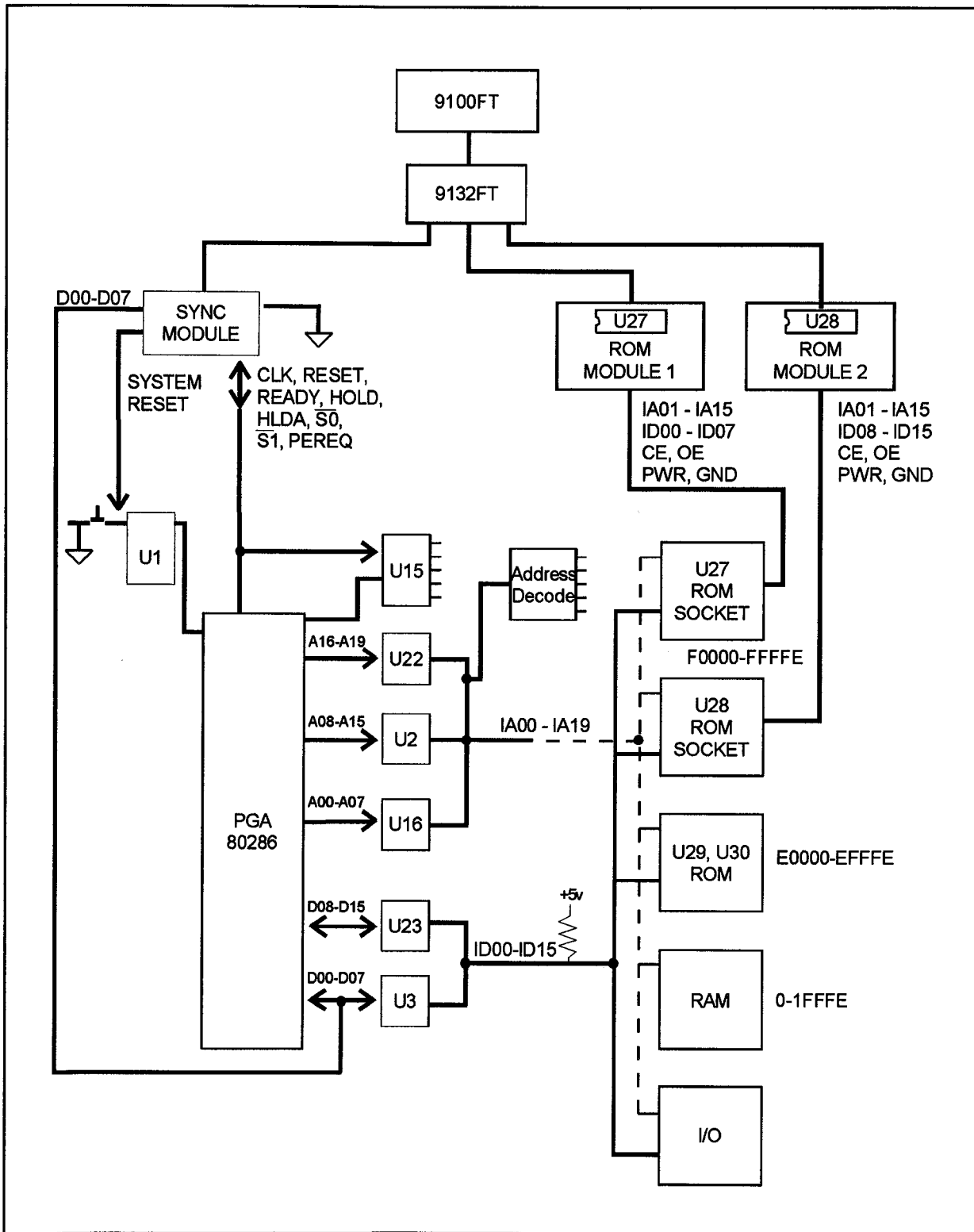
Fault 3-7	Fault Messages
Functional Test 1:	
Functional Test 2:	
Functional Test 3:	
<i>Stimulus:</i>	
<i>Sync Mode:</i>	
<i>Bad Node:</i>	



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 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 from emulation memory in the Pod.
- ③ Some UUT's may require the setup attributes 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.



PROBLEMS DUE TO A MARGINAL UUT

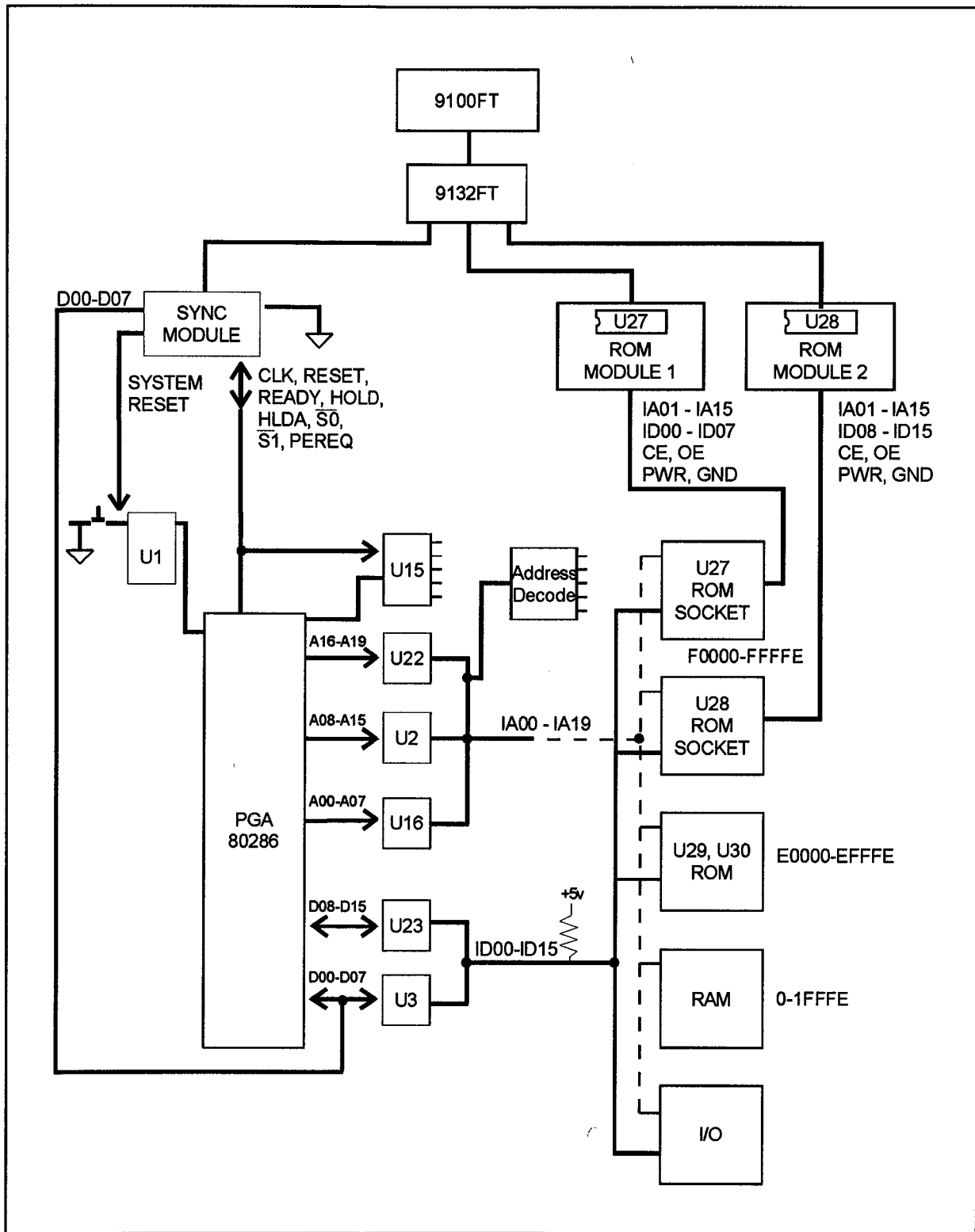
The Pod is designed to approximate, as closely as possible, the actual characteristics of the ROMs that it replaces in the UUT. However, the Pod does differ in some respects. In general, these differences tend to make marginal UUT problems more visible. A UUT may operate marginally with the UUT ROMs 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.



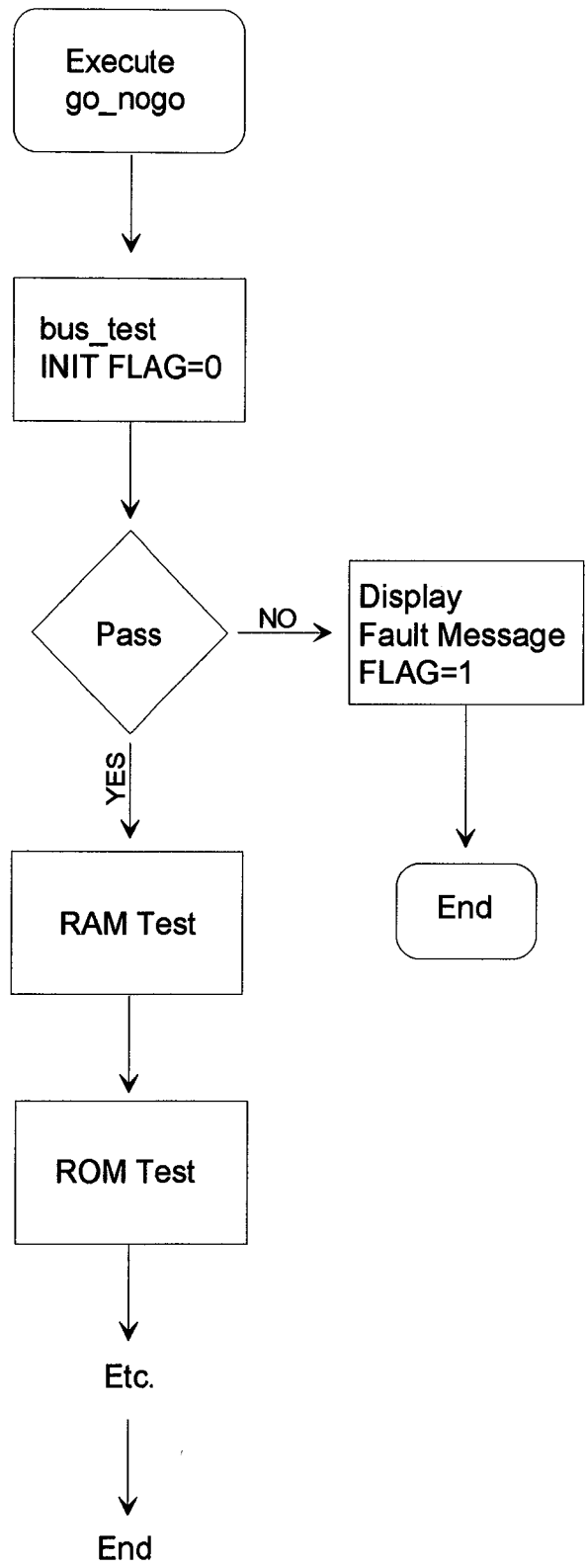
GO_NOGO PROGRAM

EXERCISE 4-16

- ① Write a GO_NOGO program to test the BUS, RAM, and ROM of the Demo Trainer Board.
- ② Verify the GO_NOGO program works properly with no faults on the UUT.
- ③ Have instructor put faults in the Demo board and use your program to find the faults.
- ④ If time permits add TL/1 support programs to expedite fault isolation.

Section 5

UFI: Memory Emulation



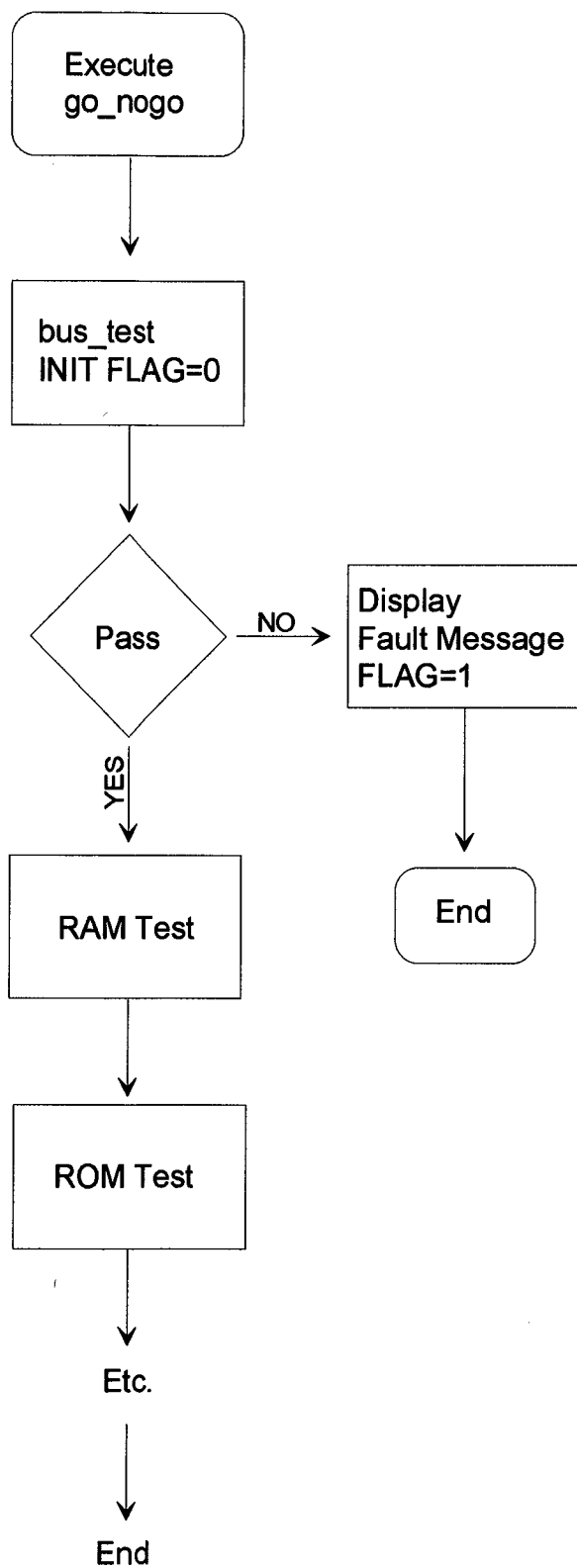
THE SOFTWARE SWITCH

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

Remember, the EFI and GFI algorithms 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 overcoming this situation is to use a *software switch*. The software switch will turn OFF any programs that cause loss of microprocessor control.

The programs on the following pages 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.

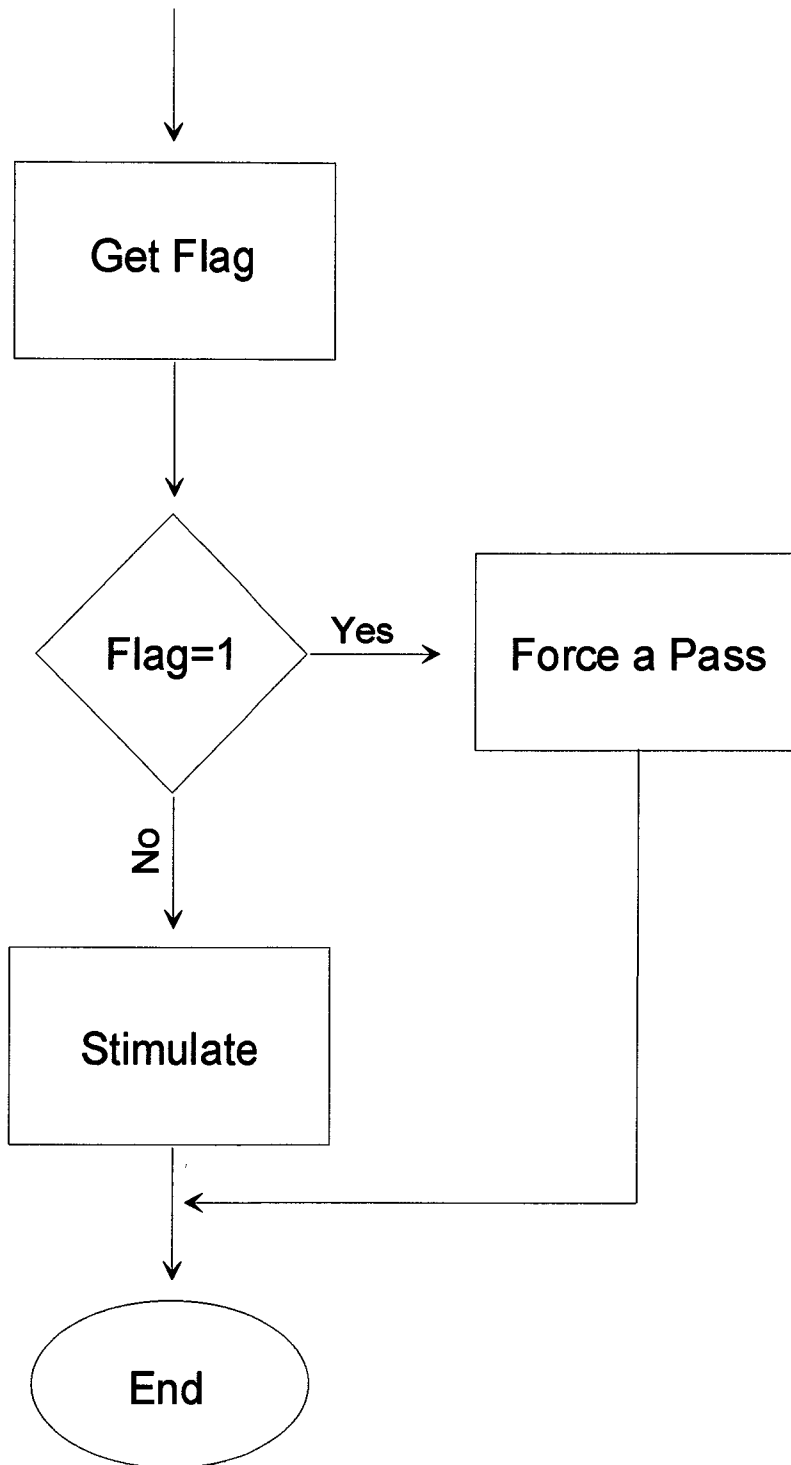


program bus_test

```

program bus_test
handle bus_data_low_tied(mask)
  declare
    string mask
    string cls = "\1B[2J"
    string curoff = "\1B[?25l"
    string line 1 = "\1B[1;1H"
    string line 3 = "\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, line 3, "==== 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

```

program rom0_data

```
program rom0_data
  handle
    fault
  end handle
  declare
    string x
    string 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
```

program k_addr

```
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 = "/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
```

program k_data_in

```
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 9100FT 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 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 (9100FT)

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, test 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 Star, 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 of 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 9100FT 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 9100FT 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 considerable 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, re-programmable 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 9100FT 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 sequences 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 9100FT to perform tests on a UUT.

TRI-STATE

Being in a high impedance state.

UUT (UNIT UNDER TEST)

The circuit board or system being tested by the (9100FT).

USERDISK

In the 9100FT, 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 changes 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 9100FT 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 its 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.

Appendix B

Technical Data Sheets

<u>TITLE</u>	<u>NUMBER</u>
9100A Series Digital Testers	A0266C
9132A Memory Interface Pod	A0321A
Microprocessor Pod Adapters	A0355B
Vector Output I/O Module	A0357A
9100A Synchronizing I/O Module and Probe with "Int" Mode	A0367A
9132A-68030 Processor Support Package	A0372A
9132A-64180 Processor Support Package	A0373A
9132A-68020 Processor Support Package	A0374A
9132A-80386 Processor Support Package	A0379A
9132A-80386SX Processor Support Package	A0380A
9132A-68000 Processor Support Package	A0384A
9132A-80486 Processor Support Package	A0385A
9132A-7810 Processor Support Package	A0386A
9132A-80286 Processor Support Package	A0387A
9100A-030 9010 to 9100FT Programming Translator	A0392A
IEEE 488 / Plus Option 9105A-015 / 9100A-015 UGK	A0393B
9132A-68HC11 Processor Support Package	A0406A
Using Multiple I/O Modules For Driving Patterns and Gathering Signatures	L0045A
Putting a Software Switch into your GFI Database	L0046A

*Appendix c***Application Notes**

<u>TITLE</u>	<u>NUMBER</u>
Determining the suitability of Digital Designs for testing with the 9132A Memory Interface Pod	B0186A
9100FT Testing of a CGA Card with the Vector I/O Module	B0192A
Developing Test Vectors for a Video Controller Chip	B0200A
Testing DMA Circuitry on the IBMPC-AT	A0350A
Adding IEEE-488 Instruments to the 9100FT Digital Test System	A0349A
9100FT Production Testing	L0030A

