9000A-80286H

INTERFACE POD

WITH *HyperTEST™*

Instruction Manual

P/N 844121
February 1988 Rev. 2 3/89

FLUKE

WARRANTY

COVERAGE

Fluke warrants the 9000A-80286H Interface Pod to be free from defects in material and workmanship under normal use and service for a period of one (1) year and the UUT cable assembly for ninety (90) days from the date of shipment. This warranty extends only to the original purchaser and does not apply to any product that has been misused, altered, or has been subjected to abnormal conditions of operation.

Fluke's obligations under this warranty is limited to repair or replacement of a product that is returned to an authorized Service Center within the warranty period, provided that we determine that the product is defective. If we determine that the failure has been caused by misuse, alteration, or abnormal conditions of operation, or if the warranty period has expired, we will repair the Pod and bill you for the reasonable repair cost.

SERVICE

If a failure occurs, send the product, postage prepaid, to the closest Service Center with a description of the difficulty. Repairs will be made or the product replaced, and it will be returned, transportation prepaid. Fluke assumes NO risk for damage in transit.

DISCLAIMER

THE FOREGOING WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS, OR ADEQUACY FOR ANY PARTICULAR PURPOSE OR USE. FLUKE SHALL NOT BE LIABLE FOR AND SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN CONTRACT, TORT, OR OTHERWISE.

GETTING ANSWERS AND ADVICE

To enhance your use of this Pod, Fluke will be happy to answer you questions about applications and use. Address all correspondence to: JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WASHINGTON 98206, ATTN: Sales Department. European customers should contact FLUKE (Holland) B.V., P.O. BOX 2269, 5600 CG, EINDHOVEN, THE NETHERLANDS.

JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WASHINGTON 98206

Table of Contents

(continued on page ii)

TABLE OF CONTENTS, *continued*

(continued on page iii)

TABLE OF CONTENTS, *continued*

(continued on page iv)

TABLE OF CONTENTS, *continued*

List of Tables

List of Figures

Section 1 Introduction

THE PURPOSE OF THE INTERFACE POD 1-1.

The 9000A-80286H Interface Pod allows you to use any Fluke 9000-Series Micro-System Troubleshooter or 9100-Series Digital Test System/Station to troubleshoot equipment that uses an 80286 microprocessor.

The 9000-Series Micro-System Troubleshooter and the 9100-Series Digital Test System/Station (referred to hereafter as the Mainframe) are used to service printed circuit boards, instruments, and systems that use microprocessors. The 9000A-80286H Interface Pod (referred to as the Pod) replaces the 80286 microprocessor in the unit under test (UUT) and serves both as an interface to allow the Mainframe access to components on the UUT and as an emulator of the UUT's microprocessor.

In normal Mainframe/Pod operation, the Pod adapts the general-purpose architecture of the Mainframe to the specific pin layout of the 80286 microprocessor. This allows the Mainframe to exercise each of the microprocessor's signals and provides complete access to devices on the UUT that are connected to the microprocessor's bus, while, at the same time, monitoring activity on the UUT. In this troubleshooting mode, the Pod typically carries out a UUT read or write operation, and the Mainframe presents the results to the user.

In the RUN UUT mode, the Pod's microprocessor is connected through buffers to the UUT to serve as a substitute microprocessor. The Pod adapts such microprocessor functions as status/control lines, interrupt handling, timing, and memory and I/O addressing.

The 9000A-80286H Pod includes *HyperTEST*™ functions, which test UUT memory much faster than the FAST or FULL memory tests. *HyperTEST* examines UUT memory with the same amount of fault coverage as the FAST memory tests. *HyperTEST* consists of HyperRAM for testing UUT RAM and HyperROM for testing UUT ROM.

NOTE

It is assumed that the user of this manual is familiar with the basic operation of one of the 9000-Series Micro-System Troubleshooters or the 9100-Series Digital Test System/Station.

PHYSICAL DESCRIPTION OF THE POD 1-2.

The Pod connects to the Mainframe through a round shielded cable, and connects to the UUT through a flexible circuit cable and plug that is inserted into the UUT's

* *HyperTEST is a trademark of John Fluke Mfg. Co., Inc.*

microprocessor socket. The UUT's microprocessor is removed from the UUT and is replaced by the Pod UUT cable plug.

The external features of the Pod are shown in Figure 1-1.

The Pod consists of a pair of printed circuit board assemblies mounted within a break-resistant case. A clock module is located near the end of the flexible circuit cable. The Pod contains the control 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.

An 80286 microprocessor in the Pod performs all the functions that are normally required by the UUT, and performs the Pod's functions as well. Figure 1-2 shows the communication between the Pod, the Mainframe, and the UUT.

The Mainframe supplies operating power for the Pod. The UUT provides the external clock signal required by the Pod, which allows the Pod to function at the designed operating speed of the UUT. (The clock module on the plug cable amplifies the clock signal to ensure that it is strong enough to drive both the cable and the Pod circuitry.)

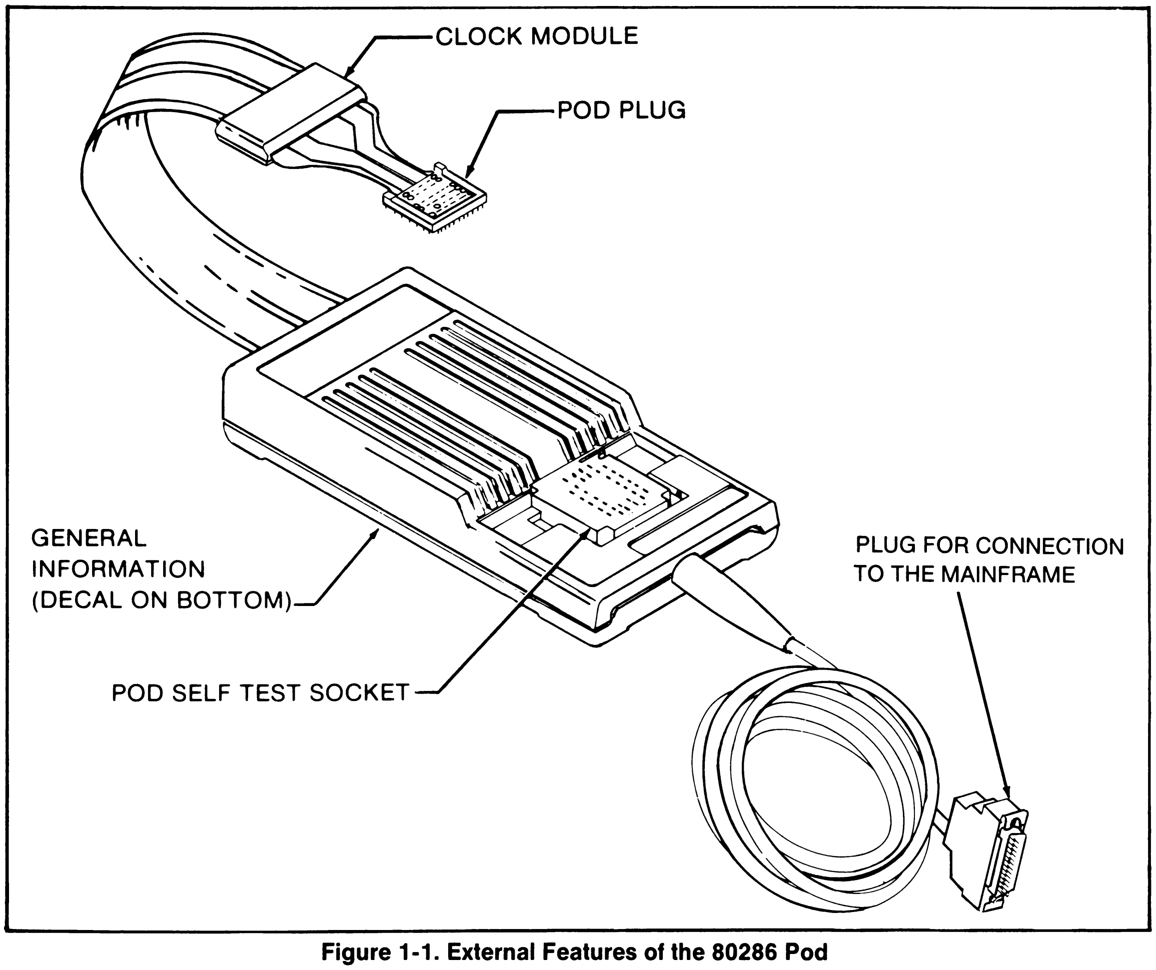

CLOCK MODULE
POD PLUG
GENERAL INFORMATION (DECAL ON BOTTOM)
PLUG FOR CONNECTION TO THE MAINFRAME
POD SELF TEST SOCKET

Figure 1-1. External Features of the 80286 Pod

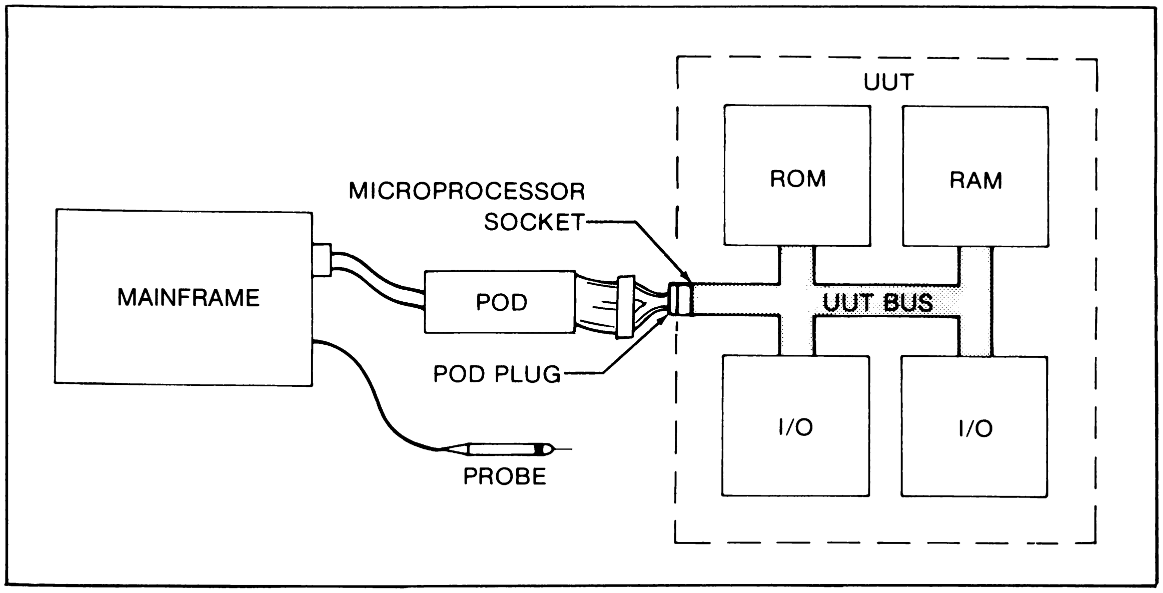


Figure 1-2. Communication Between the Mainframe, the Pod, and the UUT

Logic-level detection circuits on each line to the UUT detect bus shorts, stuck-high or stuck-low conditions, and any bus drive conflicts (two or more drivers attempting to drive the same bus line).

Over-voltage protection circuits on each line to the UUT guard against Pod damage that could result from the following:

- Incorrectly inserting the UUT cable plug into the UUT's microprocessor socket.
- UUT faults that place potentially damaging voltages on the lines to the UUT's microprocessor socket.

NOTE

The over-voltage protection circuits guard against voltages of +12V to -7V on any one pin of the Pod 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 rises above or 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 UUT cable plug is connected to the self-test socket during self test operation, which allows the Mainframe to investigate the Pod's internal functions.

CAUTION

To protect against damage to the Pod plug and cable and to prevent static damage to Pod components, insert the UUT cable plug into the self-test socket when the Pod is not in use.

POD SPECIFICATIONS 1-3.

Specifications for the Pod are listed in Table 1-1.

Table 1-1. 9000A-80286H Pod Specifications

ELECTRICAL PERFORMANCE	
Power Dissipation	10 watts max.
Maximum External Voltage	-7V to +12V (all pins)
	Voltages listed above are continuous and are referenced to ground.
MICROPROCESSOR SIGNALS*	
Clock input (pin 31):	
Input Low Voltage	0.8V max. at 3.2 mA
Input High Voltage	2.0V min. at -50 μA
All other signals:	
Input Low Voltage	0.8V max.
Input High Voltage	2.0V min.
Output Low Voltage	0.5V max. at rated current
Output High Voltage	2.4V min. at -400 μA
Tri-state Output Leakage Current	±0.02 mA typical, +0.1 to -0.2 mA max.
High Level Input Current	20 μA max.
Low Level Input Current ($\overline{\text{READY}}$)	-2.5 mA max.
Low Level Input Current (all others)	-500 μA max.
TIMING CHARACTERISTICS	
Maximum External Clock Frequency	25.0 MHz (CLK input) 12.5 MHz effective rate (PCLK)
Insertion Delays to 80286 Signals	
INPUT SIGNALS	15 ns typ.
OUTPUT SIGNALS	20 ns typ.
UUT POWER DETECTION	
Detection of Low Vcc Fault	Vcc < +4.5V
Detection of High Vcc Fault	Vcc > +5.5V
Pod Protection from UUT Low Power	Vcc < +3.4V**
GENERAL	
Size	5.7 cm H x 14.5 cm W x 27.1 cm L (2.2 in H x 5.7 in W x 10.7 in L)
Weight	1.5 kg (3.3 lbs)
Environment	
STORAGE	-40°C to +70°C, RH < 95%, non-condensing
OPERATING	0°C to +40°C, RH < 75%, non-condensing
Protection Class 3	Relates solely to insulation or grounding defined in IEC 348.

*Signals are specified as they appear at the UUT cable plug pins.
**Pod outputs set to high-impedance state.

USING THIS MANUAL 1-4.

This manual provides complete information for using the 80286H Pod, including installation and setup, operating, and troubleshooting instructions. The summary below explains briefly what kind of information is available in each of the sections:

Section 1	Contains the introduction and installation instructions for connecting the Pod to the Mainframe and the UUT. This section also contains a description of the built-in self test for ensuring that the Pod is functioning correctly.
Section 2	Describes how to use the Pod with the 9100-Series Digital Test System/Station. Explains how to initiate various Mainframe tests, then describes how to set various functions that are specific to your UUT.
Section 3	Describes how to use the Pod with the 9000-Series Micro-System Troubleshooter. Explains how to initiate various Mainframe tests, then describes how to set various functions that are specific to your UUT. Section 3 also contains quick tests and instructions for using the Pod with an oscilloscope.
Section 4	Describes the address structure of the Pod, and supplies information on detecting errors and masking errors. Section 4 also lists the 80286 signals and explains control and status lines.
Section 5	Contains theory of operation information that describes how the Pod works and gives a background for extended troubleshooting of the Pod itself, if necessary.
Section 6	Describes procedures for diagnosing and correcting failures in the Pod. If you have a Pod failure, use the procedures in Section 6 (with the schematics in Section 8) to help locate the problem.
Section 7	Contains parts lists and information to use when ordering replacement parts for your Pod.
Section 8	Contains schematic diagrams to use if it becomes necessary to troubleshoot an inoperative Pod.
Appendices	Contains miscellaneous material that may prove valuable when using the Pod.

GETTING STARTED 1-5.

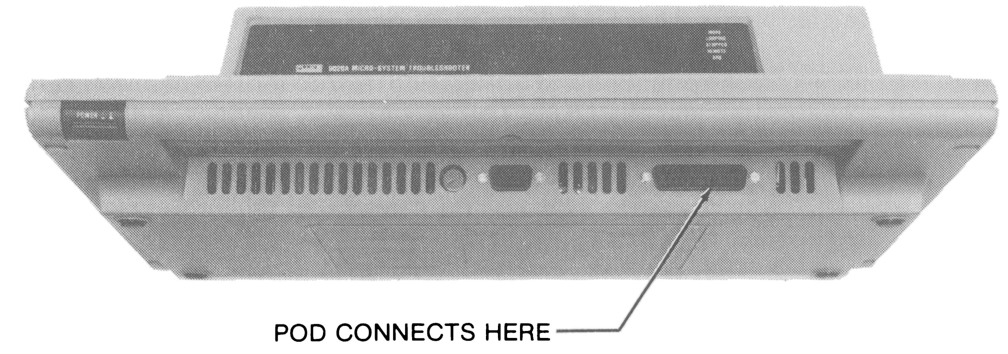
The following paragraphs contain directions for connecting the Pod to the Mainframe and your 80286-based UUT. Once the Pod is connected to the Mainframe, perform the built-in self test to ensure that the Pod is operating correctly. After the Pod has passed the self test, you can connect the Pod to the UUT and begin testing the UUT.

CONNECTING THE POD TO THE MAINFRAME 1-6.

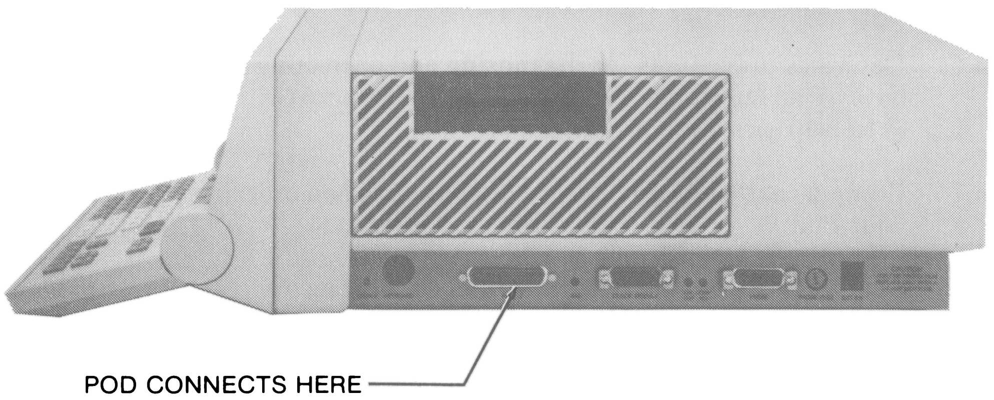
Before performing a Pod Self Test or using the Pod to troubleshoot a UUT, connect the Pod to the Mainframe as follows:

1. Check that the Mainframe is OFF.

2. Connect the Pod's round shielded cable to the Mainframe at the location shown in Figure 1-3. Secure the connector using the sliding collar.



9000-Series Micro-System Troubleshooter



9100-Series Digital Test System/Station

Figure 1-3. Connection of the Interface Pod to the Mainframe

PERFORMING THE POD SELF TEST 1-7.

To perform the built-in self test on the Pod, do the following steps:

1. Make sure that the Pod is connected properly to the Mainframe.
2. If the Pod is connected to a UUT, release the Pod UUT cable from the microprocessor socket.

CAUTION

Be sure the black ground clip lead on the UUT cable is disconnected when performing the Pod self test.

CAUTION

The Pod UUT cable is susceptible to damage from kinking or tearing, and is expensive to replace. Use caution in handling the cable.

3. Insert the Pod UUT cable into the Zero-Insertion Force (ZIF) self-test socket (as shown in Figure 1-4):

   a. Open the ZIF socket by moving the latch lever to the vertical position.

   b. If the Pod Plug is not already attached to a Pin-Grid Array (PGA) adapter, clip the Pod plug in place as follows:

      1. Insert the PGA adapter into the ZIF socket, taking care to align pin 1 in both.

      2. Secure the PGA adapter within the ZIF socket by pushing the latch lever down.

      3. After moving the wire bail out of the way, insert the Leadless Chip Carrier (LCC) plug that is on the end of the cable into the top of the PGA adapter. Insert the beveled corner first, pushing it back against the corner spring, then lower the rest of the plug into place and hold it with your finger.

      4. Place the retainer clip on top of the LCC plug by pushing the ends of it back into the cavities in the PGA adapter, then lowering it against the LCC plug. Hold the clip firmly in place with your fingers.

      5. Flip the wire bail over the tabs on the edge of the retaining clip to secure the assembly.

   c. If the LCC (the Pod plug) on the end of the Pod cable is already connected to a Pin-Grid Array adapter, then insert the pins of the PGA into the ZIF socket and secure the adapter in place by pushing down the latch lever.

4. Turn Mainframe power on.

5. Begin the Pod self test. On 9100-Series Mainframes, press the Main Menu key to obtain the display *MAIN: SELFTEST POD.* Then press the ENTER key. On 9000-Series Mainframes, run the Pod Bus Test as described in the appropriate Mainframe Operator Manual.

   When the Pod passes the self test, the Mainframe displays a message indicating the Pod is functioning correctly. If the Mainframe displays a message indicating the Pod has failed the self test, turn to Section 6 for information on diagnosing self-test failures. (Failure codes are also listed in Section 6.)

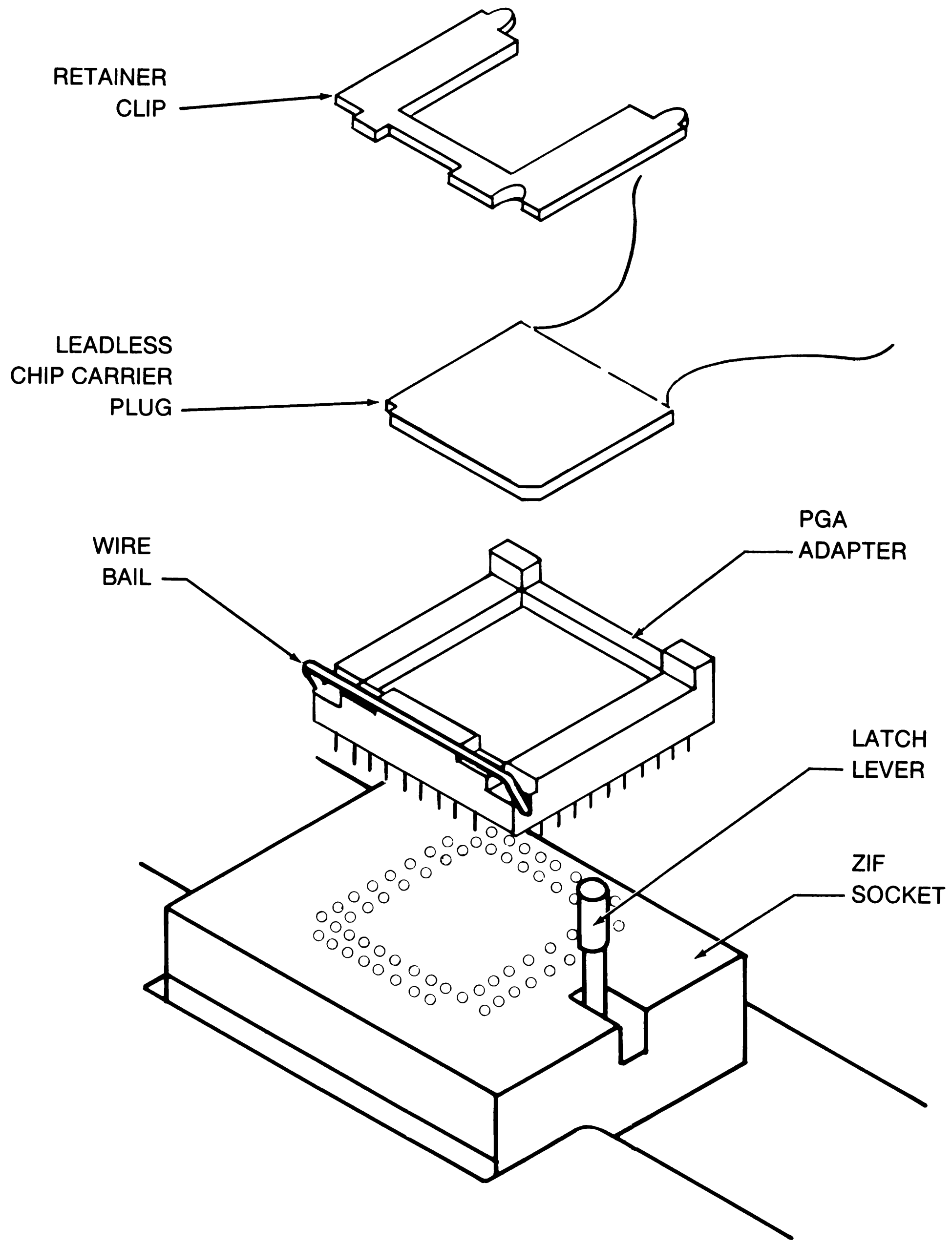


Figure 1-4. Inserting Pod Cable into Self Test Socket

CONNECTING THE POD TO THE UUT 1-8.

WARNING

TO PREVENT POSSIBLE HAZARDS TO THE OPERATOR OR DAMAGE TO THE UUT, DISCONNECT ALL HIGH-VOLTAGE POWER SUPPLIES, THERMAL ELEMENTS, MOTORS, OR MECHANICAL ACTUATORS THAT ARE CONTROLLED OR PROGRAMMED BY THE UUT MICROPROCESSOR BEFORE CONNECTING THE POD.

Connect the Pod to the UUT as follows:

1. Be sure that power is removed from the UUT.

2. Disconnect UUT analog outputs or potentially hazardous UUT peripheral devices as described in the previous warning.

3. If necessary, disassemble the UUT to gain access to the microprocessor socket. If the microprocessor is still in the socket, remove it.

4. If the Pod plug is inserted into the self-test socket, remove it as follows:

   a. If the UUT's microprocessor socket is a Leadless Chip Carrier, remove the LCC plug that is on the end of the Pod cable from the PGA socket by flipping the wire bail off the retainer clip, removing the retainer clip, and lifting the LCC plug free.

   b. If the UUT's microprocessor socket is a pin grid socket, remove the Pod plug with the PGA adapter attached by pulling up the latch lever of the ZIF socket and lifting the Pod plug free.

CAUTION

The Pod UUT cable is susceptible to damage from kinking or tearing, and is expensive to replace. Use caution in handling the cable.

5. Insert the Pod plug into the UUT's microprocessor socket and secure it (using the same means used to secure the microprocessor). Make sure that pin 1 of the Pod plug is aligned with pin 1 of the microprocessor socket. If the UUT uses a Leadless Chip Carrier socket, insert just the LCC Pod plug. If the UUT uses a pin-grid socket, insert the Pod plug into a pin-grid array adapter before attaching it to the UUT. See the self-test procedure above for instructions for attaching the pin-grid array adapter. If the UUT uses a soldered-in processor, see Appendix C of this manual, Testing UUTs With Soldered-in Microprocessors.

CAUTION

Do not operate the Pod upside down to change the orientation of the cable to the UUT. The internal cooling system of the Pod is designed to operate in the upright position.

6. Connect the UUT cable black wire to logic signal ground on the UUT at a point near the Pod plug.

7. Reassemble the UUT, using extender boards if necessary.

CAUTION

To prevent damage to the Pod, you must apply power to the Mainframe before turning UUT power on. This activates protection circuits within the Pod.

8. Apply power to the UUT.

Section 2
9100-Series Digital Test System/Station Operation

USING POD FUNCTIONS 2-1.

This section describes how to use the Pod functions to test components and circuitry on a UUT using the 9100-Series Digital Test System/Station. The following subjects are included:

Read and Write Operations	How to read and write to UUT memory.
Testing the UUT Bus	How to use the Mainframe and Pod to test the Bus circuitry on the UUT.
Testing the UUT RAM	How to use the Mainframe and Pod to test the RAM circuitry on the UUT.
Testing the UUT ROM	How to use the Mainframe and Pod to test the ROM circuitry on the UUT.
Testing Interrupt Circuitry	How to use the Pod to evaluate Interrupt circuitry on the UUT.
Using the RUN UUT Mode	How to exercise the UUT by using the Pod to emulate its microprocessor.
Using Breakpoints	How to set breakpoints to interrupt the UUT program at specific addresses.
Using the Overlay RAM	How to substitute known-good RAM in the Pod for RAM on the UUT.

You should be aware of the following notation conventions in this manual:

- For ease of reading, the two halves of the address are separated with a space in this manual (HHHH LLLL). However, do not try to enter addresses with a space. The Mainframe does not display addresses with a space.
- X denotes hex or binary digits, where the specific value may be any valid number. For example, the data value XX00 means that it is only important that the two least-significant digits are zero; the other two digits may be any value.

- Some of the examples show addresses in binary form. Address and data values are only entered and displayed on the Mainframe in hexadecimal form, so binary information must be converted to hex before using. For example, an address specification that is illustrated in binary form as 1111 0000 1100 0000 1010 0011 is entered into the Mainframe as F0 C0A3.

NOTE

Whenever a function change is made to the values stored in the 9100-Series Digital Test System/Station, the new values are generally retained until the Test Station is either reset or power is turned off. In this manual, all the example displays show the default values for the fields (the default values appear after system power-on or after pressing the RESET key). If you have stored new values in the fields, your display may vary from the examples shown. If the cursor on your Test Station is not placed in the field as described in the example displays, move the cursor to the right by pressing the right arrow key (→) or move the cursor to the left by pressing the left arrow key (←).

READ AND WRITE OPERATIONS 2-2.

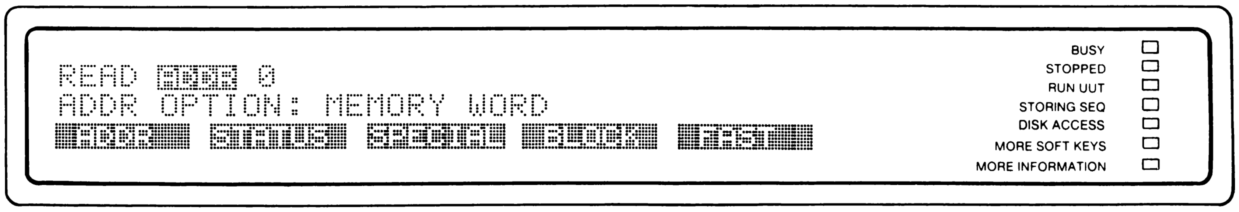
Read and write operations are accessed by pressing the READ or WRITE keys on the Mainframe keypad. Read allows you to obtain data from specific locations in UUT memory, to check status bits, and read data from the Pod special addresses. Write allows you to change the data at specific locations in UUT memory, to fill blocks of memory with data, to control the UUT by writing to the Pod's special addresses, and to pulse control bits.

NOTE

If a forcing line error occurs during a read or write operation, you must disable the forcing line indicated by the Mainframe display to continue testing the UUT. For information on disabling forcing lines, see the heading "User-Enableable Forcing Lines" further on in this section.

Read Operations 2-3.

To enter the Read menu of the Mainframe, press READ on the keypad. The Mainframe displays the following message:



```
READ ADDR 0
ADDR OPTION: MEMORY WORD
 ADDR   STATUS  SPECIAL  BLOCK   FAST
```

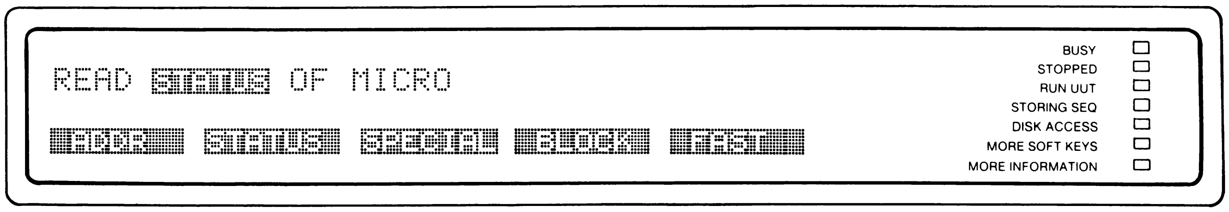
From this menu you can retrieve data from specific addresses, determine if the UUT processer is malfunctioning due to an incorrect status signal, and check the address and data lines.

READ AT UUT ADDRESS 2-4.

To read an address in UUT memory, press the READ key, move the cursor by pressing the right arrow key (→), press the ADDR (F1) softkey for the first field, press the right arrow key again, and enter the address. For UUT memory, the address should be between 00 0000 and FF FFFE (for a word access). The Mainframe limits entries to valid values either by accepting only as many digits as are valid, or by displaying an error message. Press ENTER on the keypad to read the data for the address you selected.

READ PROCESSOR STATUS 2-5.

To check the current state of the status lines, press the READ key and select the STATUS (F2) softkey. The Mainframe displays the following message:



```
READ STATUS OF MICRO

ADDR  STATUS  SPECIAL  BLOCK  FAST
```

Press ENTER on the keypad to read the status lines. The status bits are normally displayed as a 2-digit hex value (i.e., C1). This hex number represents the current state of the status line (i.e., if the line is high, the value of the bit in the hex number is set to 1; if the line is low, the bit is set to 0). For example, if $\overline{\text{READY}}$ is low (or Ready is true) and the rest of the lines are false (bits 7 and 6 high and bits 5, 4, 3, 2, and 1 low), the hex value for the status lines would be displayed as C0.

The status display result contains from 1 to 4 digits, depending on the position of bits set to 1. Leading zeroes are suppressed.

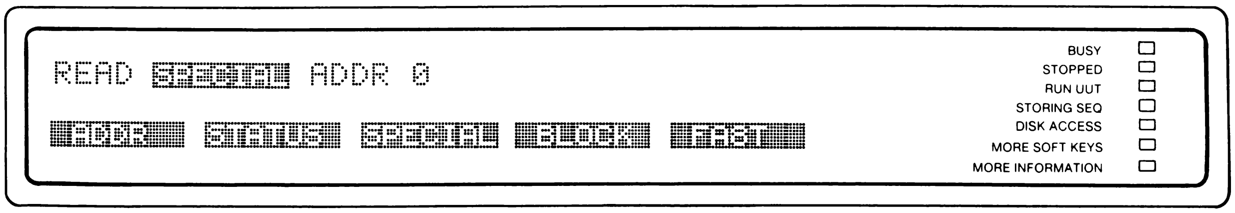
NOTE

Pod special addresses are only meaningful when troubleshooting the Pod or to emulate the 9000-Series interface between the Mainframe and the Pod. Reads to special addresses are not needed in normal 9100-Series operation.

For more information on status and pseudo-status lines, see Section 4 of this manual.

READ POD SPECIAL ADDRESS 2-6.

To read the contents of the Pod's special addresses, press the READ key and select the SPECIAL (F3) softkey. The Mainframe displays the following message:



```
READ SPECIAL ADDR 0

ADDR  STATUS  SPECIAL  BLOCK  FAST
```

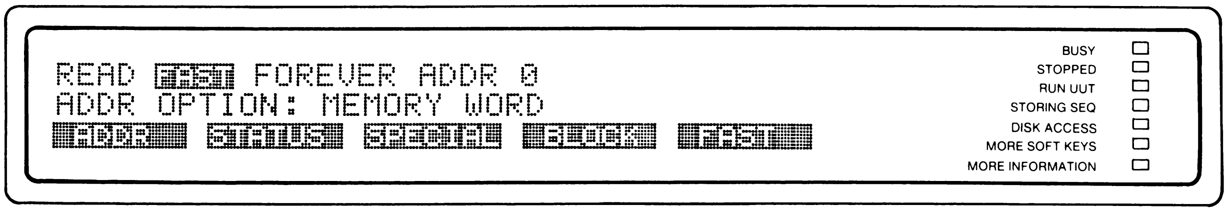
Press the right arrow key (→) to move the cursor, and enter the address you want to read. Special addresses can be entered as F000 XXXX, where F000 indicates a special address, and XXXX is the location of the special Pod function. Press ENTER on the keypad. The data from the special address is displayed as a 4-digit hex value (with leading zeros suppressed).

READ BLOCK 2-7.

The BLOCK (F4) function copies UUT memory data to a Mainframe disk file. For information on this function, see the 9100-Series Technical User's Manual.

REPEATED READS 2-8.

The Pod allows you to perform repeated reads at an address (for example, to check the accuracy of the address or data bus lines with an oscilloscope). Press the READ key to enter the Read menu. Select the FAST (F5) softkey. The Mainframe displays the following message:

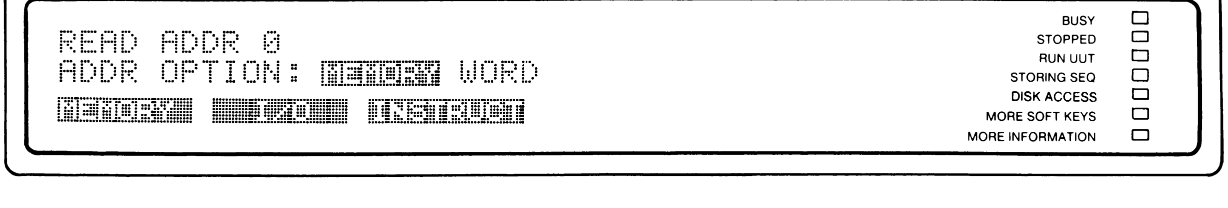


```
READ FAST FOREVER ADDR 0
ADDR OPTION: MEMORY WORD
ADDR   STATUS   SPECIAL   BLOCK   FAST
```

Press the right arrow key (→) to move the cursor, and enter the address you want to read. Press ENTER on the keypad. The Pod continuously reads the address you specify until you press the STOP key or reset the Mainframe. Data that is read during this operation is not displayed.

READ ADDRESS OPTIONS 2-9.

The second line of the Read menu is the Address Option function. To change the address option, move the cursor to the second line of the Read menu by pressing the down arrow (↓) key. The Mainframe displays the following message:



```
READ ADDR 0
ADDR OPTION: MEMORY WORD
MEMORY   I/O   INSTRUCT
```

After you choose the address option, press the ENTER key. The Mainframe then performs the Read operation shown on the first line of the menu.

The address option may be changed to one of five different settings.

- Memory word
- Memory byte
- I/O word
- I/O byte
- Instruction

The memory word option accesses the UUT memory at any even word boundary. To select the memory word option, press the MEMORY (F1) softkey, move the cursor by pressing the right arrow key (→), then press WORD (F1).

The memory byte option accesses any byte in the UUT memory. To select the memory byte option, press the MEMORY (F1) softkey, move the cursor by pressing the right arrow key (→), then press BYTE (F2).

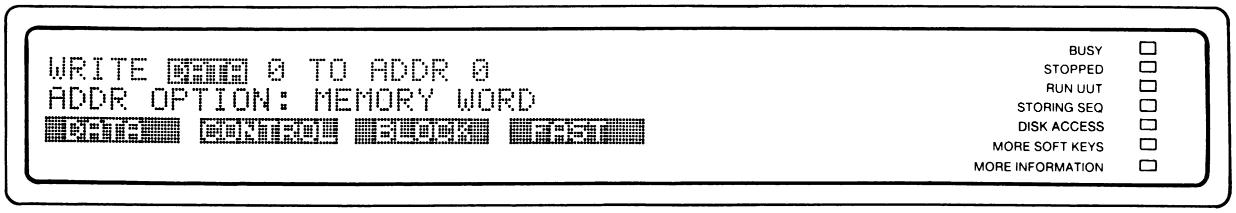
The I/O word option reads a word of data from an I/O port on the UUT. The port address range for the I/O word option is 0000 to FFFE. To select the I/O word option, press the I/O (F2) softkey, move the cursor by pressing the right arrow key (→), then press WORD (F1).

The I/O byte option reads a byte of data from an I/O port on the UUT. The port address range for the I/O byte is 0000 to FFFF. To select the I/O byte option, press the I/O (F2) softkey, move the cursor by pressing the right arrow key (→), then press BYTE (F2).

The Instruction option returns two executable opcode bytes from the even word address specified in the read command. Instruction space differs from memory space in that the COD/$\overline{\text{INTA}}$ control line is high when accessing instruction space, and low for memory space. The Instruction option selects an address from 0000 to FF FFFE. To select the Instruction option, press the INSTRUCT (F3) softkey.

Write Operations 2-10.

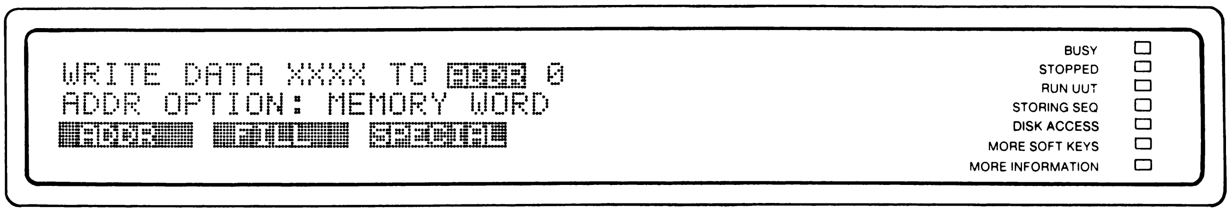
To enter the Write menu of the Mainframe, press WRITE on the keypad. The Mainframe displays the following message:


```
WRITE DATA 0 TO ADDR 0
ADDR OPTION: MEMORY WORD
DATA  CONTROL  BLOCK  FAST
```

From the write menu you can enter data in a specific address, insert data into blocks of address space (for example, to test video memory), change the state of the processor control lines, and check the accuracy of the address and data lines.

WRITE UUT ADDRESS 2-11.

To write data to an address in UUT memory, press the WRITE key, select the DATA (F1) softkey for the first field, move the cursor by pressing the right arrow key (→), and enter the data to be written to the address. Move the cursor to the next field by pressing the right arrow key. The Mainframe displays the following message:



```
WRITE DATA XXXX TO ADDR 0
ADDR OPTION: MEMORY WORD
ADDR   FILL   SPECIAL
```

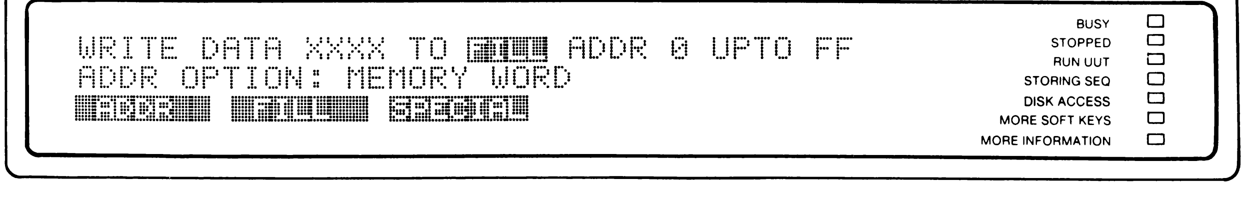
NOTE

XXXX is the data you entered.

Press the ADDR (F1) softkey, move the cursor by pressing the right arrow key again, and select the address where the data is to be written. For UUT memory, the address should be between 00 0000 and FF FFFE (for a word access). Press ENTER to write to the address you selected.

FILL MEMORY AREA 2-12.

To fill an area of memory with data, press the WRITE key, select DATA (F1) for the leftmost field in the Write menu, move the cursor to the next field and enter the data to be written. Move the cursor to the next field and enter FILL (F2). The Mainframe displays the following message:

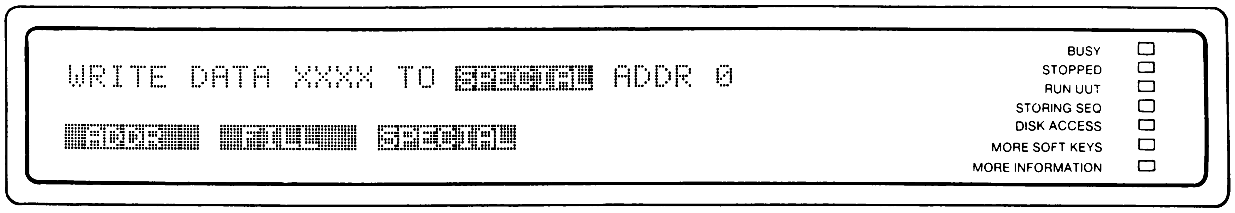


```
WRITE DATA XXXX TO FILL ADDR 0 UPTO FF
ADDR OPTION: MEMORY WORD
ADDR   FILL   SPECIAL
```

Move the cursor by pressing the right arrow key. Select the starting address and enter it into this field. Move the cursor to the next field and enter the ending address of the fill. Press ENTER on the keypad to begin filling memory.

WRITE POD SPECIAL ADDRESS 2-13.

To write data to a special address, press the WRITE key, select DATA (F1) for the leftmost field in the write menu, move the cursor to the next field and enter the data to be written. Move the cursor to the next field and enter SPECIAL (F3). The Mainframe displays the following message:


WRITE DATA XXXX TO SPECIAL ADDR 0

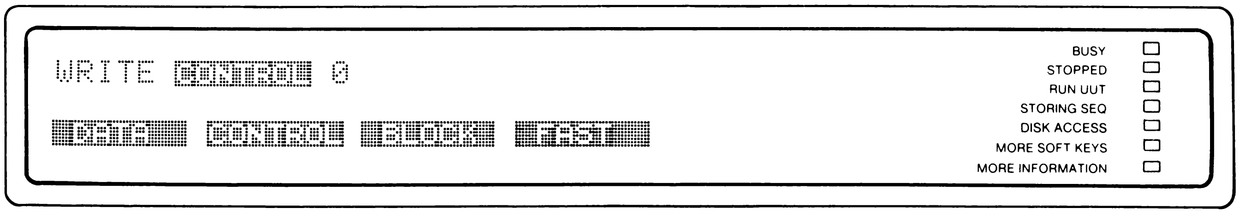
Move the cursor to the next field and enter the address you want to read. Special addresses can be entered as F000 XXXX, where F000 indicates a special address, and XXXX is the location of the special Pod function. Press ENTER on the keypad.

NOTE

Pod special addresses are only meaningful when troubleshooting the Pod or to emulate the 9000-Series interface between the Mainframe and the Pod. Writes to special addresses are not needed in normal 9100-Series operation.

WRITE PROCESSOR CONTROL 2-14.

To exercise the writable control lines, press the WRITE key and select the CONTROL (F2) softkey. The Mainframe displays the following message:


WRITE CONTROL 0

Move the cursor by pressing the right arrow key (→). Enter the number corresponding to the control lines to be activated.

NOTE

Two of the writable control lines are active low (bit 0, $\overline{\text{LOCK}}$ *and bit 2,* $\overline{\text{PEACK}}$*), and one is active high (bit 1, HLDA). To drive the three lines to their active states, enter 2 (binary 10) in the rightmost field of the WRITE CONTROL menu.*

NOTE

The Write Control function sets a line high or low for only one UUT access, just long enough to verify that the line can be driven. Following the UUT access, the lines return to their inactive state.

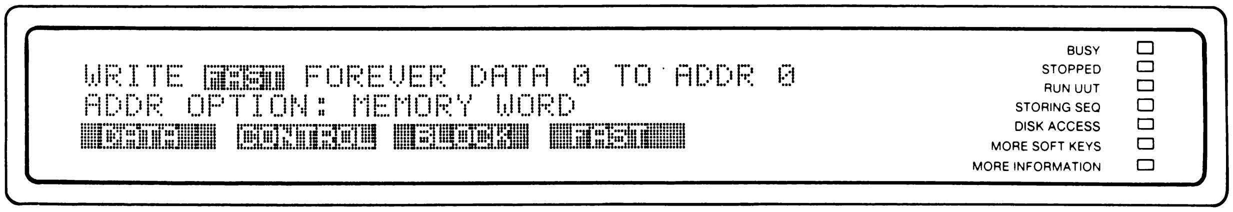
For more information on the control lines, see the heading "Control Lines" in Section 4 of this manual.

WRITE BLOCK 2-15.

The BLOCK (F3) function copies data from a Mainframe disk file to UUT memory. For information on this function, see the 9100-Series Technical User's Manual.

REPEATED WRITES 2-16.

The Pod allows you to perform repeated writes to an address (to check the accuracy of the address or data bus lines with an oscilloscope, for example). Press the WRITE key and select the FAST (F4) softkey. The Mainframe displays the following message:



```
WRITE FAST FOREVER DATA 0 TO ADDR 0
ADDR OPTION: MEMORY WORD
 DATA   CONTROL   BLOCK   FAST
```

Press the right arrow key (→) to move the cursor, and enter the data to be sent to the specified address. Press the right arrow key and enter the address you want to write to. Press ENTER on the keypad. The Pod continuously writes the data to the address you specify until you press the STOP key or reset the Mainframe.

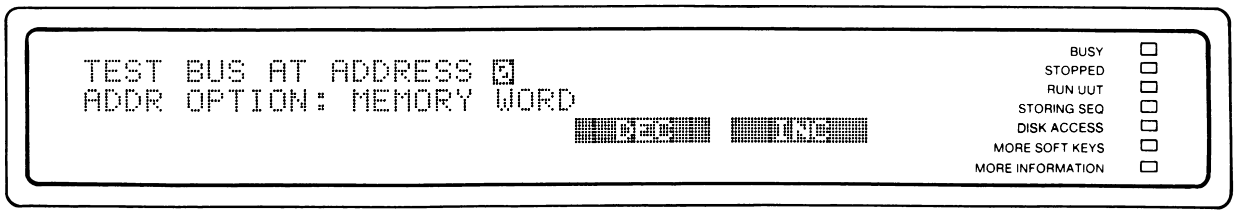
WRITE ADDRESS OPTIONS 2-17.

The address option line of the Write function operates the same as the address option line of the Read function (except instruction space, which is read only and may not be specified in a write operation). For more information, see the heading "Read Address Options" earlier in this section.

TESTING THE UUT BUS 2-18.

Bus Test is a test of the electrical integrity of UUT control, address, and data buses connected directly to the processor. Bus Test identifies control lines that are tied high or tied low, as well as address lines that are tied high, tied low, or tied together, and data lines that are tied high, tied low, or tied together.

To specify the Bus Test, press the BUS key on the Mainframe keypad. As soon as the key is pressed, the Mainframe displays the following message:



```
TEST BUS AT ADDRESS 0
ADDR OPTION: MEMORY WORD
                             DEC      INC
```

Bus Test includes several write operations that determine if any data lines are not drivable. The address that is specified for Bus Test is used for this portion of the test. To change the address of the Bus Test, either press the DEC function key (F4) or the INC function key (F5), or enter the address on the Mainframe keypad. To begin the test, press the ENTER key. The BUSY indicator on the right side of the Mainframe display shows the test is in progress. When the UUT passes the Bus Test, the BUSY light goes out. If the UUT fails the Bus Test, an error message appears on the second line of the Mainframe display.

The address option line of the Bus Test operates the same as the address option line of the Read function (except instruction space, which is read only and does not function with the Bus Test). For more information, see the heading "Read Address Options" earlier in this section.

For more information on the Mainframe Bus Test, see the 9100-Series Technical User's Manual.

TESTING THE UUT RAM 2-19.

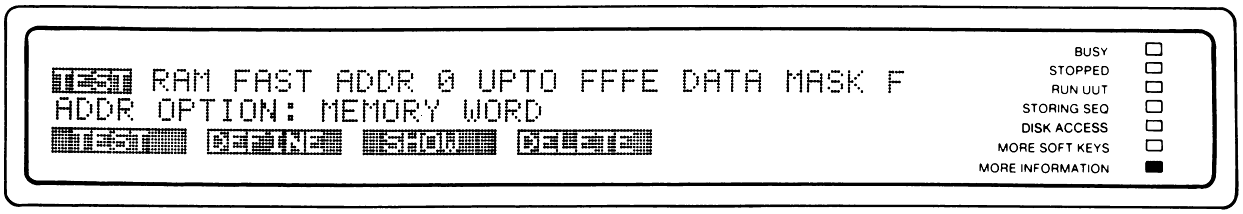
To ensure that all RAM failures are identified and yet optimize test times, the Mainframe provides the HyperRAM, RAM Fast, and RAM Full tests for the UUT RAM. Hyper RAM and RAM Fast are shorter and faster tests than RAM Full.

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

RAM Fast Test 2-20.

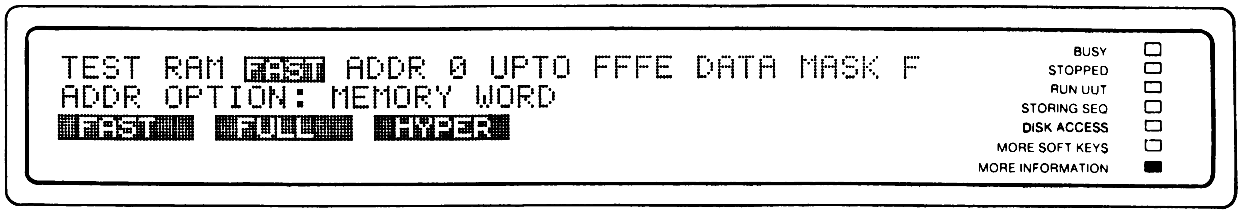
RAM Fast is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable. The following steps describe how to select the RAM Fast test.

1. Press the RAM key on the Mainframe keypad. The Mainframe displays the RAM Test menu:



```
TEST RAM FAST ADDR 0 UPTO FFFE DATA MASK F
ADDR OPTION: MEMORY WORD
TEST  DEFINE  SHOW  DELETE
```

2. Press the right arrow key (→) on the Mainframe keypad once to move the cursor to the Fast/Full/Hyper field on the display. If the Fast/Full/Hyper field is already set to Fast, skip the rest of this step and proceed with Step 3. To select the RAM Fast test, press F1 on the Mainframe keypad. The Mainframe displays the following menu:



```
TEST RAM FAST ADDR 0 UPTO FFFE DATA MASK F
ADDR OPTION: MEMORY WORD
 FAST   FULL   HYPER
```

3. Press the right arrow key once to move the cursor to the Addr/Ref field. Select ADDR by pressing F1 on the Mainframe keypad.

4. Press the right arrow key again to move the cursor to the starting address field. Key in the starting address.

5. Press the right arrow key again to move the cursor to the ending address field. Key in the ending address.

6. Press the right arrow key again to move the cursor to the data mask field. The data mask field allows you to select significant data bits that you want to test. For example, to test a single bit of data (such as 40) in a byte address, enter 40 into the data mask field. To test all the data in a word address, enter FFFF into the field. To test the most significant byte of a word address, enter FF00 into the field.

7. Press the right arrow key again to move the cursor to the address step field. Key in the step number. For example, if you are testing byte addresses, change the address step setting to 1 so each byte address is checked. If you are testing word addresses, change the setting to 2.

8. Press the right arrow key again to move the cursor to the delay field. This field contains the delay (in milliseconds) between the end of a write pass to RAM and a subsequent read from the RAM. The delay checks the refresh on dynamic RAMs. The delay field has a range of 0 to 99999 (the default is 250).

9. Press the right arrow key again to move the cursor to the seed field. The seed value determines how the data is generated. If seed is zero, the sequence of random data words is different each time the test is invoked. If seed is not zero, for a given seed value, the sequence of random data words is the same for each test of the memory. The default seed of "0" is sufficient for almost all RAM tests.

10. After all the correct information is entered into the fields, press the ENTER key to start the RAM Fast test.

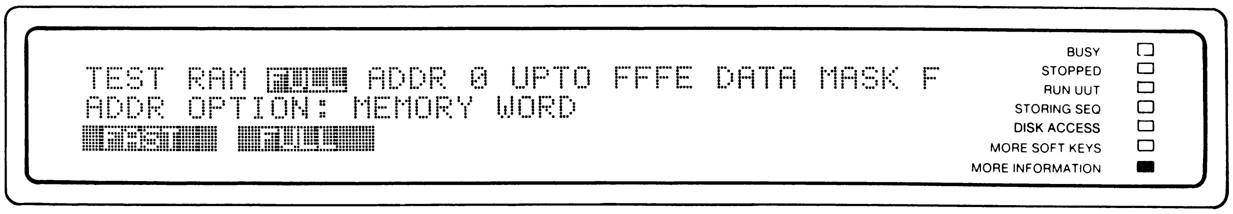
RAM Fast is executed on the address block specified. When the UUT passes RAM Fast, the BUSY light goes out. If the UUT fails RAM Fast, an error message appears on the second line of the Mainframe display.

RAM Full Test 2-21.

The RAM Full test is the most comprehensive RAM test algorithm available in the 80286H 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. The following steps describe how to select the RAM Full test.

1. Press the RAM TEST key on the Mainframe keypad. The Mainframe displays the RAM Test menu.

2. Press the right arrow key (→) on the Mainframe keypad once to move the cursor to the Fast/ Full/ Hyper field on the display. If the Fast/ Full/ Hyper field is already set to Full, skip the rest of this step and proceed with Step 3. To select the RAM Full test, press F2 on the Mainframe keypad. The Mainframe displays the following RAM Full test menu:



```
TEST RAM FULL ADDR 0 UPTO FFFE DATA MASK F
ADDR OPTION: MEMORY WORD
FAST  FULL
```

3. Press the right arrow key once to move the cursor to the Addr/ Ref field. Select ADDR by pressing F1 on the Mainframe keypad.

4. Press the right arrow key again to move the cursor to the starting address field. Key in the starting address.

5. Press the right arrow key again to move the cursor to the ending address field. Key in the ending address.

6. Press the right arrow key again to move the cursor to the data mask field. The data mask field allows you to select significant data bits that you want to test. For example, to test a single bit of data (such as 40) in a byte address, enter 40 into the data mask field. To test all the data in a word address, enter FFFF into the field. To test the most significant byte of a word address, enter FF00 into the field.

7. Press the right arrow key again to move the cursor to the address step field. Key in the step number. For example, if you are testing byte addresses, change the address step setting to 1 so each byte address is checked. If you are testing word addresses, change the setting to 2.

8. Press the right arrow key again to move the cursor to the delay field. This field contains the delay (in milliseconds) between the end of a write pass to RAM and a subsequent read from the RAM. The delay checks the refresh on dynamic RAMs. The delay field has a range of 0 to 99999 (the default is 250).

9. Press the right arrow key again to move the cursor to the seed field. The seed value determines how the data is generated. If seed is zero, the sequence of random data words is different each time the test is invoked. If seed is not zero, for a given seed value, the sequence of random data words is the same for each test of the memory. The default seed of "0" is sufficient for almost all RAM tests.

10. The last selection in the RAM Full menu is COUPLING ON/OFF (for more information on the coupling function, see the 9100-Series Technical User's Manual).

11. After all the correct information is entered into the fields, press the ENTER key to start the RAM Full test.

RAM Full is performed on the address block specified. When the UUT passes RAM Full, the BUSY light goes out. If the UUT fails RAM Full, an error message appears on the second line of the Mainframe display.

For more information on the Mainframe RAM Fast and RAM Full tests, see the 9100-Series Technical User's Manual.

HyperRAM Test 2-22.

HyperRAM is the fastest RAM test algorithm available in the 80286H Pod. It is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable.

NOTE

Data located in Overlay RAM may be changed by the HyperRAM test.

NOTE

The default configuration for the Overlay RAM $\overline{READY}$ source has $\overline{READY}$ generated by the UUT. If synchronization problems occur during the HyperRAM test, the $\overline{READY}$ source may need to be changed. For more information, see the heading "Setting the Source of $\overline{READY}$ for Overlay RAM" further on in this section.

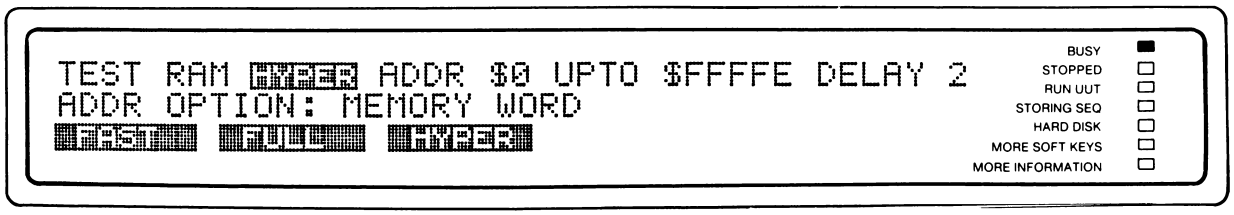
NOTE

HyperTEST functions are sensitive to the state of the forcing and interrupt status lines listed in Table 4-3. If the UUT asserts these signals during a HyperTEST, the test may abort. If a HyperTEST aborts, try to prevent the UUT from asserting the forcing signal or use an alternate memory test function.

The following steps describe how to select the HyperRAM test.

1. Press the RAM TEST key on the Mainframe keypad. The Mainframe displays the RAM Test menu.

2. Press the right arrow key (→) on the Mainframe keypad once to move the cursor to the Fast/Full/Hyper field on the display. To select the HyperRAM test, press F3 on the Mainframe keypad and press ENTER. The Mainframe displays the following HyperRAM test menu:



```
TEST RAM HYPER ADDR $0 UPTO $FFFFE DELAY 2
ADDR OPTION: MEMORY WORD
```

3. Press the right arrow key again to move the cursor to the starting address field. Key in the starting address. The valid address range is 0 to FF DFFC (word space) or FF DFFE (byte space).

4. Press the right arrow key again to move the cursor to the ending address field. Key in the ending address. The address limit is FF DFFE (word space) or FF DFFF (byte space).

5. Press the right arrow key again to move the cursor to the delay field. This field contains the delay between the end of a write pass to RAM and a subsequent read from the RAM. The delay checks the refresh on dynamic RAMs. The delay field has a range of 0 to 65535 (the default is 250). Each unit of delay roughly equals 10,000 periods of the effective processor clock signal. This means that 1 unit of delay approximates 1 millisecond for an effective UUT clock of 10 MHz with no wait states.

6. Press the right arrow key again to move the cursor to the seed field. The seed value determines how the data is generated. If seed is zero, the sequence of random data words is different each time the test is invoked. If seed is not zero, for a given seed value, the sequence of random data words is the same for each test of the memory. The default seed of "0" is sufficient for almost all RAM tests.

7. After all the correct information is entered into the fields, press the ENTER key to start the HyperRAM test.

HyperRAM is executed on the address block specified. When the UUT passes HyperRAM, the BUSY light goes out. If the UUT fails HyperRAM, an error message appears on the second line of the Mainframe display.

NOTE

Certain RAM test fields that are programmable for other RAM test methods are not shown for the HyperRAM test because they are fixed. The data mask field is fixed to check all data bits. The address step is fixed at 2 when the address option is set to word and at 1 when the address option is set to byte.

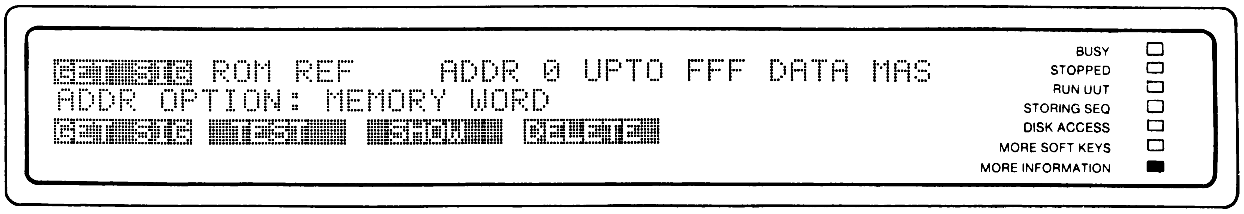
TESTING THE UUT ROM 2-23.

ROM Test computes the ROM signature for the user-specified address block and compares the signature to the previously stored, user-specified signature. A signature is an algorithmic compression (cyclic redundancy check) of a digital bit stream into a four-digit hexadecimal number. Applying the ROM Test algorithm over a range of addresses produces a signature that characterizes that range. The idea of using signatures is to obtain them from data in a known-good UUT, and then compare the good values with values obtained from the suspect UUT.

To obtain the ROM signature, perform the following procedure:

1. Plug the Pod into a known good board.

2. Select the ROM Test by pressing ROM on the Mainframe keypad. The Mainframe displays the ROM signature gathering menu:



```
GET SIG ROM REF     ADDR 0 UPTO FFF DATA MAS
ADDR OPTION: MEMORY WORD
GET SIG   TEST   SHOW   DELETE
```

3. Press the right arrow key (→) to move the cursor to the reference field. Use this field to name the signature file you are storing. The 9100A Mainframe allows you to store multiple signatures. For more information on storing multiple signatures, see the 9100-Series Technical User's Guide.

4. Press the right arrow key to move the cursor to the address field. Enter the first ROM address in this field. Press the right arrow key again and enter the last ROM address in this field.

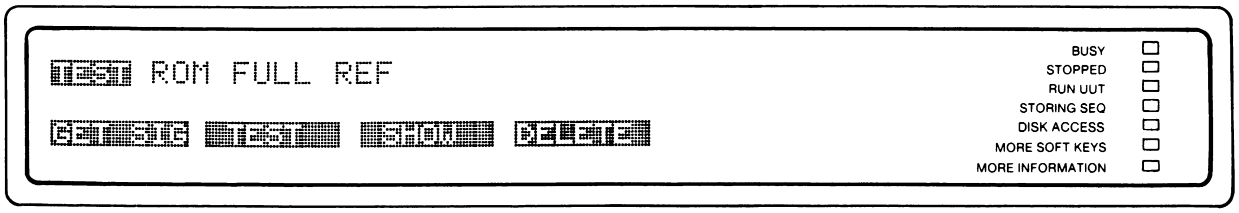
5. Press the right arrow key again to move the cursor to the data mask field. The data mask field allows you to mask off bits of data you do not want to test. For example, to test a single byte of data in a byte address, enter FF into the data mask field. To test all the data in a word address, enter FFFF into the field. To test the most significant byte of a word address, enter FF00 into the field.

6. Press the right arrow key again to move the cursor to the address step field. Key in the step number. For example, if you are testing byte addresses, change the address step setting to 1 so each byte address is checked. If you are testing word addresses, change the setting to 2.

7. Press ENTER to begin the signature gathering routine.

After the BUSY indicator on the right side of the Mainframe display goes out, the bottom line of the display shows the stored signature value.

A ROM Test can be performed after the signature is obtained. Select the ROM Test by pressing ROM on the Mainframe keypad. Move the cursor to the leftmost field using the left arrow key (←). Press F2 on the Mainframe keypad to select the ROM test. The Mainframe now displays the ROM Test menu:



```
TEST ROM FULL REF

GET SIG   TEST   SHOW   DELETE
```

To optimize test times, the Mainframe provides the HyperROM and ROM Full tests for UUT ROM. The HyperROM test is much faster than the ROM Full test.

NOTE

Data located in Overlay RAM may be changed by the HyperROM test.

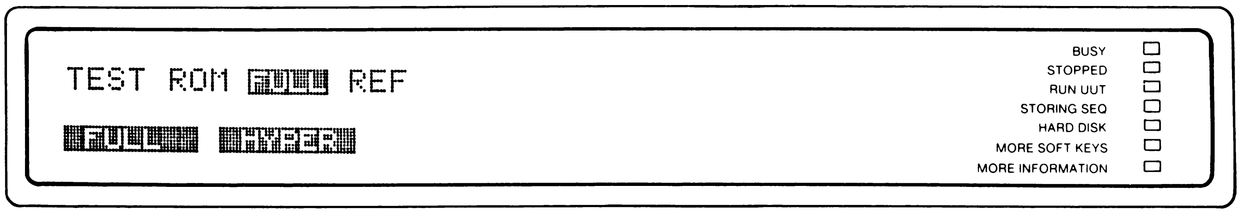
NOTE

The default configuration for the Overlay RAM $\overline{READY}$ source has $\overline{READY}$ generated by the UUT. If synchronization problems occur during the HyperROM test, the $\overline{READY}$ source may need to be changed. For more information, see the heading "Setting the Source of $\overline{READY}$ for Overlay RAM" in this section.

NOTE

HyperTEST functions are sensitive to the state of the forcing and interrupt status lines listed in Table 4-3. If the UUT asserts these signals during a HyperTEST, the test may abort. If a HyperTEST aborts, try to prevent the UUT from asserting the forcing signal or use an alternate memory test function.

Press the right arrow key (→) to move the cursor to the Full/Hyper field.



```
TEST ROM FULL REF

FULL    HYPER
```

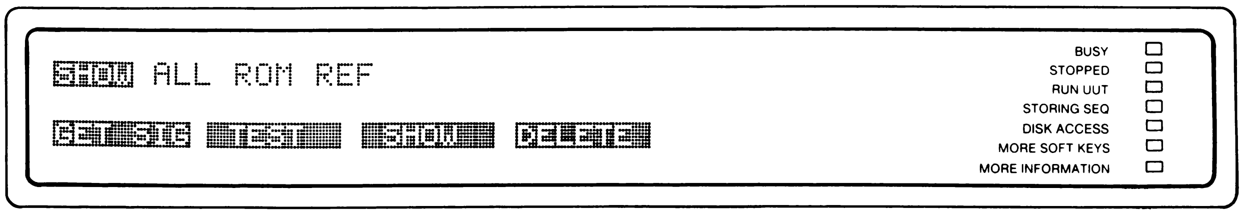
Select F1 to run the ROM Full test. Select F2 to run the HyperROM test. In a functional UUT, the ROM test signatures are the same for the ROM Full test and the HyperROM test.

For the ROM Full test, the starting address range is 0 to FF FFFC (word space) or 0 to FF FFFE (byte space). The ending address range is 2 to FF FFFE (word space) or 1 to FF FFFF (byte space).

For the HyperROM test, the starting address range is 2000 to FF FFFC (word space) or 2000 to FF FFFE (byte space). The ending address range is 2002 to FF FFFE (word space) or 2001 to FF FFFF (byte space).

Move the cursor to the Ref/All Ref field and select REF (F1) if you want to test the ROM with a specific signature file, or select ALL REF (F2) to test the ROM using all the stored signature files. If you select REF, press the right arrow key to move the cursor to the reference field and key in the name of the signature file you want to use. Press ENTER. When the UUT passes the ROM Test, the BUSY light goes out. If the UUT fails the ROM Test, an error message appears on the second line of the Mainframe display.

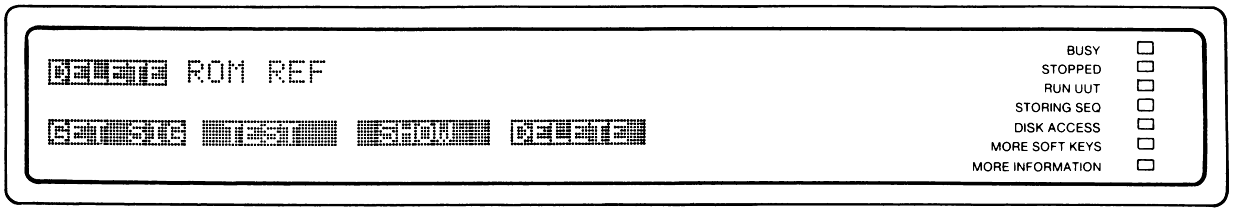
To obtain a list of all the signatures on file, select the ROM Test by pressing ROM on the Mainframe keypad. Move the cursor to the leftmost field using the left arrow key (←). Select SHOW (F3). The Mainframe now displays the signature file SHOW menu:



```
SHOW ALL ROM REF

GET SIG   TEST   SHOW   DELETE
```

Press ENTER. The list of signature files is shown on the Mainframe display. If the MORE INFORMATION indicator on the display is lit, you can scroll though the file by pressing the down arrow (↓) to move down or the up arrow (↑) to move up.

To delete a signature file, select the ROM Test by pressing ROM on the Mainframe keypad. Move the cursor to the leftmost field using the left arrow key (←). Select DELETE (F4). The Mainframe now displays the signature file DELETE menu:



```
DELETE ROM REF

GET SIG  TEST  SHOW  DELETE
```

Move the cursor by pressing the right arrow key (→). Enter the name of the signature file to be deleted. Press ENTER. The signature file is deleted from the list. If the file you selected to be deleted does not exist, the Mainframe displays an *Entry not found for XXXX* error message.

For more information on the ROM Test and signature gathering, see the 9100-Series Technical User's Manual.

TESTING INTERRUPT CIRCUITRY 2-24.

Once you begin testing your UUT, you may need to investigate the operation of interrupts. The Pod provides special functions for reading the type and address information that results from received interrupts. It also provides the ability to force interrupt-acknowledge signals, so that you can exercise other parts of the UUT's interrupt-handling facilities.

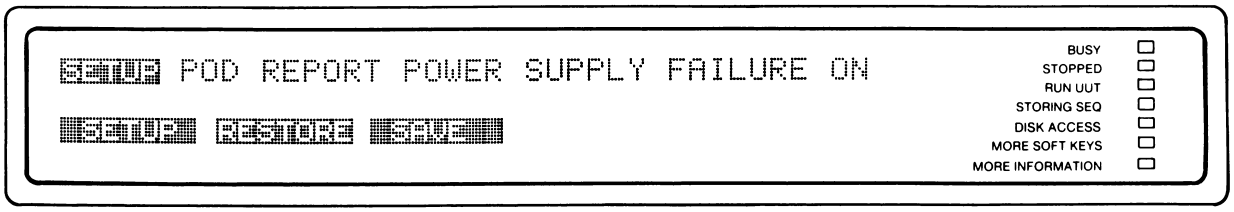
NOTE

All interrupts are handled by a common diagnostic routine in the Pod. User-specified interrupt routines are supported in RUN UUT mode only.

Enabling Interrupts 2-25.

The following steps allow the Pod to respond to interrupts generated by the UUT.

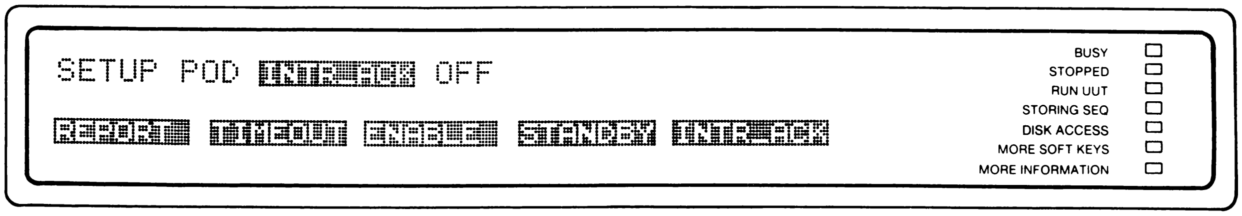
1. Press the SETUP MENU key on the Mainframe keypad. The following menu appears on the display:



```
SETUP POD REPORT POWER SUPPLY FAILURE ON

SETUP  RESTORE  SAVE
```

2. Press the right arrow key (→) to move to the second field. If this field is not set to POD, press F1.

3. Press the right arrow key to move to the next field. Press the F5 softkey to select the INTR_ACK function. The following menu appears on the display.



4. Press the right arrow key. Select the ON (F2) function key.

Enabling interrupts also clears the INT VECT status bit, the interrupt type, and the cascade addresses.

To disable interrupts, use the same steps listed above to enter the INTR_ACK menu. Move the cursor to the OFF/ON field and select OFF (F1). Press ENTER on the keypad to disable interrupts. Reading the interrupt type (function RD_REARM in the Pod menu), as described further on in this section, re-enables the interrupt and clears the INT VECT status bit.

Reading Interrupt Types 2-26.

When an interrupt is acknowledged, the interrupting device places a byte of data on the bus that contains the interrupt type (0-FF hex) associated with the device requesting service. The Pod captures this information from the UUT so that you can view it later.

The interrupt type can be determined by pressing the POD key and selecting either RD_NOARM (F4) or RD_REARM (F5). Note that selecting RD_REARM (F5) also clears the associated status bit and re-enables the interrupts. The other function, RD_NOARM (F4), does not re-enable the interrupts.

RD_NOARM (F4)	Returns the interrupt type, but does not clear the INT VECT bit or re-enable INTR.
RD_REARM (F5)	Returns the interrupt type. Reading this Pod Function address re-enables INTR and clears the INT VECT status bit.

These and other Pod functions used for interrupt handling are listed in Table 2-1.

Table 2-1. Interrupt Handling

POD FIELD	DESCRIPTION
FRC_INT	Force Interrupt Acknowledge Bus Cycle
FRC_RINT	Force Interrupt Acknowledge Bus Cycle and Loop
RD_CSCD	Read Cascade Address
RD_NOARM	Read Interrupt Type and do not Re-enable
RD_REARM	Read Interrupt Type and Re-enable
VEC_CAPT	Read Status and Test for INT VECT

NOTE

Whenever an interrupt is acknowledged, succeeding interrupts are temporarily disabled until the current interrupt type is read. This prevents multiple interrupts from removing the current interrupt type before it can be read by the Mainframe.

Reading Cascade Addresses 2-27.

When the UUT has multiple slave Programmable Interrupt Controllers (PICs), the Master PIC provides a cascade address during the interrupt-acknowledge routine to indicate which slave originated an interrupt. If the UUT is so designed, the PIC can put the cascade address on the UUT's address bus, and the Pod can capture it during INTA cycles. The cascade address can be read by entering the Pod menu, selecting RD_CSCD (F3), and pressing ENTER. Selecting this function does not re-enable the interrupts or clear the INT VECT status bits.

Forcing Interrupt Acknowledge 2-28.

To help test interrupts, the Pod allows you to force interrupt-acknowledge cycles. Selecting FRC_INT (F1) from the Pod menu forces an interrupt-acknowledge cycle without requiring a UUT interrupt request to occur. The forced interrupt clears previous types and cascade addresses and replaces them with new information. The Mainframe then displays the new interrupt type.

You can verify the data path from the interrupt controller to the 80286 by using the Mainframe probe in pulser mode (see Fluke Application Bulletin B0157 for an example of this technique).

Alternatively, the interrupt-type and cascade address information captured by the 80286H Pod can be read as described earlier in this section under the heading "Reading Interrupt Types".

Selecting FRC_RINT (F2) in the Pod menu results in the same function as FRC_INT (F1), but also continuously repeats the function. FRC_RINT increases the repetition rate of the function and, as an example, makes the interrupt signal easier to see on an oscilloscope.

Using the Interrupt-Acknowledge Sync 2-29.

You may use the Mainframe's Synchronization function to synchronize both probe operation and the rear panel TRIGGER OUTPUT pulse (for an oscilloscope) to the Pod's interrupt-acknowledge cycle. Press SYNC on the Mainframe keypad to select the interrupt-acknowledge sync mode. Use the right arrow key to move the cursor one field to the right. Select the POD function (F2). Press the right arrow key again and select the INTA function (F2).

When interrupt-acknowledge sync is selected, the scope trigger pulses low at the start of the interrupt-acknowledge cycle, and pulses high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle (see Figure 5-4 in Section 5 of this manual). For more information about the Mainframe synchronization function, see the 9100-Series Technical User's Manual.

USING THE RUN UUT MODE 2-30.

The RUN UUT mode allows the Pod to emulate the UUT's microprocessor by executing a program directly from the UUT's memory.

The 80286 has two modes of memory addressing: Real Address Mode and Protected Address Mode. In Real Address Mode, 20 physical address lines (A0 through A19) and $\overline{\text{BHE}}$ are used to address up to one megabyte (1,048,576 bytes) of memory. In Protected Address Mode, 24 physical address lines (A0 through A23) and $\overline{\text{BHE}}$ are used to address up to 1 gigabyte of virtual address space mapped into a physical address space of 16 megabytes.

All addresses that are executed in RUN UUT are formed by the 80286 CS and IP registers to make up the full 20-bit virtual address pointer. Three address formats are valid for RUN UUT at other than the default address: 00FF XXXX, 000Y XXXX, and F000 XXXX. The Pod configures the entry state of the 80286 depending on the address format specified by the RUN UUT instruction. The first format type (00FF XXXX) runs in the upper 64k bytes of the 16M byte memory space. The second format type (000Y XXXX) runs in the bottom 1M bytes of the 16M byte memory space, as the processor appears in Real Address mode. The Pod can only enter RUN UUT from an address in Real Address Mode (0000 0000 to 000F FFFE) or in the top 64k bytes of the 16M byte memory space (00FF 0000 to 00FF FFFE). RUN UUT operation in other address areas is possible if UUT software makes the switch to Protected Mode.

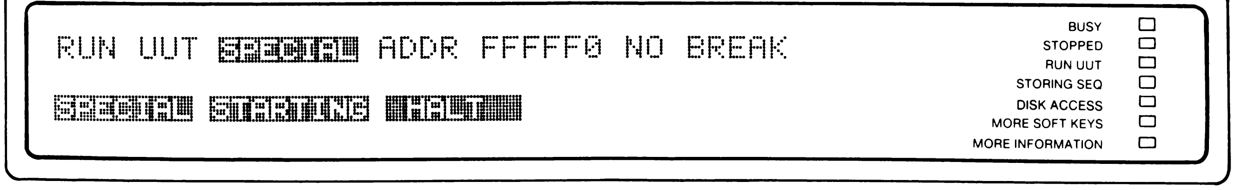
Whenever any RUN UUT starting address in the bottom 1M bytes is specified, the Pod considers the offset address (the IP register contents) to be equal to the four least-significant digits, and the CS register contents to equal the fifth hexadecimal digit of the address followed by three zeros (as shown in Table 2-2, RUN UUT Type 2).

When you select RUN UUT, you may either use the default execution address or you may explicitly specify the address where execution begins. Each of these cases are described in the following paragraphs. The Pod Function addresses that pertain to the RUN UUT mode are listed in Table 2-2.

Using the Default Execution Address 2-31.

If the RUN UUT mode is selected and you do not specify the execution address, the Mainframe supplies the address 00FF FFF0 as the execution address. Execution at this address sets the segment registers to their reset values: 0000 for the ES, SS, and DS registers, and F000 for the CS register; the offset address (IP register) is set to FFF0. This emulates execution from a hardware reset in the UUT.

To use RUN UUT at the default address, press RUN UUT on the keypad. The Mainframe displays the following message:



```
RUN UUT SPECIAL ADDR FFFFF0 NO BREAK

SPECIAL STARTING HALT
```

Press ENTER to begin RUN UUT at the default address.

Table 2-2. 9100-Series RUN UUT Control Addresses

FUNCTION	ADDRESS	DESCRIPTION
RUN UUT Type 1 (Using the Default Execution Address)	FF FFF0	This is the RUN UUT default address that is supplied if the operator does not specify an execution address. RUN UUT execution at FF FFF0 sets all segment registers to their reset values (0000 for ES, SS, and DS, and F000 for CS) and sets the offset (IP register) to FFF0.
RUN UUT Type 2 (Specifying the Execution Address — bottom 1M bytes)	Y XXXX	The RUN UUT execution address may be specified by entering the address from the Mainframe as Y XXXX, where the CS register equals Y000, and the offset (IP register) is XXXX. This allows you to begin operation anywhere in the bottom 1M byte of UUT memory.
RUN UUT Type 3 (Specifying the Execution Address — top 64k bytes)	FF XXXX	The RUN UUT execution address may be specified by entering the address from the Mainframe as FF XXXX, where the upper four address bits are set to F, the CS register equals F000, and the offset (IP register) is XXXX. This allows you to begin operation anywhere in the upper 64k bytes of UUT memory.
RUN UUT Type 4 (Specifying the Special Function Address)	F000 XXXX	First, explicitly load the CS register by pressing SETUP MENU, selecting the SEG_REG function, moving to the register field and selecting the CS register, and then entering the data in the data field. Next, press RUN UUT, select address F000 XXXX where XXXX is the offset (IP register), and press ENTER. The resulting address is the combination of the CS register shifted left 4 bits and the offset.

NOTE

The Segment Registers may be defined prior to a Type 4 RUN UUT by entering the Setup functions, selecting the SEG_REG function, then selecting the appropriate register.

F1 = CS register initial contents (default=F000)
F2 = DS register initial contents (default=0000)
F3 = SS register initial contents (default=0000)
F4 = ES register initial contents (default=0000)

After the desired values are written into the segment register fields, execute RUN UUT at the address F000 XXXX, where XXXX is the desired offset address. The specified contents are loaded into the segment registers. As usual, the RUN UUT execution addresses are formed using the CS register: CS register contents are shifted left four bits, then added to the offset for the execution address.

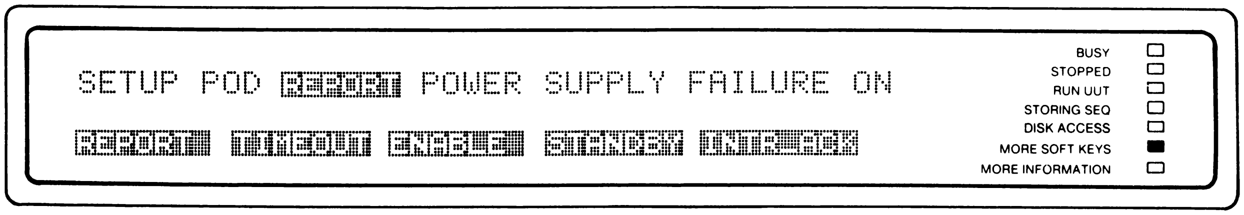
Specifying the RUN UUT Address 2-32.

You may select the address where RUN UUT execution begins by using either of two methods: by changing the default RUN UUT entry address in the address field of the Mainframe RUN UUT command, or by using Special Function addresses within the Pod.

You may change the RUN UUT default address by specifying the new address in the RUN UUT command line on the Mainframe display. Press RUN UUT on the keypad, select STARTING (F2), press the right arrow ( ) key to move to the address field, and select the new address. The address to be specified must be within the range of 00 0000 to 0F FFFF or FF 0000 to FF FFFF.

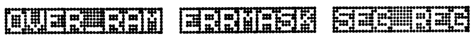
Using the Special Function address F000 XXXX for RUN UUT causes the RUN UUT to begin with the segment registers initialized to user-specified values. Using this method, you can specify RUN UUT to begin anywhere in the bottom 1M bytes of memory only, without the segment registers being reset when RUN UUT begins. For example, to perform a RUN UUT at A BCDE, first load the CS register with A000 using the following steps:

1. Press the SETUP MENU key on the Mainframe keypad. Select SETUP in the first field, press the right arrow key (→) to move the cursor to the second field, and select POD. Press the right arrow key to move the cursor to the third field. The following menu appears on the display:



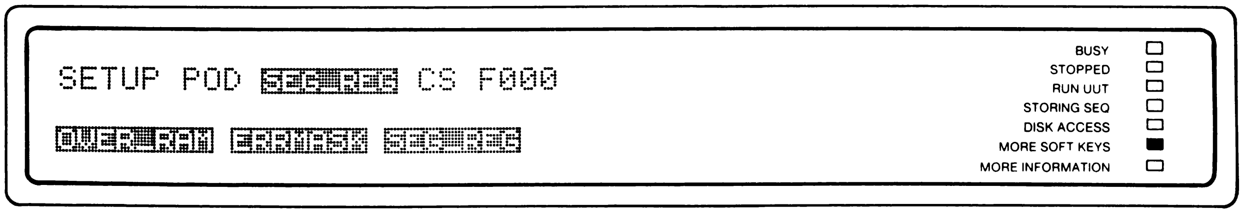
```
SETUP POD REPORT POWER SUPPLY FAILURE ON

REPORT  TIMEOUT  ENABLE  STANDBY  INTR_ACK
```

2. Press SOFT KEYS on the keypad. The softkeys at the bottom of the display change to:



```
OVER_RAM  ERRMASK  SEG_REG
```

Select SEG_REG (F3). The following menu now appears on the display:



```
SETUP POD SEG_REG CS F000

OVER_RAM  ERRMASK  SEG_REG
```

3. Press the right arrow key (→). The cursor moves the the segment register field. Press F1 to select the CS register.

4. Press the right arrow key again. The cursor moves to the segment register data field. Enter A000 into this field. To finish, press ENTER on the keypad.

Next, specify a RUN UUT using the F000 XXXX address, RUN UUT SPECIAL ADDR F000 BCDE NO BREAK. When you press ENTER, the bits in the CS register are shifted left and added to the lower four digits of the address, and the Pod begins RUN UUT.

```
┌──1010 0000 0000 0000◄──┐   1011 1100 1101 1110    (binary)
│   A    0    0    0     │    B    C    D    E      (hex)
└────────────────────────┘
```

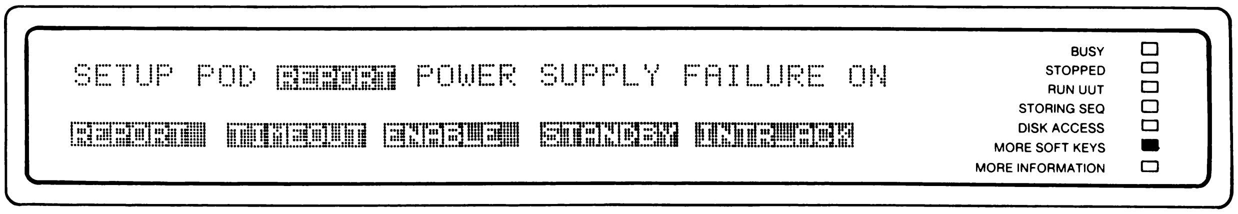
becomes

```
0000 0000 0000 1010 1011 1100 1101 1110    (binary)
  0    0    0    A    B    C    D    E      (hex)
```

Specifying Segment Register Contents 2-33.

The contents of any or all of the segment registers may be specified before RUN UUT execution begins. To specify the segment register contents, use the following procedure:

1. Press the SETUP MENU key on the Mainframe keypad. Select SETUP in the first field, press the right arrow key (→) to move the cursor to the second field, and select POD. Press the right arrow key to move the cursor to the third field. The following menu appears on the display:

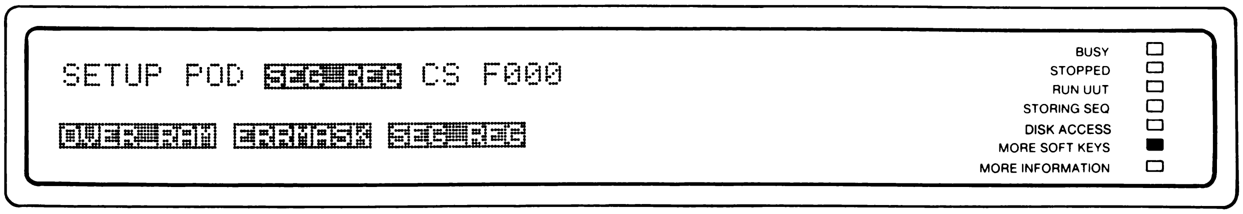


```
SETUP POD REPORT POWER SUPPLY FAILURE ON

REPORT  TIMEOUT  ENABLE  STANDBY  INTR_ACK
```

2. Press SOFT KEYS on the keypad. The softkeys at the bottom of the display change to:

   OVER_RAM ERRMASK SEG_REG

   Select SEG_REG (F3). The following menu now appears on the display:



```
SETUP POD SEG_REG CS F000

OVER_RAM  ERRMASK  SEG_REG
```

3. Press the right arrow key (→). The cursor moves the the segment register field. Select the segment register by pressing the appropriate softkey.

   F1 = CS register initial contents (default=F000)
   F2 = DS register initial contents (default=0000)
   F3 = SS register initial contents (default=0000)
   F4 = ES register initial contents (default=0000)

4. Press the right arrow key (→) again, then write the desired 16-bit value into the data field.

5. After the desired values are written to the data field, specify RUN UUT at the address F000 XXXX (XXXX equals the offset address). When RUN UUT begins, the initial contents of the segment registers equal the values placed into the data field.

NOTE

During execution of RUN UUT, no information is passed back to these data fields; even though the segment register contents change, the values in the Pod segment register fields are unchanged by RUN UUT. As usual, the RUN UUT execution addresses are formed using the CS register contents and the offset address. (The value in the CS register is shifted left four bits and then added to the offset address.)

USING BREAKPOINTS 2-34.

The 80286H Pod implements hardware breakpoints as an optional means of terminating the RUN UUT operation. A "break" occurs in the RUN UUT operation when the address on the 80286 external physical address bus matches the user-specified break address. Once the break occurs, control is transferred to the Mainframe.

Enabling Breakpoints 2-35.

Breaks are enabled by entering the RUN UUT menu, moving the cursor to the Break/No Break field, and selecting BREAK (F2). Breaks are disabled by entering the RUN UUT menu, moving the cursor to the Break/No Break field, and selecting NO BREAK (F1). If breaks are disabled and then reenabled without changing the break address, the break address remains the same.

Setting the Break Address 2-36.

To set the breakpoint address, enter the RUN UUT menu, move the cursor to the Break/No Break field and select BREAK (F2). Move the cursor to the rightmost field, and enter the address. Only one break address can be in effect at a time.

A break only occurs on instruction fetch cycles. As soon as the physical address of the 80286 matches the contents of the RUN UUT break address field (and breaks are enabled), the break occurs. Since an instruction fetch is always a word or upper byte access, the break ignores the A0 bit.

Once a break occurs, the Pod resumes standby reads to the UUT (unless standby reads are disabled). The 80286 register contents before the break are lost and cannot be recovered. The contents of the Overlay RAM remain intact and can be read.

NOTE

Due to pipelined instruction fetches, actual execution may be several bytes behind the current instruction fetch. Therefore, a break could occur improperly if the breakpoint is set for a point close to, but just after a jump or call instruction. Even though the breakpoint is not executed, it may be prefetched and cause a break to occur. To prevent this type of error, position the breakpoint after the destination of a jump or call instruction.

To confirm that a break has occurred, check the RUN UUT indicator on the right side of the Mainframe display. After a break, the RUN UUT indicator turns off. Enter RUN UUT and select HALT(F3). A message indicates that a break has occurred. No message means the break did not occur.

USING THE OVERLAY RAM 2-37.

Overlay RAM is 8k bytes of contiguous RAM contained in the Pod. Overlay RAM may be substituted for UUT memory to execute machine code in RUN UUT to test the UUT. You may select the starting address of the Overlay RAM by entering the address in the Setup menu. You may also enable $\overline{\text{READY}}$ from either the Pod or the UUT from this menu.

Whenever the Overlay RAM is mapped to begin at a particular address and is enabled, a read within the Overlay RAM address range returns the data in the Overlay RAM rather than the UUT memory. Overlay RAM can be mapped to any memory address location, even UUT ROM space. Breakpoints can also be used in conjunction with Overlay RAM to exit a test routine.

Overlay RAM can be located at any 8k byte boundary in memory. (This is a 2000 hex increment, e.g., 78000, 22000, FE000.) If Overlay RAM is used together with RUN UUT outside the Real Address mode range (0000 0000 to 000F FFFE) or the top 64k bytes of the 16M byte memory space (00FF 0000 to 00FF FFFE), UUT software must make the switch to Protected Mode.

Any value written to Overlay RAM is also written to the UUT at the same address. Likewise, a read from the Overlay RAM is also performed on the UUT (but the UUT data is ignored).

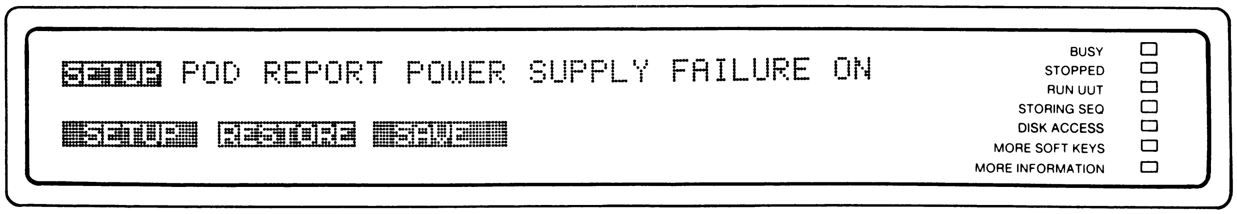
NOTE

Although Overlay RAM is logically mapped into the UUT's address space, physically it is not in the UUT. Because of this, unexpected results could occur. No data read from the Overlay RAM is present on any of the UUT's data buses. This may cause a problem for any UUT circuits that depend on the processor data (such as an 80287 math coprocessor).

Enabling Overlay RAM 2-38.

When power is applied to the Pod, the Overlay RAM is disabled (default) and does not interact with the UUT in any way. (The Pod does not perform a RUN UUT out of the Overlay RAM until the Overlay RAM is enabled.) To enable the Overlay RAM:

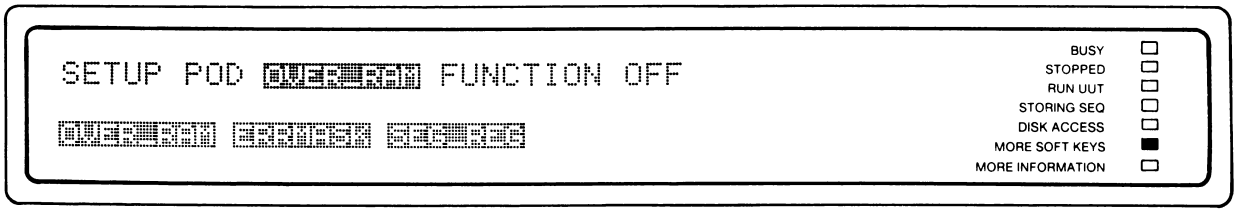
1. Press the SETUP MENU key on the Mainframe keypad to enter the Setup Menu. The Mainframe displays the following menu:



```
SETUP POD REPORT POWER SUPPLY FAILURE ON
```

2. Press the right arrow key (→) to move the cursor to the second position. If the second field does not say POD, press POD (F1).

3. Press the right arrow key to move the cursor to the third position on the menu. Press SOFT KEYS to scroll the menu selections. Press OVER_RAM (F1) to select the Overlay RAM function. The Mainframe now displays the following menu:



```
SETUP POD OVER_RAM FUNCTION OFF
```

4. Press the right arrow key to move the cursor to the fourth position on the menu. If the fourth field does not say FUNCTION, press the FUNCTION function key (F1).

5. Press the right arrow key to move the cursor to the ON/OFF field of the menu. Press ON (F2) and then press ENTER on the keypad to enable Overlay RAM.

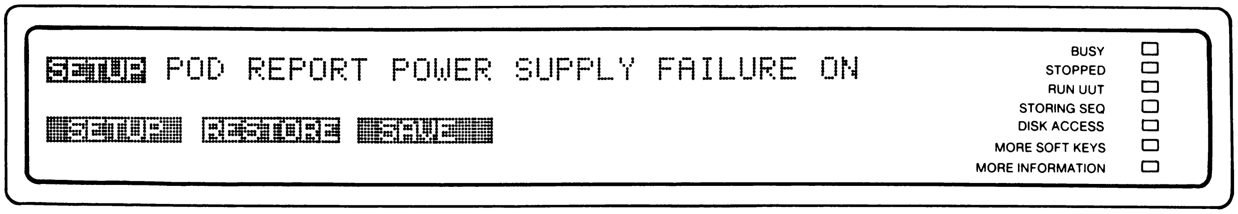
Once enabled, the Overlay RAM is defined by the Pod as 8k bytes of memory beginning at either the default address or the address specified by the user.

Setting the Overlay RAM Start Address 2-39.

The default start address for Overlay RAM is FF E000 (at the top of memory), which puts the Overlay RAM into the microprocessor power-up reset ROM area.

To change the address area covered by the Overlay RAM:

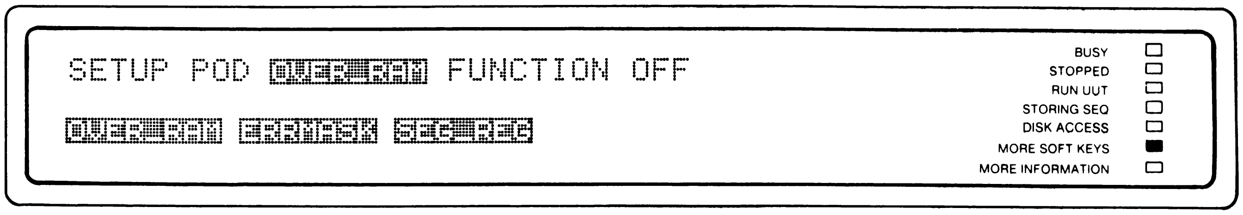
1. Press the SETUP MENU key on the Mainframe keypad to enter the Setup Menu. The Mainframe displays the following menu:



```
SETUP POD REPORT POWER SUPPLY FAILURE ON
```

2. Press the right arrow key (→) to move the cursor to the second position. If the second field does not say POD, press the POD function key.

3. Press the right arrow key to move the cursor to the third position on the menu. Press SOFT KEYS to scroll the menu selections. Press OVER_RAM (F1) to select the Overlay RAM function. The Mainframe now displays the following menu:



SETUP POD OVER_RAM FUNCTION OFF

OVER_RAM ERRMASK SEG_REG

4. Press the right arrow key to move the cursor to the fourth position on the menu. Set this field to BASE_ADR by pressing F2 on the keypad.

5. Press the right arrow key to move the cursor to the last position on the menu. Enter the start address of the Overlay RAM.

   The beginning of Overlay RAM is entered as the 11 most-significant bits of data (D5 through D15). D0 through D4 are ignored. The data entered from the Overlay RAM menu is then converted to the 11 most-significant bits of the Overlay RAM start address. A0 through A12 are set to 0 (Overlay RAM starts at 8k byte boundaries). For example, if you want to set the starting address to FF E000 (the default address, enter the start address as FFE0, or as illustrated below in binary and hex:

	DATA ENTERED	START ADDRESS OF ORAM
Binary:	1111 1111 1110 0000	1111 1111 1110 0000 0000 0000
Hex:	FFE0	FF E000

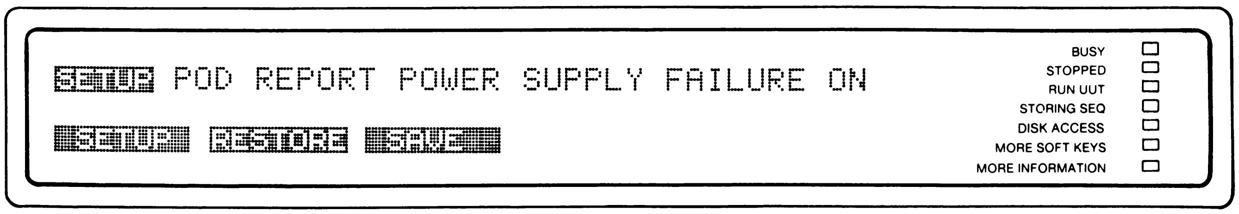
6. Press the ENTER key to set the start address.

Once the starting address is moved and Overlay RAM is enabled, any UUT access (such as RUN UUT) at the addresses within the space assigned to Overlay RAM is mapped to the Overlay RAM. Also, any Read or Write function from the Mainframe that accesses the Overlay RAM space is mapped to Overlay RAM.

Setting the Source of $\overline{\text{READY}}$ for Overlay RAM 2-40.

While using Overlay RAM, the Pod can take the processor $\overline{\text{READY}}$ signal from the UUT or from a signal generated internally in the Pod. The default source of $\overline{\text{READY}}$ is from the UUT. Use the following method to select the source of the $\overline{\text{READY}}$ signal to the 80286:

1. Press the SETUP MENU key on the Mainframe keypad to enter the Setup Menu. The Mainframe displays the following menu:

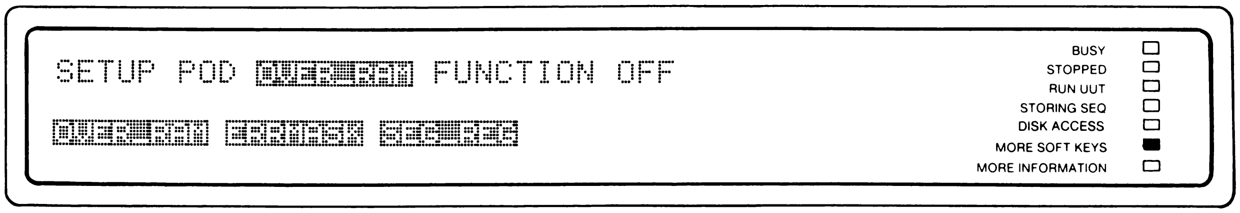


SETUP POD REPORT POWER SUPPLY FAILURE ON

SETUP RESTORE SAVE

2. Press the right arrow key (→) to move the cursor to the second position. If the second field does not say POD, press the POD function key.

3. Press the right arrow key to move the cursor to the third position on the menu. Press SOFT KEYS to scroll the menu selections. Press OVER_RAM (F1) to select the Overlay RAM function. The Mainframe now displays the following menu:



```
SETUP POD OVER_RAM FUNCTION OFF

OVER_RAM ERR_MSK SEG_REG
```

4. Press the right arrow key again to move the cursor to the fourth position on the menu. Set this field to RDY_SRC by pressing F3 on the keypad.

5. Press the right arrow key to move the cursor to the last field. To enable $\overline{\text{READY}}$ from the UUT, press UUT (F1). To enable $\overline{\text{READY}}$ from the Pod with no wait states, press AUTO (F2).

6. Press the ENTER key to set the source of $\overline{\text{READY}}$.

NOTE

Synchronization problems could occur if the Pod is using the Overlay RAM as a substitute for UUT RAM but $\overline{\text{READY}}$ from the UUT is not enabled within the Pod. If the UUT tries to insert wait states (while the Overlay RAM is running with no wait states), the Pod and the UUT may no longer be synchronized.

STANDBY READS 2-41.

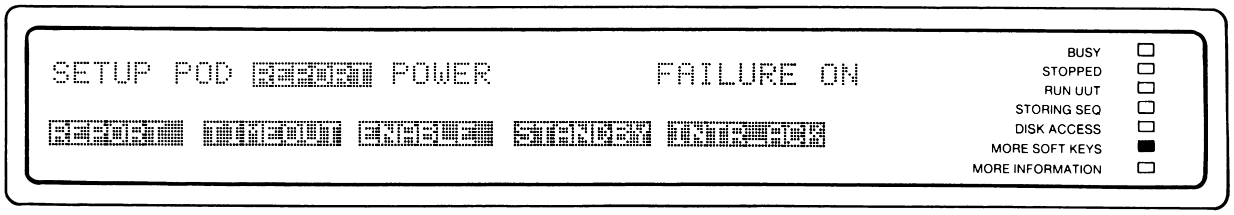
To provide UUTs with memory accesses that refresh dynamic memory devices, the Pod does repetitive standby or "transparent" read operations. When the UUT is idle (not doing a RUN UUT operation), its normal methods for refreshing memory may not work. These transparent or standby read operations continually read a single address to keep the UUT's memory functioning correctly.

Changing the Standby Read Address 2-42.

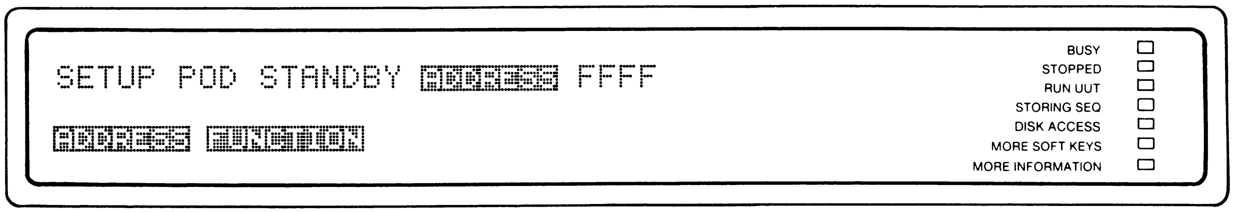
If the default address provided by the Pod does not access RAM on your UUT, or if that particular address should not be read (because, for example, it causes an unwanted operation on the UUT), then you may change the address that is used.

The default value for the standby read address is FFFF. A value of FFFF sets the default address to FF FF00. The least significant 8 bits (D0 through D7) of the standby read address are always zero.

To change the standby read address, first press the SETUP MENU key on the Mainframe keypad. Select SETUP (F1). Press the right arrow key (→) to move the cursor. Select POD (F1). Press the right arrow key again. The Mainframe displays the following menu:


SETUP POD REPORT POWER FAILURE ON

Select STANDBY by pressing the F4 softkey. Press the right arrow key. The Mainframe displays the following menu:


SETUP POD STANDBY ADDRESS FFFF

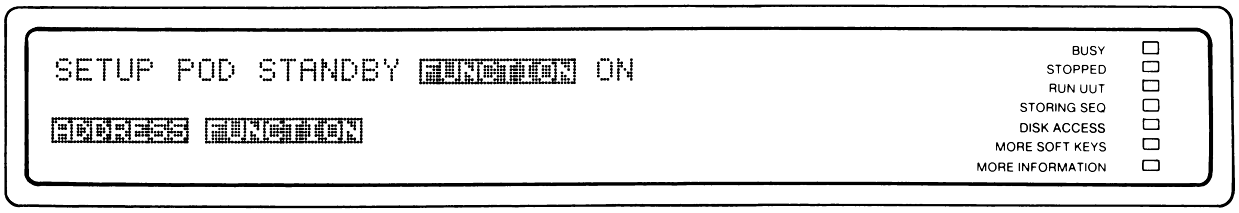
If this field is not set to ADDRESS, press F1. Press the right arrow key to move the cursor to the address field. Enter the new standby read address. To select this address, press ENTER on the keypad.

NOTE

Pod operations can be made to execute faster by selecting a standby read address that causes zero wait states to be inserted into the standby read cycles.

Enabling the Standby Read Operation 2-43.

The standby read operation may be enabled by selecting FUNCTION (F2) from the standby read field. The Mainframe displays the following menu:


SETUP POD STANDBY FUNCTION ON

Press the right arrow key (→) to move the cursor to the ON/OFF selection field. Select ON (F1) to turn standby reads on. To turn standby reads off, select OFF (F2).

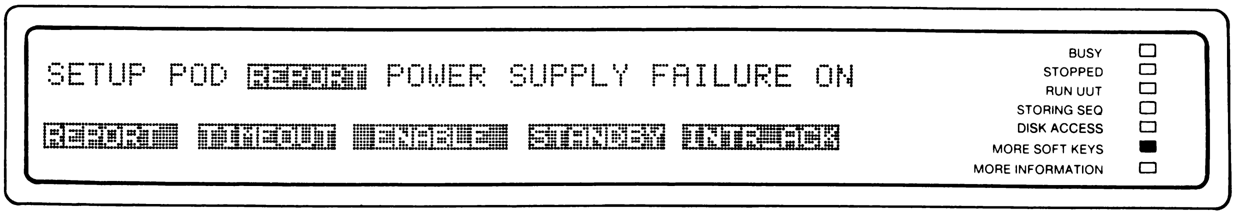
NOTE

Standby reads are disabled while an active RESET signal is present on the UUT.

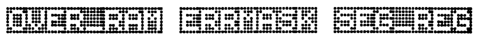
MASKING ERRORS 2-44.

You can use several masks to control how the Pod and Mainframe report errors. To suppress the reporting of individual errors, possibly because a particular error is recurring or may be the result of a design quirk in the UUT, you may instruct the Pod and Mainframe to continue without stopping or displaying an error message when that error occurs. You can use this capability to avoid getting stuck at certain errors during troubleshooting. (To mask an entire class of faults, use the SETUP REPORT functions. For more information, see the 9100-Series Technical User's Manual).

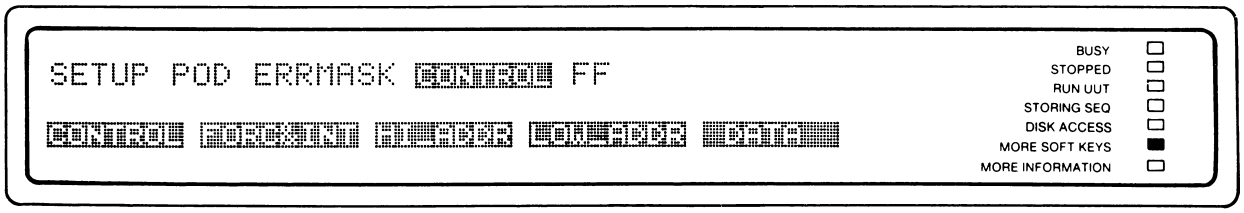
To set the Mainframe to the error mask menu, first press the SETUP MENU key on the Mainframe keypad. Select SETUP (F1). Press the right arrow key (→) to move the cursor. Select POD (F1). Press the right arrow key again. The Mainframe displays the following menu:


SETUP POD REPORT POWER SUPPLY FAILURE ON
REPORT TIMEOUT ENABLE STANDBY INTR ACK
BUSY
STOPPED
RUN UUT
STORING SEQ
DISK ACCESS
MORE SOFT KEYS
MORE INFORMATION

Press SOFT KEYS on the keypad. The softkeys at the bottom of the display change to:



Select ERRMASK by pressing the F2 softkey. Press the right arrow key. The Mainframe displays the following menu:



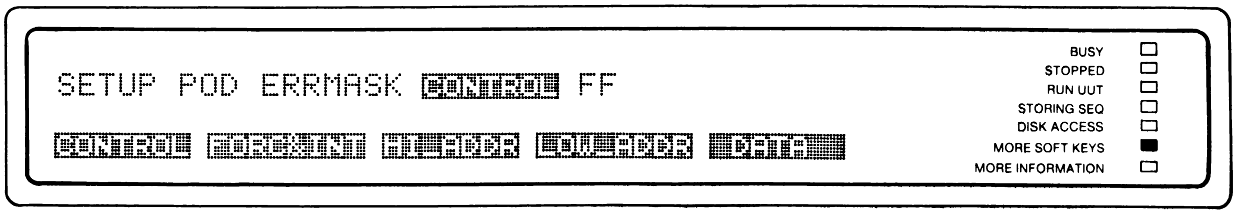
All of the error masks are selected from this menu. A "0" in any bit position signifies that the error assigned to that bit space is not reported.

The default value for all of the masks is no bits masked (all bits "1") or all errors reported. The mask data remains intact until it is changed by writing a new value, performing a Mainframe reset, or removing and restoring power to the Mainframe.

Control Drivability Error Mask 2-45.

The Control Drivability Error Mask masks individual control line bit errors from being reported to the Mainframe. The data corresponds to the control line bit assignments in Table 4-4 in Section 4 of this manual. A data bit value of 0 masks the corresponding control bit. The power up default mask is FF.

To change the Control Mask, enter the ERRMASK menu as described earlier, then press the CONTROL (F1) softkey. The Mainframe displays the following menu:



SETUP POD ERRMASK CONTROL FF

CONTROL FORC&INT HI_ADDR LOW_ADDR DATA

Press the right arrow key (→), select the value of the mask for the Control Drivability Error Mask, and press ENTER.

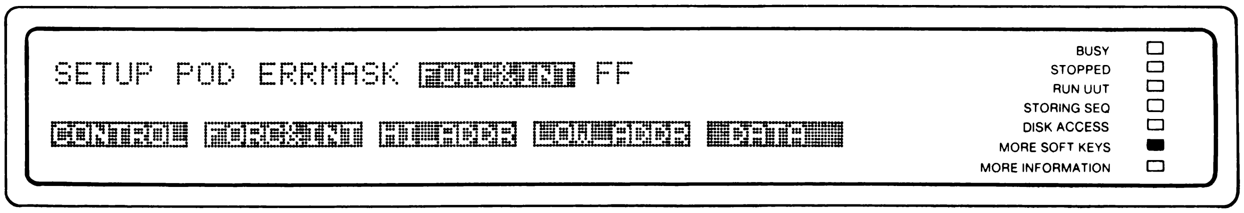
Forcing Line and Interrupt Error Mask 2-46.

The Forcing Line and Interrupt Error Mask masks individual forcing lines and interrupt lines from being reported to the Mainframe. The lines that are masked correspond to any data bits that are specified as 0 in the bit assignments described below.

BIT	STATUS LINE
0	$\overline{\text{READY}}$
1	HOLD
2	NMI
3	INTR
4	RESET
5	PEREQ
6	$\overline{\text{BUSY}}$
7	$\overline{\text{ERROR}}$

Data bits 8 through 15 are ignored and normally set to 0. The power-up default mask is FF (no lines masked).

To change the Forcing Line and Interrupt Error Mask, enter the ERRMASK menu as described earlier, then press the FORC&INT (F2) softkey. The Mainframe displays the following menu:



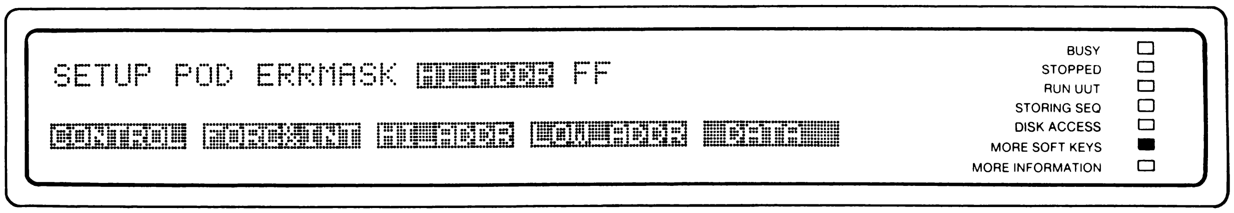
SETUP POD ERRMASK FORC&INT FF

CONTROL FORC&INT HI_ADDR LOW_ADDR DATA

Press the right arrow key (→), select the value of the mask for the Forcing Line and Interrupt Error Mask, and press ENTER.

High Address Drivability Error Mask 2-47.

The High Address Drivability Error Mask masks drivability errors on the high byte (A16 through A23) of the UUT's address bus. The power up default value is FF. To change this error mask, enter the ERRMASK menu as described earlier, then press the HI_ADDR (F3) softkey. The Mainframe displays the following menu:

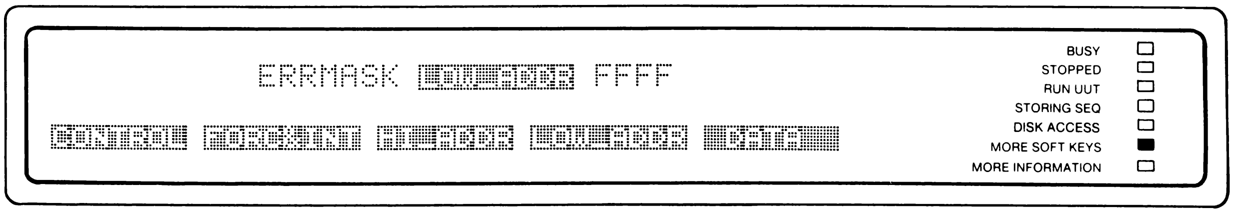


```
SETUP POD ERRMASK HI_ADDR FF
```

Press the right arrow key (→), select the value of the mask for the High Address Drivability Error Mask, and press ENTER.

Low Address Drivability Error Mask 2-48.

The Low Address Drivability Error Mask masks drivability errors on the low order 16 bits (A0 through A15) of the UUT's address bus. The power up default value is FFFF. To change this error mask, enter the ERRMASK menu as described earlier, then press the LOW_ADDR (F4) softkey. The Mainframe displays the following menu:

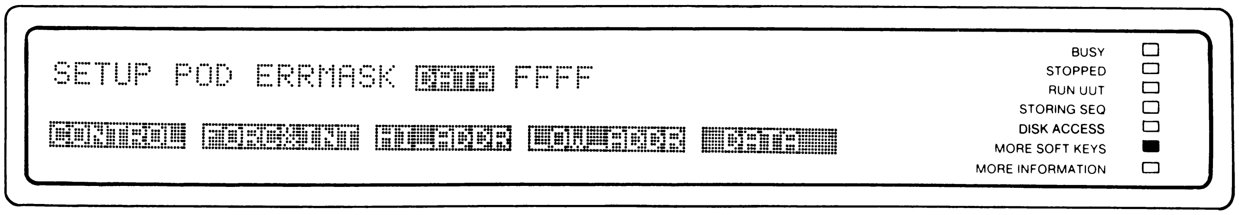


```
ERRMASK LOW_ADDR FFFF
```

Press the right arrow key (→), select the value of the mask for the Low Address Drivability Error Mask, and press ENTER.

Data Drivability Error Mask 2-49.

The Data Drivability Error Mask masks the map of the UUT's data bus bits. The power up default condition for this mask is FFFF. To change this error mask, enter the ERRMASK menu as described earlier, then press the DATA (F5) softkey. The Mainframe displays the following menu:



```
SETUP POD ERRMASK DATA FFFF
```

Press the right arrow key (→), select the value of the mask for the Data Drivability Error Mask, and press ENTER.

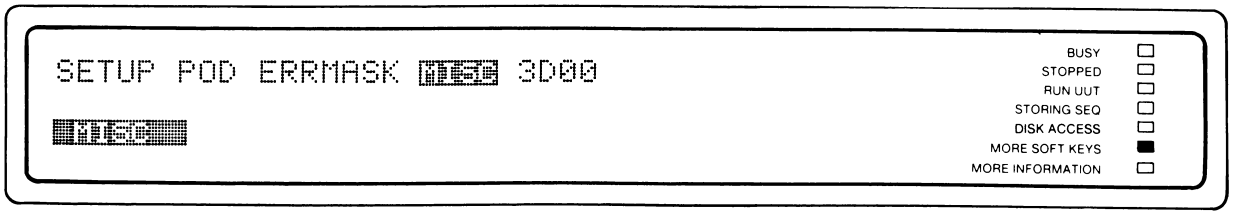
Miscellaneous Line Error Mask 2-50.

The Miscellaneous Line Error Mask masks individual pseudo-status lines and the CAP FAIL line from being reported to the Mainframe. The lines that are masked correspond to any data bits that are specified as 0 in the bit assignments described below.

BIT	STATUS LINE
8	CAP FAIL
9	—
10	PWR FAIL 30
11	PWR FAIL 62
12	GND FAIL 35
13	GND FAIL 9

Data bits 0 through 7, 9, 14, and 15 are ignored and normally set to 0. The power-up default mask is 3D00 (no lines masked).

To change the Miscellaneous Line Error Mask, enter the ERRMASK menu as described earlier, press SOFT KEYS, and select MISC (F1). The Mainframe displays the following menu:



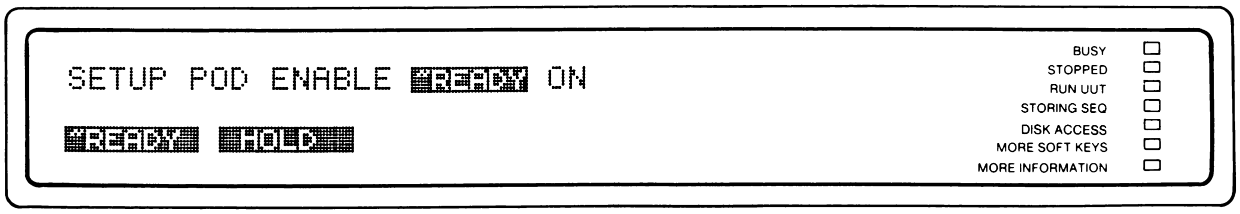
```
SETUP POD ERRMASK MISC 3D00

MISC
```

Press the right arrow key (→), select the value of the mask for the Miscellaneous Line Error Mask, and press ENTER.

USER-ENABLEABLE FORCING LINES 2-51.

The 80286 has two status lines that are individually enabled or disabled using the Mainframe's Setup function: $\overline{\text{READY}}$ and HOLD. When these user-enableable lines are disabled, their inputs to the Pod microprocessor are disabled, but the Pod continues to monitor their condition. If the user-enableable lines are asserted, the Pod reports to the Mainframe that a forcing line is active.

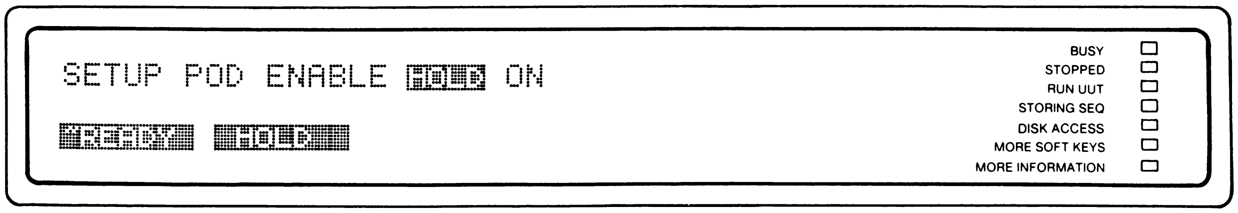
To enable or disable the $\overline{\text{READY}}$ line, enter the Setup menu by pressing the SETUP MENU key. Press the right arrow key (→) to move the cursor to the next field and select POD (F1). Press the right arrow key again and select ENABLE (F3). Press the right arrow key. The Mainframe displays the following menu:



```
SETUP POD ENABLE ~READY ON

~READY  HOLD
```

Select ~READY (F1) and press the right arrow key. To enable the $\overline{\text{READY}}$ status line, press ON (F1). To disable the $\overline{\text{READY}}$ status line, press OFF (F2).

To enable or disable the HOLD line, enter the Setup menu as described above, select POD and ENABLE, and press the HOLD (F2) softkey. The Mainframe displays the following menu:



```
SETUP POD ENABLE HOLD ON

READY  HOLD
```

To enable the HOLD status line, press ON (F1). To disable the HOLD status line, press OFF (F2).

NOTE

To disable the reporting of all active forcing lines, select the Mainframe function, SETUP POD REPORT FORCING SIGNAL ACTIVE OFF. The Pod no longer monitors the lines, and the Mainframe does not interrupt its operation to display the active Forcing line error message. Sometimes it is useful to disable the reporting of active forcing lines, particularly if the information is not needed by the operator.

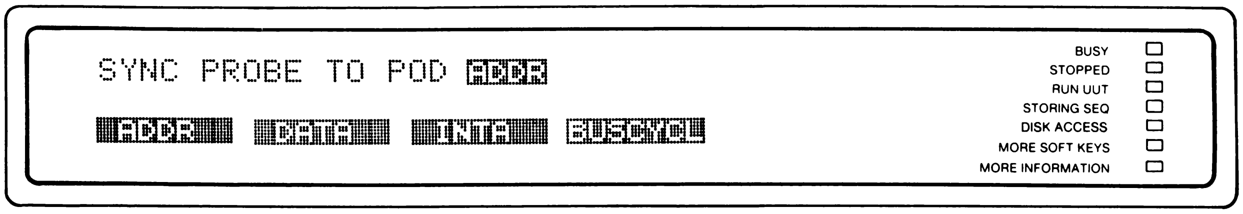
For more information on forcing lines and user-enableable forcing lines, see the heading "Status Lines" in Section 4 of this manual.

PROBE AND SCOPE SYNCHRONIZATION MODES 2-52.

You may use the Mainframe's Synchronization function (selected with the SYNC key) to synchronize both probe operation and the rear panel TRIGGER OUTPUT pulse (for an oscilloscope) to events on the Pod's buses. The five synchronization modes that are available and their Mainframe selection codes are:

ADDR = Address Sync
BUSCYCL = Bus Cycle Sync
DATA = Data Sync
INTA = Interrupt-Acknowledge Sync
FREERUN = Free-Run Sync

To select one of these synchronization functions (except FREERUN), press the SYNC key on the Mainframe, move to the second field by pressing the right arrow key, and then select POD (F2). Move the cursor to the next field. The Mainframe displays the following menu:



```
SYNC PROBE TO POD ADDR

ADDR  DATA  INTA  BUSCYCL
```

To select the Free-Run Sync, press the SYNC key on the Mainframe and then select FREERUN (F1).

The main purpose of the various synchronization modes is to provide proper signal timing for use with the probe. They can also be used to simplify viewing signals with an oscilloscope.

Address Sync 2-53.

If you select the Address Sync, both the probe and scope trigger output are synchronized to the address portion of the UUT access. The scope trigger pulses low shortly before the UUT access begins, and pulses high at the end of the address portion of the UUT-access cycle. If you select the probe stimulus mode, the probe pulses at the selected level (high or low) for the time between the two scope trigger pulses described above. For probe response, the probe latches on the signal level present at its tip at the time of the second or high-going scope trigger pulse.

Bus Cycle Sync 2-54.

If you select Bus Cycle Sync, the scope trigger pulses low at the start of UUT access (when $\overline{S0}$ or $\overline{S1}$ is asserted). The scope trigger pulses high at the end of the data cycle (the trailing edge of the Data Enable pulse).

Data Sync 2-55.

If you select Data Sync, the scope trigger pulses low at the start of the data portion of the UUT access (the end of the address portion). The scope trigger pulses high at the end of the data cycle (the trailing edge of the Data Enable pulse).

Interrupt-Acknowledge Sync 2-56.

If you select Interrupt-Acknowledge Sync, the scope trigger pulses low at the start of the interrupt-acknowledge cycle, and pulses high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle.

Free-Run Sync 2-57.

Free-Run Sync is generated inside the Mainframe. If you select free-run mode, the probe stimulus pulses are generated at a frequency of approximately 1 kHz with a 1% duty cycle.

At power-on, the probe is in the free-run mode, and the scope trigger output pulses are switched off.

Section 3
9000-Series Micro-System Troubleshooter Operation

USING POD FUNCTIONS 3-1.

This section describes how to use the Pod functions to test components and circuitry on a UUT using the 9000-Series Micro-System Troubleshooter. The following subjects are included:

Testing the UUT Bus	How to use the Mainframe and Pod to test the Bus circuitry on the UUT.
Testing the UUT RAM	How to use the Mainframe and Pod to test the RAM circuitry on the UUT.
Testing the UUT ROM	How to use the Mainframe and Pod to test the ROM circuitry on the UUT.
Testing Interrupt Circuitry	How to use the Pod to evaluate Interrupt circuitry on the UUT.
Using the RUN UUT Mode	How to exercise the UUT by using the Pod to emulate its microprocessor.
Using Breakpoints	How to set breakpoints to interrupt the UUT program at specific addresses.
Using the Overlay RAM	How to substitute known-good RAM in the Pod for RAM on the UUT.
Using the Quick Tests and Functions	How to test the UUT using the Pod's built-in quick tests and functions.
Configuring General Pod Characteristics	How to adapt the Pod to accommodate the specific characteristics of your UUT.
Masking Errors	How to suppress the reporting of unimportant or known errors by the Pod.
Determining Errors	How to poll the Pod for specific information about previous errors.

You should be aware of the following address notation conventions in this manual:

- For ease of reading, the two halves of the address are separated with a space in this manual (HHHH LLLL). However, do not try to enter addresses with a space. The Mainframe does not display addresses with a space.
- X is used to denote hex or binary digits, where the specific value may be any valid number. For example, the data value XX00 means that it is only important that the two least-significant digits are zero; the other two digits may be any value.
- Some of the examples show addresses in binary form. Address and data bus values are only entered and displayed on the Mainframe in hexadecimal form, so binary information must be converted to hex before using. For example, an address specification that is illustrated in binary form as 1111 0000 1100 0000 1010 0011 is entered into the Mainframe as F0 C0A3.

NOTE

If the Learn operation is selected and the operator does not specify the starting and ending address for the operation, the Mainframe may take an inordinate amount of time to test the entire memory space of the UUT. If the Learn operation tries to test an area of memory that is not implemented on the UUT (and therefore does not return a $\overline{READY}$ signal), the Mainframe displays a POD TIMEOUT - ATTEMPTING RESET message and then retries the same address.

TESTING THE UUT BUS 3-2.

Bus Test is a test of the electrical integrity of UUT control, address, and data buses connected directly to the processor. Bus Test identifies control lines that are not drivable, as well as address lines that are tied high, low, or tied together, and data lines that are tied high, low, or tied together.

To specify the Bus Test, press the BUS key on the Mainframe keyboard. No other entries are required. As soon as the key is pressed, the Mainframe begins performing the test and displays the following message:

BUS TEST WAIT

After the Bus Test is completed, the Mainframe replaces WAIT with an OK or FAIL.

For more information on the Mainframe Bus Test, see the appropriate 9000-Series Micro-System Troubleshooter Operation Manual.

TESTING THE UUT RAM 3-3.

To ensure that all RAM failures are identified and yet optimize test times, the Mainframe provides the RAM Short and RAM Long tests for the UUT RAM. RAM Short is a shorter and faster test than RAM Long.

RAM Short is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable. To select the RAM Short test, press the RAM SHORT key on the Mainframe keyboard. The Mainframe prompts you to enter the first address. When you have typed in the first address, press the ENTER key. The Mainframe then prompts you to enter the ending address. When you have typed in the ending address, press the ENTER key. The Mainframe then begins the RAM Short test and displays the following message:

RAM SHORT @ ss ssss-ee eeee WAIT	(ss ssss-ee eeee designates the starting and ending addresses)

RAM Short is executed on the address block specified. After RAM Short is completed, the Mainframe replaces the word *WAIT* in the message on the display with *OK* or *FAIL.*

The steps required to select and specify RAM Long are identical to the steps required to select and specify RAM Short. To select RAM Long, press RAM LONG on the Mainframe keyboard. Then enter the first and second addresses as prompted (same method as RAM Short). After the specifications are complete, the Mainframe begins performance of RAM Long. RAM Long is performed on the address block specified. After RAM Long is completed, the Mainframe replaces the word *WAIT* in the message on the display with *OK* or *FAIL.*

For more information on the Mainframe RAM Short and RAM Long tests, see the appropriate 9000-Series Micro-System Troubleshooter Operation Manual. For information on the Quick RAM Test, see the heading "The Quick RAM Test" further on in this section.

TESTING THE UUT ROM 3-4.

ROM Test computes the ROM signature for the user-specified address block and compares the signature to the stored, user-specified signature. A signature is an algorithmic compression (cyclic redundancy check) of a digital bit stream into a four-digit hexadecimal number. Applying the ROM Test algorithm over a range of addresses produces a signature that characterizes that range. The idea of using signatures is to obtain them from data in a known-good UUT, and then compare the good values with values obtained from the suspect UUT.

To select the ROM Test, press ROM on the Mainframe keyboard. Then enter the first and second addresses as prompted (same method as RAM Short). After the addresses are entered, the Mainframe prompts you to enter the signature for the addresses you specified. You must enter the signature gathered from a known good board for the addresses you specified (as shown in the appropriate 9000-Series Micro-System Troubleshooter Operation Manual). Press ENTER. The Mainframe accepts the ROM Signature, begins the test, and displays the following message:

ROM TEST @ ss ssss-ee eeee SIG nnnn WAIT	(ss ssss-ee eeee are the beginning and ending addresses and nnnn is the signature learned from a known good board)

After the specifications are complete, the Mainframe begins performing the ROM Test. After the ROM Test is completed, the Mainframe replaces the *WAIT* with *OK* or *FAIL.*

For more information on the Mainframe ROM Test, see the appropriate 9000-Series Micro-System Troubleshooter Operation Manual.

TESTING INTERRUPT CIRCUITRY 3-5.

Once you begin testing your UUT, you may need to investigate the operation of interrupts. The Pod provides special functions for reading the type and address information that results from received interrupts. It also provides the ability to force interrupt-acknowledge signals, so that you can exercise other parts of the UUT's interrupt-handling facilities.

NOTE

All interrupts are handled by a common diagnostic routine in the Pod. User-specified interrupt routines are supported in RUN UUT mode only.

Enabling Interrupts 3-6.

A WRITE @ F000 0080 = 1 enables interrupts from the UUT to the Pod's microprocessor. The Write operation also clears the INT VECT status bits, the interrupt type, and the cascade addresses.

NOTE

To eliminate conflicts, the 80286H Pod does not work with the Mainframe Setup function to enable or disable interrupts as do Interface Pods for other microprocessors. All interrupt control is accomplished via the Pod Function addresses listed here.

Reading the interrupt type (vector) at F000 0082, as described further on in this section, re-enables interrupts and clears the INT VECT status bits.

Reading Interrupt Types 3-7.

When an interrupt is acknowledged, the interrupting device places a byte of data on the bus that contains the interrupt type (0 through FF hex) associated with the device requesting service. The Pod captures this information from the UUT so that you can view it later.

The interrupt type can be determined by reading the Pod Function addresses F000 0082 and F000 0084. Note that reading address F000 0082 also clears the associated status bit and re-enables interrupts. The other address, F000 0084, does not re-enable interrupts.

READ @ F000 0082	Returns the interrupt type. Reading this Pod Function address re-enables INTR and clears the INT VECT status bit.
READ @ F000 0084	Also returns the interrupt type, but does not clear the INT VECT bit or re-enable INTR.

These and other Pod Function addresses that are used for interrupt handling are listed in Table 3-1.

NOTE

Whenever an interrupt is acknowledged, succeeding interrupts are temporarily disabled until the current interrupt type is read. This prevents multiple interrupts from removing the current interrupt type before it can be read by the Mainframe.

Reading Cascade Addresses 3-8.

When the UUT has multiple slave Programmable Interrupt Controllers (PICs), the Master PIC provides a cascade address during the interrupt-acknowledge routine to indicate which slave originated an interrupt. If the UUT is so designed, the PIC can put the cascade address on the UUT's address bus, and the Pod can capture it during INTA cycles. The cascade address can be found by reading the two Pod Function addresses F000 008C (low word) and F000 008E (most significant byte). Reading these addresses does not re-enable the interrupts or clear the INT VECT status bits.

Forcing Interrupt Acknowledge 3-9.

To help with testing interrupts, the Pod allows you to force interrupt-acknowledge routines. A READ @ F000 0086 forces an interrupt-acknowledge cycle without an interrupt request occurring. The forced interrupt clears the previous type and cascade address and replaces them with new information. The Mainframe then displays the new interrupt type.

You can verify the data path from the interrupt controller to the 80286 by using the Mainframe probe in pulser mode (see Fluke Applications Bulletin B0157 for an example of this technique).

Table 3-1. Interrupt Handling

ADDRESS	DESCRIPTION
F000 0080	Enable Interrupt Acknowledge Cycle
F000 0082	Read Interrupt Type and Re-enable
F000 0084	Read Interrupt Type and do not Re-enable
F000 0086	Force Interrupt Acknowledge Bus Cycle
F000 0088	Force Interrupt Acknowledge Bus Cycle and Loop
F000 008C	Read Cascade Address (lower word)
F000 008E	Cascade Address (most significant byte)

Alternatively, the interrupt-type and cascade address information captured by the 80286H Pod can be read as described earlier in this section under the heading "Reading Interrupt Types".

The Pod also allows you to force interrupt-acknowledge bus cycles and then perform quick loops of the function. A READ @ F000 0088 contains the same function as address F000 0086, but also continuously repeats the function. This is quite useful in INTA sync mode for viewing synchronized signals on an oscilloscope.

Using the Interrupt-Acknowledge Sync 3-10.

You may use the Mainframe's Synchronization function to synchronize both probe operation and the rear panel TRIGGER OUTPUT pulse (for an oscilloscope) to the Pod's interrupt-acknowledge cycle. Press SYNC 1 to select the interrupt-acknowledge sync mode.

NOTE

The interrupt-acknowledge sync mode is selected by the "1" key on the Mainframe (do not use the "I" key).

If interrupt-acknowledge sync is selected, the scope trigger pulses low at the start of the interrupt-acknowledge cycle, and pulses high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle. For more information about the available sync modes, see "Using the Pod with an Oscilloscope" further on in this section.

USING THE RUN UUT MODE 3-11.

The RUN UUT mode allows the Pod to emulate the UUT's microprocessor by executing a program directly from the UUT's memory. In this way, you may check the operation of the UUT under normal operating conditions without having to remove the Pod and reinsert the UUT microprocessor. The RUN UUT mode also allows you to indirectly test the UUT microprocessor by comparing UUT performance with the Mainframe in RUN UUT mode and performance with the Pod removed and the UUT microprocessor reinserted.

The 80286 has two modes of memory addressing: Real Address Mode and Protected Address Mode. In Real Address Mode, 20 physical address lines (A0 through A19) and $\overline{\text{BHE}}$ are used to address up to one megabyte (1,048,576 bytes) of memory. In Protected Address Mode, 24 physical address lines (A0 through A23) and $\overline{\text{BHE}}$ are used to address up to 1 gigabyte of virtual address space mapped into a physical address space of 16 megabytes.

All addresses that are executed in RUN UUT are formed by the 80286 CS and IP registers to make up the full 20-bit virtual address pointer. Three address formats are valid for RUN UUT at other than the default address: 00FF XXXX, 000Y XXXX, and F000 XXXX. The Pod configures the entry state of the 80286 depending on the address format specified by the RUN UUT instruction. The first format type (00FF XXXX) runs in the upper 64k bytes of the 16M byte memory space. The second format type (000Y XXXX) runs in the bottom 1M bytes of the 16M byte memory space, as the processor appears in Real Address mode. The Pod can only enter RUN UUT from an address in Real Address Mode (0000 0000 to 000F FFFE) or in the top 64k bytes of the 16M byte memory space (00FF 0000 to 00FF FFFE). RUN UUT operation in other address areas is possible if UUT software makes the switch to Protected Mode.

Whenever any RUN UUT starting address other than the default address (FF FFF0) is specified, the Pod considers the offset address (the IP register contents) to be equal to the four least-significant digits, and the CS register contents to equal the fifth hexadecimal digit of the address followed by three zeros (as shown in Table 3-2, RUN UUT Type 2).

When you select RUN UUT, you may either use the default execution address or you may explicitly specify the address where execution begins. The Pod Function addresses that pertain to the RUN UUT mode are listed in Table 3-2.

Using the Default Execution Address 3-12.

If the RUN UUT mode is selected and you do not specify the execution address, the Mainframe supplies the address 00FF FFF0 as the execution address. Execution at this address sets the segments registers to their reset values: 0000 for the ES, SS, and DS registers, and F000 for the CS register; the offset address (IP register) is set to FFF0. This emulates execution from a hardware reset in the UUT.

Specifying the RUN UUT Address 3-13.

You may select the address where RUN UUT execution begins by using one of three methods: by changing the default RUN UUT entry address using the Mainframe Setup command, by specifying the address in the RUN UUT command line, or by using Special Function addresses within the Pod.

You may change the RUN UUT default address with the Mainframe Setup function by entering the desired address for the Setup message *SET-RUN UUT @ FFFFF0-CHANGE?* The address to be specified must be within the range of 00 0000 to 0F FFFF or FF 0000 to FF FFFF.

You may specify the start address of RUN UUT by entering the address on the command line RUN UUT @. The address to be specified must be within the range of 00 0000 to 0 FFFE or FF 0000 to FF FFFE.

Using the Special Function address F000 XXXX for RUN UUT causes RUN UUT to begin with the segment registers initialized to user-specified values. Using this method, you can specify RUN UUT to begin anywhere in the lower 1M of memory only, without the segment registers being reset when RUN UUT begins. For example, to perform a RUN UUT at A BCDE, first load the CS register with A000 by a WRITE @ F000 0130=A000, then specify a RUN UUT using the F000 XXXX address, RUN UUT

@ F000 BCDE. The bits in the CS register are shifted left and carried to the most significant four bits of the address.

```
┌──1010 0000 0000 0000◄─┐  1011 1100 1101 1110    (binary)
│   A    0    0    0    │   B    C    D    E      (hex)
└───────────────────────┘
```

becomes

```
0000 0000 0000 1010 1011 1100 1101 1110    (binary)
 0    0    0    A    B    C    D    E      (hex)
```

Table 3-2. 9000-Series RUN UUT Control Addresses

FUNCTION	ADDRESS	DESCRIPTION
RUN UUT Type 1 (Using the Default Execution Address)	FF FFF0	This is the RUN UUT default address that the Pod supplies if the operator does not specify an execution address. RUN UUT execution at FF FFF0 sets all segment registers to their reset values (0000 for ES, SS, and DS, and F000 for CS) and sets the offset (IP register) to FFF0.
RUN UUT Type 2 (Specifying the Execution Address — bottom 1M bytes)	Y XXXX	The RUN UUT execution address may be specified by entering the address from the Mainframe setup function as Y XXXX, where the CS register equals Y000, and the offset (IP register) is XXXX. This allows you to begin operation anywhere in the bottom 1M byte of UUT memory.
RUN UUT Type 3 (Specifying the Execution Address — top 64k bytes)	FF XXXX	The RUN UUT execution address may be specified by entering the address from the Mainframe setup function as FF XXXX, where the upper four address bits are set to F, the CS register equals F000, and the offset (IP register) is XXXX. This allows you to begin operation anywhere in the upper 64k bytes of UUT memory.
RUN UUT Type 4 (Specifying the Special Function Address)	F000 XXXX	Explicitly load the CS register using special address F000 0130 and then RUN UUT @ F000 XXXX, where the resulting address is the CS register shifted left 4 bits and the offset (IP register) is XXXX.

NOTE

The Segment Registers may be defined prior to a Type 4 RUN UUT by writing the data to the following Pod Function addresses:

F000 0130 = CS register
F000 0132 = DS register
F000 0134 = SS register
F000 0136 = ES register

After the desired values are written to the above addresses, execute RUN UUT at the address F000 XXXX, where XXXX is the desired offset address. The specified contents are loaded into the segment registers. As usual, the RUN UUT execution addresses are formed using the CS register: CS register contents are shifted left four bits, then added to the offset for the execution address.

Specifying Segment Register Contents 3-14.

The contents of any or all of the segment registers may be specified before RUN UUT execution begins. To specify the segment register contents, use the following procedure:

1. Write the desired 16-bit value to the following Pod Function addresses before selecting RUN UUT.

   F000 0130 = CS register initial contents (default=F000)
   F000 0132 = DS register initial contents (default=0000)
   F000 0134 = SS register initial contents (default=0000)
   F000 0136 = ES register initial contents (default=0000)

   Read operations may be performed at these Pod Function addresses to confirm that they contain the desired values.

2. After the desired values are written to the above addresses, specify RUN UUT at the address F000 XXXX (XXXX equals the offset address). When RUN UUT begins, the initial contents of the segment registers equal the values at the Pod Function addresses.

NOTE

During execution of RUN UUT, no information is passed back to these Pod Function addresses; even though the segment register contents change, the values at the Pod Function addresses are unchanged by RUN UUT. As usual, the RUN UUT execution addresses are formed using the CS register contents and the offset address. (The value in the CS register is shifted left four bits and then added to the offset address.)

USING BREAKPOINTS 3-15.

The 80286H Pod implements hardware breakpoints as an optional means of terminating the RUN UUT operation. A "break" occurs in the RUN UUT operation when the address on the 80286 external physical address bus matches the user-specified break address. Once the break occurs, control is transferred to the Mainframe.

Enabling Breakpoints 3-16.

Breaks are enabled by writing a non-zero value to the Pod function address F000 00B0. This address is readable and writable. Only bit 0 of this address is changed during a Write operation by the user. Any command that tries to write a nonzero value to any other bits is accepted without displaying an error message, but the bits remain unchanged. Breaks are disabled by writing a 0 to bit 0 of F000 00B0. If breaks are disabled and then reenabled without changing the break address, the break address remains the same.

Setting the Break Address 3-17.

The breakpoint address is specified by the contents of the Pod's function addresses F000 00B2 and F000 00B4. Only one break address can be in effect at a time. The format for entering the break address is:

ADDRESS	FORMAT	CONTENTS
F000 00B2	YYYY	Lower two bytes
F000 00B4	00XX	Upper byte

F000 00B2 contains the lower two bytes for the break address; F000 00B4 contains the upper byte (the complete break address is given as 00XX YYYY). The upper byte of address F000 00B4 is always 00. Any command to write a nonzero value to the upper byte of F000 00B4 is accepted without displaying an error message, but the byte is not modified.

A break only occurs upon instruction fetch cycles. As soon as the physical address of the 80286 matches the address contained in F000 00B2 and F000 00B4 (and breaks are enabled), the break occurs. Since an instruction fetch is always a word or upper byte access, the break ignores the A0 bit.

Once a break occurs, the Pod resumes standby reads to the UUT (unless standby reads are disabled). The 80286 register contents before the break are lost and cannot be recovered. The contents of the Overlay RAM remain intact and can be read.

NOTE

Due to pipelined instruction fetches, actual execution may be several bytes behind the current instructionfetch. Therefore, a break could occur improperly if the breakpoint is set for a point close to, but just after a jump or call instruction. Even though the breakpoint is not executed, it may be prefetched and cause a break to occur. To prevent this type of error, position the breakpoint after the destination of a jump or call instruction.

To confirm that a break has occurred, perform a Read at address F000 00B6. This address is set to 0000 upon entering RUN UUT. After a breakpoint occurs, address F000 00B6 is set to BBBB.

USING THE OVERLAY RAM 3-18.

Overlay RAM is 8k bytes of contiguous RAM contained in the Pod. Overlay RAM may be substituted for the UUT memory to execute machine code in RUN UUT to test the UUT. You may select the starting address of the Overlay RAM by writing to a special address. Another special address to enable READY from either the Pod or the UUT is also available.

Whenever the Overlay RAM is mapped to begin at a particular address and is enabled, a read within the Overlay RAM address range returns the data in the Overlay RAM rather than the UUT memory. Overlay RAM can be mapped to any memory address location, even UUT ROM space. Breakpoints can also be used in conjunction with Overlay RAM to exit a test routine.

Overlay RAM can be located at any 8k byte boundary in memory. If Overlay RAM is used together with RUN UUT outside the Real Address mode range (0000 0000 to 000F FFFE) or the top 64k bytes of the 16M byte memory space (00FF 000 to 00FF FFFE), UUT software must make the switch to Protected Mode.

For example, assume that UUT self-test software normally prints out a long report as part of a fault diagnosis. To speed up hardware checkout, you wrote a small software routine that avoids the printout. You just want the UUT to sound the beeper and stop operation instead. When it sounds, you plan to press the UUT reset switch and start the test over again. The following test sequence carries out these steps.

1. Set the Overlay RAM starting address.
2. Enable Overlay RAM.
3. Load the test routine into Overlay RAM.
4. Set the breakpoint address at the end of the test routine.
5. Enable the breakpoint.
6. Enter RUN UUT.

The buzzer sounds when the processor executes the test routine, and the Pod exits RUN UUT.

Setting the starting address, enabling, and loading the Overlay RAM are described in the following paragraphs.

NOTE

Although Overlay RAM is logically mapped into the UUT's address space, physically it is not in the UUT. Because of this, unexpected results could occur. No data read from the Overlay RAM is present on any of the UUT's data buses and could cause a problem for any UUT circuits that depend on the processor data (such as an 80287 math coprocessor).

Enabling Overlay RAM 3-19.

When power is applied to the Pod, the Overlay RAM is disabled (default) and does not interact with the UUT in any way. (The Pod does not perform a RUN UUT out of the Overlay RAM until the Overlay RAM is enabled.) To enable the Overlay RAM, write any non-zero value to address F000 00A0, such as

```
WRITE @ F000 00A0 = 01
```

Once enabled, the Overlay RAM is defined by the Pod as 8k bytes of memory beginning at either the default address or the address specified by the user.

Setting the Overlay RAM Start Address 3-20.

The default start address for Overlay RAM is FF E000, which puts the Overlay RAM into the microprocessor power-up reset ROM area.

To change the address area covered by the Overlay RAM, new data must be written to address F000 00A2. The beginning of Overlay RAM is entered as the 11 most-significant bits of data at address F000 00A2 (D5 through D15). D0 through D4 are ignored. The data entered at this address is then converted to the 11 most-significant bits of the Overlay RAM start address. Because Overlay RAM starts only at 8k byte boundaries, A0 through A12 are set to 0. (This is an increment of 2000 hex, e.g., 78000, 22000, FE000). For example, if you want to set the starting address to FF E000 (the default address), you have to enter the data as:

WRITE @ F000 00A2 = FFE0

or as illustrated below in binary and hex:

	DATA ENTERED AT F000 00A2	START ADDRESS OF ORAM
Binary:	1111 1111 1110 0000	1111 1111 1110 0000 0000 0000
Hex:	FFE0	FF E000

Once the starting address is moved and Overlay RAM is enabled, any UUT access at the addresses within the space assigned to Overlay RAM is mapped to the Overlay RAM. Also, any Read or Write function from the Mainframe that accesses the Overlay RAM space is mapped to Overlay RAM.

Setting the Source of $\overline{\text{READY}}$ for Overlay RAM 3-21.

The source of the $\overline{\text{READY}}$ signal to the 80286 may be selected by changing the data at address F000 00A4. If this address contains a zero, $\overline{\text{READY}}$ is enabled from the UUT. If F000 00A4 contains any non-zero number, the Pod generates the $\overline{\text{READY}}$ signal and the Overlay RAM executes with zero wait states. The default for this address is 0000, or $\overline{\text{READY}}$ enabled from the UUT.

NOTE

Synchronization problems could occur if the Pod is using the Overlay RAM as a substitute for UUT RAM but $\overline{\text{READY}}$ from the UUT is not enabled within the Pod. If the UUT tries to insert wait states (while the Overlay RAM is running with no wait states), the Pod and the UUT will no longer be synchronized.

USING *HyperTEST* FUNCTIONS 3-22.

The Pod performs two *HyperTEST* functions that test various components of the UUT faster than similar tests executed by specific Mainframe function keys or quick tests. These functions reside in the Pod rather than in the Mainframe and are accessed by reading or writing to special addresses outside the Pod's standard address space. The *HyperTEST* functions test RAM and ROM on the UUT by performing a high-speed evaluation of blocks of RAM or ROM memory.

Since the software routines that control these operations reside in the Pod and not in the Mainframe, you select these functions by writing to and reading from Pod Function addresses in the format shown below. The normal 24-bit address and Pod control bits are used to specify the physical address and access type. In addition to those 7 hexadecimal digits, another hex digit is placed in the eighth hex digit of the address to denote which *HyperTEST* is used. This process is described in detail in the following paragraphs.

NOTE

Data located in Overlay RAM may be changed by the HyperRAM or HyperROM tests.

NOTE

The default configuration for the Overlay RAM $\overline{READY}$ source has $\overline{READY}$ generated by the UUT. If synchronization problems occur during the HyperRAM or HyperROM tests, the $\overline{READY}$ source may need to be changed. For more information, see the heading "Setting the Source of $\overline{READY}$ for Overlay RAM" in this section.

NOTE

HyperTEST functions are sensitive to the state of the forcing and interrupt status lines listed in Table 4-3. If the UUT asserts these signals during a HyperTEST, the test may abort. If a HyperTEST aborts, try to prevent the UUT from asserting the forcing signal or use an alternate memory test function.

HyperRAM Test 3-23.

The HyperRAM test allows you to check RAM address blocks more quickly than you can by using the Mainframe's RAM SHORT test or the Quick RAM test. The HyperRAM test is particularly well suited for programming applications, because read/write and pattern errors do not interrupt the execution of a program, as they would if you used the Mainframe's RAM tests.

Addresses to be used with a HyperRAM test are defined by placing a "6" in the eighth hex digit of the address (6SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations the Pod uses to control its built-in tests and functions.

Use the following procedure to perform a HyperRAM test on a section of the UUT's RAM memory:

1. Define the starting address by a WRITE @ 6SXX XXXX=0, where SXX XXXX is the first address to be tested by the HyperRAM test. (The address must be even in a word address space.) The valid address range is 0 to FF DFFC (word space) or FF DFFE (byte space).

2. Define the ending address and begin the test by a single WRITE @ 6SYY YYYY=0001, where:

S	Is the address space being tested. S may be either 0 = memory word 2 = memory byte
YY YYYY	Is the desired ending address. The ending address must be greater than the starting address. The address limit is FF DFFE (word space) or FF DFFF (byte space).

The HyperRAM test begins execution as soon as you complete the entry of the ending address. During and after execution of the test, the Mainframe does not display any information about the progress or results of the test unless you request it. The test may be aborted before completion by selecting another operation.

NOTE

The starting and ending address must be within the same address space. Otherwise the test aborts and records an address error.

To determine if the HyperRAM test is still in progress, or what the test results are, you should perform a Read at the ending address. (Press READ @ ENTER to command a READ operation at the ending address.)

The Mainframe displays a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-3.

Two other parameters are available to characterize the HyperRAM test: DELAY and SEED. Any changes to these values are effective for the next test performed.

1. The DELAY value determines the delay between the end of a write pass to RAM and a subsequent read from the RAM. The delay checks the refresh on dynamic RAMs. Each unit of delay roughly equals 10,000 periods of the effective processor clock signal. This means that 1 unit of delay approximates 1 millisecond for an effective UUT clock of 10 MHz with no wait states.

   Set the delay value by a WRITE @ F000 6016.

   The DELAY value has a range from 0 to FFFF. The default value is FA, or 250 milliseconds for a clock of 10 MHz with no wait states.

2. The HyperRAM test writes a sequence of randomly generated data words to memory. The SEED value determines how the data is generated. If SEED is zero, the sequence of random data words is different each time the test is invoked. This procedure is always recommended. If SEED is not zero, for a given SEED value, the sequence of random data words is the same for each test of the memory.

   Set the SEED value by a WRITE @ F000 6018.

   The SEED value has a range from 0 to FFFF. The default value is 0.

Several read-only Pod Function addresses in the F000 60XX range contain additional information about the HyperRAM test, including records of addresses used and any errors detected. The Pod Function addresses for the HyperRAM test are described in Table 3-3.

NOTE

Unless the test has been started, the information contained at these Pod Function addresses pertains to a previous test rather than to the current test. Trying to read any of these addresses (or performing any other operation except a READ @ ENTER) while the current test is in progress aborts the test.

The following example demonstrates the procedure used to specify a HyperRAM test. Assume that the UUT has 4K of RAM memory, from address 1 5000 through 1 5FFF. The HyperRAM specification would look like this:

```
WRITE @ 6001 5000=0
WRITE @ 6001 5FFE=1
```

Line 1 defines the starting address to be 1 5000 in a word address space. Line 2 defines the ending address to be 1 5FFE. It also causes the test to start. DELAY and SEED are unchanged.

To monitor the test results and look for any error codes that might result from the test do:

```
READ @ ENTER
```

Table 3-3. HyperRAM Test

ACTION	MEANING
WRITE @ 6SXX XXXX = 0	SXX XXXX = starting address.
WRITE @ 6SYY YYYY = 0001	S = address space 0 = memory word 2 = memory byte YY YYYY = ending address
WRITE @ F000 6016 = XXXX	XXXX = DELAY value
WRITE @ F000 6018 = XXXX	XXXX = SEED value
READ @ ENTER	Returns status as follows:

CODE	MEANING
0000	No test requested
00A0	Aborted, new command entered
00A1	Aborted, illegal data in cmd
00A2	Aborted, illegal adr in cmd
00A3	Aborted, Pod timeout occurred
00B0	Busy, performing rd/wr check
00B1	Busy, performing adr dcd check
00B2	Busy, performing pattern verify
00C0	Complete, no errors
00F0	Failed, read/write error
00F1	Failed, address decoding error
00F2	pattern verify error

READ @ F000 6000	Low word of starting address
READ @ F000 6002	High word of starting address
READ @ F000 6004	Low word of ending address
READ @ F000 6006	High word of ending address
READ @ F000 6008	Low word of error address
READ @ F000 600A	High word of error address
READ @ F000 600C	Data expected at error address
READ @ F000 600E	Actual data returned from error adr
READ @ F000 6010	Returns completion code
READ @ F000 6012	Hex mask of error bits
READ @ F000 6016	Returns DELAY value
READ @ F000 6018	Returns SEED value

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses in the F000 60XX range. For example, you can find the low word of the address where the error occurred by:

READ @ F000 6008

You can get a hex mask of any bad data bits by:

READ @ F000 6012

The information from all of the other Pod Function addresses is retrieved in the same manner.

HyperROM Test 3-24.

The HyperROM test allows you to test ROM address blocks more quickly than you can by using the Mainframe's built-in ROM test. The HyperROM test has an extensive error-reporting capability that can detect inactive data bits, and a signature test that can detect a faulty ROM device with a high degree of confidence.

The HyperROM test works much like the HyperRAM test described above. Addresses to be used with a HyperROM test are defined by placing a "7" in the eighth hex digit space of the address (7SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations that the Pod uses to control its built-in tests and functions.

Use the following procedure to perform a HyperROM test on a section of the UUT's ROM memory:

1. Define the starting address by a WRITE @ 7SXX XXXX=0, where SXX XXXX is the first address to be used by the HyperROM test (must be even in a word address space). The valid address range is 2000 to FF FFFC (word space) or 2000 to FF FFFE (byte space).

2. Define the ending address and address increment by a single WRITE @ 7SYY YYYY=ZZZ1, where:

S	Is the address space being tested. S may be either 0 = memory word 2 = memory byte
YY YYYY	Is the desired ending address. The ending address must be greater than the starting address.
ZZZ	Is the desired increment. If Z is omitted or specified as 0, the address increment defaults to 2. (Must be even in a word address space.)
1	Denotes "Ending Address".

The HyperROM test begins execution as soon as you complete the entry of the ending address. During and after execution of the test, the Mainframe does not display any information about the progress or results of the test unless you request it. The test may be aborted before completion by selecting another operation.

To determine if the HyperROM test is still in progress, or what the test results are, you should perform a Read at the ending address. (Press READ @ ENTER to command a READ operation at the ending address.)

The Mainframe displays a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-4.

A mask parameter is also available to characterize the HyperROM test. Masking allows you to be selective about the signature gathered. For example, if the UUT has a 16-bit data bus and two 8-bit ROM components, you can gather a signature for just one component by masking half he bus - the half written to by the other component. Bit positions corresponding to "1" in MASK are tested. Set the MASK value by a WRITE @ F000 7016. The range of MASK is 0 to FF (byte space) or 0 to FFFF (word space). The default value is all bits tested.

Table 3-4. HyperROM Test

ACTION	MEANING	
WRITE @ 7SXX XXXX = 0	SXX XXXX = starting address.	
WRITE @ 7SYY YYYY = ZZZ1	S = address space 0 = memory word 2 = memory byte YY YYYY = ending address ZZZ = increment 1 = byte addresses 0,2 = word addresses.	
WRITE @ F000 7016 = XXXX	XXXX = MASK value	
READ @ ENTER	Returns status as follows:	
	CODE	**MEANING**
	0000	No test requested
	00A0	Aborted, new command entered
	00A1	Aborted, illegal data in cmd
	00A2	Aborted, illegal adr in cmd
	00A3	Aborted, Pod timeout occurred
	00B0	Busy, test in progress
	00C0	Complete, no errors
	00C1	Complete, inactive bits detected
READ @ F000 7000	Low word of starting address	
READ @ F000 7002	High word of starting address	
READ @ F000 7004	Low word of ending address	
READ @ F000 7006	High word of ending address	
READ @ F000 700C	Signature	
READ @ F000 700E	Hex mask of bits detected as inactive	
READ @ F000 7010	Returns completion code	
READ @ F000 7014	Returns address increment and function	
READ @ F000 7016	Returns MASK value	

Several read-only Pod Function addresses in the F000 70XX range contain additional information about the HyperROM test, including records of addresses used and any errors detected. The Pod Function addresses for the HyperROM test are described in Table 3-4.

NOTE

Unless the test has been started, the information contained at the Pod Function addresses pertains to a previous test rather than to the current test. Trying to read any of the Pod Function addresses (or perform any other operation except a Read at the last test address) while the current test is in progress aborts the current test.

The following example demonstrates the procedure to conduct a HyperROM test. Assume that a UUT has 8K words of ROM memory, from F C000 through F FFFF, using four 4K X 8 ROM chips. To test properly, you need to have a separate test for each ROM device. This means that you'll have to divide the total address block in half, and test both upper and lower bytes separately in each of these half blocks. This makes two address ranges: F C000 through F DFFF and F E000 through F FFFF. To test the lower byte in the first block, you would perform the following operations:

```
WRITE @ 720F C000=0
WRITE @ 720F DFFE=21
```

Line 1 defines the starting address of the Quick ROM test to be F C000, with byte accesses. Line 2 defines the ending address to be F DFFE, with byte accesses, and the test to be done with an increment of 2 so as to access only the lower (even) bytes. The Write in Line 2 also starts the test.

To monitor the test results and look for any error codes that might result from the test, press

```
READ @ ENTER
```

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses (shown in Table 3-4).

To test the upper byte in the same address range, you would use a similar command, but start the test at the odd address.

```
WRITE @ 720F C001=0
WRITE @ 720F DFFF=21
```

This tests the remaining ROM chip that provides the odd byte. As before, you can monitor the test results and look for any error codes that might result from the test by pressing

```
READ @ ENTER
```

NOTE

The only error that can be reported is stuck bits. The test always returns a signature value to address F000 700C, but there is no indication whether this is the correct value. You must compare this calculated value with a known good value to determine if the HyperROM test passed.

USING THE QUICK TESTS AND FUNCTIONS 3-25.

The Pod performs five quick functions that test various components of the UUT faster than similar tests executed by specific Mainframe function keys. These functions reside in the Pod rather than in the Mainframe and are accessed by reading or writing to special addresses outside the Pod's standard address space. The quick functions test various components on the UUT by looping on read or write, performing a high-speed evaluation of blocks of RAM or ROM memory, and activating a ramp function to create signatures for a fast signature analysis of data signals on the UUT. Oscilloscope displays may be enhanced by synchronizing the Mainframe probe to various signals on the UUT and running the Pod's Quick Looping function.

TESTING RAM QUICKLY 3-26.

The Pod provides several Quick Tests and Functions that allow you to conduct high-speed tests to verify RAM memory in the UUT. They allow you to do simple Read/Write and Pattern Verify tests on large portions of memory. The diagnostics performed by the Pod during the execution of these Quick functions are less rigorous than the diagnostics performed during the execution of the Mainframe's built-in tests. The Quick Tests and Functions should be used when high speed testing is worth some sacrifice in thoroughness. Each function is described below.

Since the software routines that control these operations reside in the Pod and not in the Mainframe like the other tests and functions, you select these functions by writing to and reading from Pod Function addresses in the format shown below. The normal 24-bit address and Pod control bits are used to specify the physical address and access type. In addition to those 7 hexadecimal digits, another hex digit is placed in the eighth hex digit of the address to denote which of the Quick Functions or Tests are to be used. This process is described in detail in the following paragraphs.

The same process is used to form the addresses for controlling the Quick ROM test (see Testing ROM Quickly) and the Quick Looping test (see Using the Pod with an Oscilloscope).

The Quick RAM Test 3-27.

The Quick RAM Test allows you to test RAM address blocks more quickly than you can by using the Mainframe's RAM Short test. The Quick RAM Test is considerably faster than the RAM Short test and is almost as rigorous. The Quick RAM test is particularly well suited for programming applications, because read/write and pattern errors do not interrupt the execution of a program, as they would if you used the Mainframe's RAM tests.

The Quick RAM Test is available in two variations: the normal RAM test and a Pattern Verification test.

- The normal RAM test consists of two phases: the first test phase is a read-write check, and the second test checks address decoding. The read-write check is performed by writing and reading a 1 and a 0 to and from each bit of each test address to ensure that there are no bits held high or low. After the read-write check is completed, a unique bit pattern is written to each address. For the address decoding check, the Pod reads each address and compares the read data with the unique word that is expected.

- The Pattern Verification test simply verifies that memory still contains the data that was last written in the Quick RAM test. Because dynamic RAM may often retain data for a long time (as much as a minute) without being refreshed, the Pattern Verification test is provided to check that the memory refresh is working and that the memory is retaining data. If problems with dynamic RAM are suspected, begin execution of the pattern verification test no sooner than a minute after a Quick RAM test has been performed.

Addresses to be used with a Quick RAM test are defined by placing a "2" in the eighth hex digit of the address (2SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations the Pod uses to control its built-in tests and functions.

Use the following procedure to perform a Quick RAM test on a section of the UUT's RAM memory:

1. Define the starting address by a WRITE @ 2SXX XXXX=0, where SXX XXXX is the first address to be tested by the Quick RAM test. (The address must be even in a word address space.)

2. Define the ending address, address increment, and test specification by a single WRITE @ 2SYY YYYY=ZZZW, where:

S	Is the address space being tested. S may be either 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte
YY YYYY	Is the desired ending address. The ending address must be greater than the starting address.
ZZZ	Is the desired increment. If ZZZ is omitted or specified as 0, the address increment defaults to 2. (Must be an even value in a word address space.)
W	Is the test specification. W may be either 1 = normal Quick RAM test 2 = Pattern Verification test

The Quick RAM Test begins execution as soon as you complete the entry of the ending address. During and after execution of the test, the Mainframe does not display any information about the progress or results of the test unless you request it. The test may be aborted before completion by selecting another operation.

NOTE

The starting and ending addresses must be within the same type of space. For example, unknown results could occur if the starting address of the Quick RAM Test is in the UUT RAM address space and the ending address is in I/O address space.

To determine if the Quick RAM Test is still in progress, or what the test results are, you should perform a Read at the ending address. (Press READ @ ENTER to command a READ operation at the ending address.)

The Mainframe displays a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-5.

Several read-only Pod Function addresses in the F000 20XX range contain additional information about the Quick RAM Test, including records of addresses used and any errors detected. The Pod Function addresses for the Quick RAM test are described in Table 3-5.

Table 3-5. Quick RAM Test

ACTION	MEANING
WRITE @ 2SXX XXXX = 0	SXX XXXX = starting address.
WRITE @ 2SYY YYYY = ZZZW	S = address space 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte YY YYYY = ending address ZZZ = increment 1 = byte addresses 0, 2 = word addresses W = function type 1 = RAM test 2 = Pattern Verify
READ @ ENTER	Returns status as follows:

CODE	MEANING
0000	No test requested
00A0	Aborted, new command entered
00A1	Aborted, illegal data in cmd
00A2	Aborted, illegal adr in cmd
00A3	Aborted, Pod timeout occurred
00B0	Busy, performing rd/wr check
00B1	Busy, performing adr dcd check
00B2	Busy, performing pattern verify
00C0	Complete, no errors
00F0	Failed, read/write error
00F1	Failed, address decoding error
00F2	Failed, pattern verify error

ACTION	MEANING
READ @ F000 2000	Low word of starting address
READ @ F000 2002	High word of starting address
READ @ F000 2004	Low word of ending address
READ @ F000 2006	High word of ending address
READ @ F000 2008	Low word of error address
READ @ F000 200A	High word of error address
READ @ F000 200C	Data expected at error address
READ @ F000 200E	Actual data returned from error adr
READ @ F000 2010	Returns most recent code returned
READ @ F000 2012	Hex mask of error bits
READ @ F000 2014	Returns increment and function type

NOTE

Unless the test has been started, the information contained at these Pod Function addresses pertains to a previous test rather than to the current test. Trying to read any of these addresses (or doing any other operation except a Read at the last test address) while the current test is in progress aborts the test.

The following example demonstrates the procedure used to specify a Quick RAM test. Assume that the UUT has 4k of RAM memory, from address 1 5000 through 1 5FFF. To test the entire 4k of RAM, it would be best to use word accesses. The Quick RAM specification would look like this:

```
WRITE @ 2001 5000=0
WRITE @ 2001 5FFE=1 (or =21)
```

Line 1 defines the starting address to be 1 5000. Line 2 defines the ending address to be 1 5FFE and the test to be a full RAM test (as opposed to the Pattern Verification test) with an increment of 2 (by default). It also causes the test to start.

To monitor the test results and look for any error codes that might result from the test do:

```
READ @ ENTER
```

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses in the F000 20XX range. For example, you can find the low word of the address where the error occurred by:

```
READ @ F000 2008
```

You can get a hex mask of any bad data bits by:

```
READ @ F000 2012
```

The information from all of the other Pod Function addresses is retrieved in the same manner.

To check the refresh function in this same section of memory, wait a short time to allow the memory to decay (if the refresh is not working) then perform a Pattern Verification test. To begin, enter

```
WRITE @ 2001 5FFE=2 (or =22)
```

The specification for the starting address remains the same as the previous Quick RAM test and does not need to be repeated. This new line specifies the same ending address, only this time using the Pattern Verification test. As before, you can press

```
READ @ ENTER
```

to check the progress of the test and, afterwards, the corresponding Pod Function address again contains other information about the test.

The Quick Fill and Quick Verify Functions 3-28.

The Quick Fill and Quick Verify functions allow you to fill blocks of memory with the same value of data and then verify the accuracy of the contents. The Quick Fill function works much faster than the equivalent operation programmed from a series of Mainframe Write commands. The Quick Fill and Quick Verify functions allow you to customize special memory diagnostics, such as might be desirable when testing a memory-mapped video display. You could, for example, just fill the video memory with a known data value, then simply look for errors on the UUT's video display.

The Quick Fill and Quick Verify functions may be used individually or they may be specified to work together in one step.

The Quick Fill and Quick Verify functions are controlled by writing setup information into Pod Function addresses in the same manner as the Quick RAM test described above.

- The Quick Fill function writes the data that is contained in the starting address to all of the addresses in the block.
- The Quick Verify function reads data from all of the addresses in the block and compares each one to the data contained in the starting address. Errors are reported via the Pod Function addresses described below.

Addresses used with a Quick Fill and Verify function are defined by placing a "4" into the eighth hex digit of the address (4SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations that the Pod uses to control its built-in tests and functions.

Use the following procedure to perform one of the Fill and Verify functions on a section of the UUT's RAM memory:

1. Specify the data to be used by a WRITE @ SXX XXXX = DDDD, where:

S	Is the address space being tested. S may be either 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte
XX XXXX	Is the starting address. (The address must be even in a word address space.)
DDDD	Is the data to be filled throughout the defined address block.

NOTE

Step 1 is only used with the Quick Fill function.

2. Define the starting address by a WRITE @ 4SXX XXXX=0, where XX XXXX is the starting address. (Must be even in a word address space.)

3. Define the ending address, address increment, and test specification by a single WRITE @ 4SYY YYYY=ZZZW, where:

S	Is the address space being tested. S may be either 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte

YY YYYY	Is the desired ending address. The ending address must be greater than the starting address.
ZZZ	Is the desired increment. If ZZZ is omitted or specified as 0, the address increment defaults to 2. (Must be even in a word address space.)
W	Is the test specification. W may be either 1 = Quick Fill function 2 = Quick Verify function 3 = Quick Fill and Quick Verify functions

The Quick Fill and Quick Verify functions begin execution as soon as you complete the entry of the ending address. During and after execution of the functions, the Mainframe does not display any information about the progress or results of the functions unless you request it. The functions may be aborted before completion by selecting another operation.

NOTE

If you specify a beginning address that is not readable or writable, the Quick Fill or Quick Verify functions may write or read incorrect data at the remaining addresses.

NOTE

The starting and ending address must be within the same type of space. For example, unknown results could occur if the starting address of the Quick RAM Test is in the UUT RAM address space and the ending address is in I/O address space.

To determine if the Quick Fill or Quick Verify function is still in progress, or what the results are, you should perform a Read at the ending address. (Press READ @ ENTER to command a READ operation at the last-entered address.)

The Mainframe displays a code indicating the status of the test or the results. The status codes and their meanings are shown in Table 3-6.

Several read-only Pod Function addresses in the F000 40XX range contain additional information about the Quick Fill and Quick Verify function, including records of addresses used and errors. The Pod Function addresses for the Quick Fill and Quick Verify functions are also described in Table 3-6.

NOTE

Unless the function has been started, the information contained at the Pod Function addresses pertains to a previous test rather than to the current function. Trying to read any of these addresses (or doing any other operation except a Read at the last test address) while the current function is in progress aborts the current function.

Table 3-6. Quick Fill and Verify Function

ACTION	MEANING
WRITE @ 4SXX XXXX = 0	SXX XXXX = starting address.
WRITE @ 4SYY YYYY = ZZZW	S = address space 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte YY YYYY = ending address ZZZ = increment 1 = byte addresses 0, 2 = word addresses W = function type 1 = Fill 2 = Verify 3 = Fill then Verify
READ @ ENTER	Returns status as follows:

CODE	MEANING
0000	No test requested
00A0	Aborted, new command entered
00A1	Aborted, illegal data in cmd
00A2	Aborted, illegal adr in cmd
00A3	Aborted, Pod timeout occurred
00B0	Busy, performing fill
00B1	Busy, performing verify
00C0	Complete, no errors
00F0	Error, data does not match

READ @ F000 4000	Low word of starting address
READ @ F000 4002	High word of starting address
READ @ F000 4004	Low word of ending address
READ @ F000 4006	High word of ending address
READ @ F000 4008	Low word of error address
READ @ F000 400A	High word of error address
READ @ F000 400C	Data written by fill
READ @ F000 400E	Actual data returned from error address
READ @ F000 4010	Returns most recent code
READ @ F000 4012	Hex mask of error bits
READ @ F000 4014	Returns increment and function type

The following example demonstrates the procedure used to conduct a combined Quick Fill and Quick Verify function. To specify the functions over the RAM addresses 1 2000 through 1 2FFF with the default address increment of 2 and data 5555, do the following three operations:

```
WRITE @ 1 2000=5555
WRITE @ 4001 2000=0
WRITE @ 4001 2FFE=3
```

Line 1 inserts the data to be used into the first address. Line 2 defines the starting address to be 1 2000. Line 3 defines the ending address to be 1 2FFE and the test to be a combined Quick Fill and Quick Verify function with an increment of 2 (by default). The Write operation in Line 3 also starts the function.

To monitor the test results and look for any error codes that might result from the the test, press

```
READ @ ENTER
```

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses (as shown in Table 3-6).

TESTING ROM QUICKLY 3-29.

The Quick ROM Test allows you to test ROM address blocks more quickly than you can by using the Mainframe's built-in ROM Test. The Quick ROM Test is not as rigorous and reliable as the signature analysis used by the Mainframe's built-in ROM Test, nor does the Quick ROM Test have as extensive an error-reporting capability. However, the Quick ROM Test can detect inactive data bits, and the signature can be used to detect a faulty ROM device with a high degree of confidence.

The Quick ROM Test works much like the Quick RAM Test described above. Addresses to be used with a Quick ROM test are defined by placing a "3" in the eighth hex digit space of the address (3SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations that the Pod uses to control its built-in tests and functions.

Use the following procedure to perform a Quick ROM test on a section of the UUT's ROM memory:

1. Define the starting address by a WRITE @ 3SXX XXXX=0, where SXX XXXX is the first address to be used by the Quick ROM test. (Must be even in a word address space.)

2. Define the ending address and address increment by a single WRITE @ 3SYY YYYY=ZZZ1, where:

S	Is the address space being tested. S may be either 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte 4 = instruction word

YY YYYY	Is the desired ending address. The ending address must be greater than the starting address.
ZZZ	Is the desired increment. If Z is omitted or specified as 0, the address increment defaults to 2. (Must be even in a word address space.)
1	Denotes "Ending Address".

The Quick ROM Test begins execution as soon as you complete the entry of the ending address. During and after execution of the test, the Mainframe does not display any information about the progress or results of the test unless you request it. The test may be aborted before completion by selecting another operation.

To determine if the Quick ROM Test is still in progress, or what the test results are, you should perform a Read at the ending address. (Press READ @ ENTER to command a READ operation at the ending address.)

The Mainframe displays a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-7.

Several read-only Pod Function addresses in the F000 30XX range contain additional information about the Quick ROM Test, including records of addresses used and any errors detected. The Pod Function addresses for the Quick ROM test are described in Table 3-7.

NOTE

Unless the test has been started, the information contained at the Pod Function addresses pertains to a previous test rather than to the current test. Trying to read any of the Pod Function addresses (or perform any other operation except a Read at the last test address) while the current test is in progress aborts the current test.

The following example demonstrates the procedure to conduct a Quick ROM Test. Assume that a UUT has 8k words of ROM memory, from F C000 through F FFFF, using four 4k X 8 ROM chips. To test properly, you need to have a separate test for each ROM device. This means that you must divide the total address block in half, and test both upper and lower bytes separately in each of these half blocks. This makes two address ranges: F C000 through F DFFF and F E000 through F FFFF. To test the lower byte in the first block, you would perform the following operations:

```
WRITE @ 320F C000=0
WRITE @ 320F DFFE=21
```

Line 1 defines the starting address of the Quick ROM test to be F C000, with byte accesses. Line 2 defines the ending address to be F DFFE, with byte accesses, and the test to be done with an increment of 2 so as to access only the lower (even) bytes. The Write in Line 2 also starts the test.

To monitor the test results and look for any error codes that might result from the test, press

```
READ @ ENTER
```

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses (shown in Table 3-7).

To test the upper byte in the same address range, you would use a similar command, but start the test at the odd address.

```
WRITE @ 320F C001=0
WRITE @ 320F DFFF=21
```

This tests the remaining ROM chip that provides the odd byte. As before, you can monitor the test results and look for any error codes that might result from the test by pressing

```
READ @ ENTER
```

Table 3-7. Quick ROM Test

ACTION	MEANING
WRITE @ 3SXX XXXX = 0	SXX XXXX = starting address.
WRITE @ 3SYY YYYY = ZZZ1	S = address space 0 = memory word 1 = I/O word 2 = memory byte 3 = I/O byte 4 = instruction word YY YYYY = ending address ZZZ = increment 1 = byte addresses 0, 2 = word addresses
READ @ ENTER	Returns status as follows: **CODE** — **MEANING** 0000 — No test requested 00A0 — Aborted, new command entered 00A1 — Aborted, illegal data in cmd 00A2 — Aborted, illegal adr in cmd 00A3 — Aborted, Pod timeout occurred 00B0 — Busy, test in progress 00C0 — Complete, no errors 00C1 — Complete, inactive bits detected
READ @ F000 3000	Low word of starting address
READ @ F000 3002	High word of starting address
READ @ F000 3004	Low word of ending address
READ @ F000 3006	High word of ending address
READ @ F000 300C	Signature
READ @ F000 300E	Hex mask of bits detected as inactive
READ @ F000 3010	Returns most recent code
READ @ F000 3014	Returns address increment and function

NOTE

The only error that can be reported is stuck bits. The test always returns a signature value to address F000 300C, but there is no indication whether this is the correct value. You must compare this calculated value with a known good value to determine if the Quick ROM test passed.

USING THE QUICK RAMP FUNCTION 3-30.

Introduction 3-31.

The Quick Ramp function allows you to do a rapid write of data, ramping from 0000 to FFFF at a single memory address. Like the quick memory tests described above, the Quick Ramp function is similar to the Mainframe's built-in tests, only much faster. The Quick Ramp function is controlled by writing setup information into Pod Function addresses in the same manner as the other quick tests and functions.

Addresses to be used with a Quick Ramp function are defined by placing a "5" into the eighth digit of the address (5SXX XXXX). Such addresses are not physical addresses in the UUT's address space; they are special read/write operations that the Pod uses to control its built-in tests and functions.

Use the following single-step procedure to perform a Quick Ramp function on an address in the UUT's RAM memory:

1. Define the address by a WRITE @ 5SXX XXXX=0, where:

S	Is the address space being tested. S may be either 0 = memory word 1 = I/O word
XX XXXX	Is the address to be used with the Quick Ramp Function.

NOTE

The Quick Ramp function is illegal at a byte address. The function defaults to the next lowest address if a byte address is specified.

The Quick Ramp function begins execution as soon as you complete the entry. The function begins by writing the data value 0000 to the specified address. The Function follows with 65,535 write operations, incrementing the data each time until the data equals FFFF. During and after execution of the function, the Mainframe does not display any information about the progress or results of the function unless you request it. The function may be aborted before completion by selecting another operation.

To determine if the Quick Ramp function is still in progress, or what the results are, you should perform a Read at the last entered address. (Press READ @ ENTER to command a READ operation at the last entered address.)

The Mainframe displays a code indicating the status of the function. The status codes and their meanings are shown in Table 3-8.

Table 3-8. Quick Ramp Function

ACTION	MEANING
WRITE @ 5SXX XXXX = ZZZZ	S = address space 0 = memory word 1 = I/O word XX XXXX = Ramp address ZZZZ = any data
READ @ ENTER	Returns status as follows:

CODE	MEANING
0000	No test requested
00A0	Aborted, new command entered
00A2	Aborted, illegal adr in cmd
00A3	Aborted, Pod timeout occurred
00B0	Busy, performing ramp
00C0	Complete

ACTION	MEANING
READ @ F000 5000	Low word of address
READ @ F000 5002	High word of address
READ @ F000 5010	Returns most recent code

Several read-only Pod Function addresses in the F000 50XX range contain additional information about the Quick Ramp function. The Pod Function addresses for the Quick Ramp function are also described in Table 3-8.

NOTE

Unless the function has been started, the information contained at the special addresses pertains to a previous test rather than the current function. Trying to read any of these addresses (or doing any other operation except a Read at the last test address) while the current function is in progress aborts the current function.

The following example demonstrates the simple procedure used to perform a Ramp function. To specify a Ramp function at address 4 2F0A, do the following operation:

WRITE @ 5004 2F0A=0

To see if the Ramp function has been completed, or if it has aborted, press

READ @ ENTER

Generating Probe Signatures 3-32.

The Quick Ramp function is commonly used as a stimulus for creating probe signatures. The Mainframe Operator manual contains complete information on how to conduct a signature analysis investigation on your UUT. The basic procedure using the Pod's Quick Ramp function is as follows:

1. Press PROBE SYNC D to select the data sync mode for the probe.

2. Probe the selected data-related signal on the UUT.

3. Press READ PROBE to clear the probe.

4. Do the Quick Ramp function using the procedure indicated above.

5. Press READ PROBE to read the probe.

6. Compare the displayed signature with the expected signature.

USING THE POD WITH AN OSCILLOSCOPE 3-33.

When you use an oscilloscope in conjunction with a Mainframe and Pod to investigate a UUT, you should be familiar with two Pod-specific items: the Quick-Looping Function and the Synchronization modes.

The primary use of the Pod's built in Quick-Looping function is to increase the repetition rate of the oscilloscope display to enhance its brightness. The following description under the heading "Using the Quick-Looping Function" provides instructions for utilizing this feature.

The Mainframe's Trigger Output can be used to synchronize your oscilloscope with specific cycles of Pod operations. The Probe and Synchronization Modes topic below describes the synchronization modes that are provided by the 80286H Pod for the Mainframe's probe as well as for an oscilloscope input.

Using the Quick-Looping Function 3-34.

The Quick-Looping read or write function is used primarily to brighten the display on an oscilloscope that is synchronized to the TRIGGER OUTPUT pulse (on the Mainframe's rear panel). When looping on a Mainframe function, the signal trace on the oscilloscope screen is dim because of a low repetition rate; the Quick-Looping function can increase the repetition rate to make the signal trace much more visible.

To select the Quick-Looping function at address SXX XXXX, place a "1" in the eighth hex digit of the address so that it becomes 1SXX XXXX. The Pod first performs a read or write operation at address SXX XXXX in the normal manner, reporting to the Mainframe any UUT system errors that might be detected (such as *ACTIVE FORCE LINE*, or *CTL ERR*, etc.). Then, the Pod enters the Quick Looping mode. Read or write operations in the Quick Looping mode are performed several times faster than the ordinary Looping mode that is specified by pressing the LOOP key on the Mainframe keyboard. During Quick Looping, the Pod does not check for any UUT system errors that may occur. Quick Looping continues until the operator selects another operation.

For example, if the operator specifies the operation READ @ 1200 F000, the Pod performs a Quick-Looping Read operation at the address 200 F000 (a byte access). If the operator specifies the operation WRITE @ 1100 B007 = 2F, the Pod performs a Quick-Looping Write operation at address 100 B007 (an I/O access), writing the data 2F.

The Quick-Looping function may be used with read or write operations at any valid address. The Quick-Looping function is not intended to be used with any of the other troubleshooting functions or tests.

If both error reporting and the Quick Looping feature are desired, you may combine the Mainframe Looping function with the Pod's Quick-Looping read or write, such as

READ @ 1SXX XXXX LOOP. The slow rate of the Mainframe Loop function provides periodic updates to allow error reporting. When the Mainframe is not polling the Pod, the Pod's Quick Loop Test runs at full speed to exercise the UUT. For every ordinary UUT operation, the Pod interjects a few Quick-Looping read or write operations (with no error reporting), which enhances oscilloscope viewing.

Probe and Scope Synchronization Modes 3-35.

You may use the Mainframe's Synchronization function (selected with the SYNC key) to synchronize both probe operation and the rear panel TRIGGER OUTPUT pulse (for an oscilloscope) to events on the Pod's buses. The five synchronization modes that are available and their Mainframe selection codes are:

A = Address Sync
B = Bus Cycle Sync
D = Data Sync
1 = Interrupt Acknowledge Sync
F = Free-Run Sync

The main purpose of the various synchronization modes is to provide proper signal timing for use with the probe. They can also be used to simplify viewing signals with an oscilloscope.

ADDRESS SYNC 3-36.

If you select the Address Sync, both the probe and scope trigger output are synchronized to the address portion of the UUT access. The scope trigger pulses low shortly before the UUT access begins, and pulses high at the end of the address portion of the UUT-access cycle. If you select the probe stimulus mode, the probe pulses at the selected level (high or low) for the time between the two scope trigger pulses described above. For probe response, the probe latches on the signal level present at its tip at the time of the second or high-going scope trigger pulse.

BUS CYCLE SYNC 3-37.

If you select Bus Cycle Sync, the scope trigger pulses low at the start of UUT access (when $\overline{S0}$ or $\overline{S1}$ is asserted). The scope trigger pulses high at the end of the data cycle (the trailing edge of the Data Enable pulse).

DATA SYNC 3-38.

If you select Data Sync, the scope trigger pulses low at the start of the data portion of the UUT access (the end of the address portion). The scope trigger pulses high at the end of the data cycle (the trailing edge of the Data Enable pulse).

INTERRUPT-ACKNOWLEDGE SYNC 3-39.

If you select Interrupt-Acknowledge Sync, the scope trigger pulses low at the start of the interrupt-acknowledge cycle, and pulses high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle.

NOTE

The interrupt-acknowledge sync mode is selected by the "1" key on the Mainframe.

FREE-RUN SYNC 3-40.

Free-Run Sync only applies to the Mainframe's probe; it does not affect the scope trigger output. If you select free-run mode, the probe stimulus pulses are generated at a frequency of approximately 1 kHz with a 1% duty cycle. The scope trigger output pulses remain synchronized to whatever other sync mode that may have been selected previously (either address, data, or interrupt acknowledge), even if free-run is selected.

At power-on, the probe is in the free-run mode, and the scope trigger output pulses are switched off.

CONFIGURING GENERAL POD CHARACTERISTICS 3-41.

Several built-in characteristics of the Pod provide default addresses and other functions for your convenience. These characteristics have been predefined to accommodate a majority of UUTs, but they can be changed to adapt the Pod to your unique requirements.

The Pod characteristics are changed by writing to Pod function addresses in the Pod. The individual functions are described in the following paragraphs.

Changing the Standby Read Address (ADDRESS F000 0002) 3-42.

To provide UUTs with memory accesses that refresh dynamic memory devices, the Pod does repetitive standby or "transparent" read operations. When the UUT is idle (not doing a RUN UUT operation), its normal methods for refreshing memory may not work. These transparent read operations serve as a substitute to keep the UUT's memory functioning correctly.

If the default address provided by the Pod does not access RAM on your UUT, or if that particular address should not be read (because, for example, it might cause an unwanted operation on the UUT), then you may change the address that is used. You may specify the location used for these transparent read operations by writing to Pod Function address F000 0002. (This same address also performs non-specific read operations that are used in other operations. See the Theory of Operation in Section 5.)

The default value for Pod Function address F000 0002 is FFFF. A value of FFFF sets the default standby address to FF FF00. The least significant 8 bits (D0 through D7) of the standby read address are always zero.

The contents of F000 0002 may be read to view the current value for the standby read address.

NOTE

Pod operations can be made to execute faster by selecting a standby read address that causes zero wait states to be inserted into the standby read cycles.

Enabling the Standby Read Address 3-43.

The standby read function may be enabled or disabled by writing to address F000 0006. A non-zero value in F000 0006 enables the standby read address. A value of zero disables the standby read address. The default value of F000 0006 is 0001, or enabled.

NOTE

Standby reads are disabled while an active RESET signal is present on the UUT.

MASKING ERRORS 3-44.

You can use several masks to control how the Pod and Mainframe report errors. If you wish to suppress the reporting of individual errors, possibly because a particular error is recurring or may be the result of a design quirk in the UUT, you may instruct the Pod and Mainframe to continue without stopping or displaying an error message when that error occurs. You can use this capability to avoid getting stuck at certain errors during troubleshooting.

Like the other Pod functions, these masks are used by writing to and reading from Pod Function addresses. For all of these masks, a "0" in a bit position signifies that the error assigned to that bit space is not reported. Details of the individual masks are described in the following paragraphs.

The default value for all of the masks is no bits masked (all errors reported). The mask data remains intact until it is changed by writing a new value or by removing and restoring power to the Pod.

Error Summary Mask (ADDRESS F000 0060) 3-45.

The Error Summary Mask address is intended to support operation with 9000-Series Mainframes only. After each Pod/UUT operation, the Pod sends a byte (8 bits) of information to the 9000-Series Mainframe to indicate possible error conditions or significant changes in status.

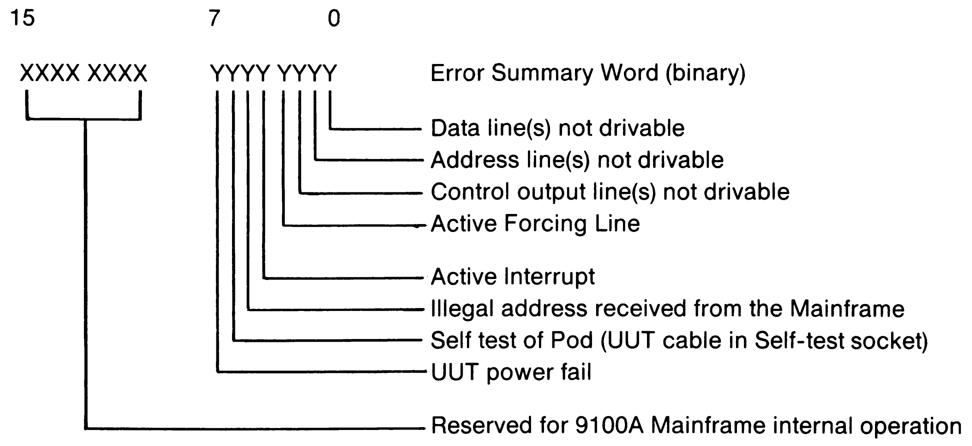
The Mainframe uses this information to generate the correct response to each condition. For example, if the UUT Power Fail bit is set, the Mainframe displays a Bad Power Supply message. (The Power Fail bit is set by the Pod if the voltage at one or both of the power supply pins at the UUT's microprocessor socket is not between +4.5V dc and +5.5V dc, or if the Vss pins are not a ground potential.)

A Write of data XXYY to address F000 0060 masks the corresponding bits in the error-summary word from being reported to the Mainframe. A "0" in the bit position means that the corresponding bit in the error summary byte is to be masked (ignored during error reporting).

NOTE

The data bits denoted by Y represent the 9000-Series Mainframe error-summary byte bits. The most-significant bits denoted by X are ignored by the 9000-Series Mainframe.

The individual bits in the error-summary mask are assigned as follows:


15
7
0
XXXX XXXX
YYYY YYYY
Error Summary Word (binary)
Data line(s) not drivable
Address line(s) not drivable
Control output line(s) not drivable
Active Forcing Line
Active Interrupt
Illegal address received from the Mainframe
Self test of Pod (UUT cable in Self-test socket)
UUT power fail
Reserved for 9100A Mainframe internal operation

For example, a WRITE at F000 0060=007F indicates that you want the Pod and Mainframe to ignore the occurrence of an apparent Bad Power Supply. You may want to mask this error if you are troubleshooting a UUT that intentionally uses a lower power supply voltage. If you do not mask this error, the Pod and Mainframe always stop and report the Bad Power Supply and do not allow you to do other tests.

The power-up default value of the data at address F000 0060 is FFFF; all fault reporting enabled.

A Read at address F000 0060 returns the hex value of the mask as last programmed.

Control Drivability Error Mask (ADDRESS F000 0062) 3-46.

A Write to address F000 0062 masks individual control line bit errors from being reported to the Mainframe. The data corresponds to the control line bit assignments in Table 4-4. A data bit value of 0 masks the corresponding control bit. The power-up default mask is 00FF. A Read at this address returns the current mask.

Forcing Line and Interrupt Error Mask (ADDRESS F000 0064) 3-47.

Data written to address F000 0064 individually masks forcing lines and interrupt lines from being reported as active. The lines that are masked correspond to any data bits that are specified as 0 in the bit assignments described below.

BIT	STATUS LINE
0	$\overline{\text{READY}}$
1	HOLD
2	NMI
3	INTR
4	RESET
5	PEREQ
6	$\overline{\text{BUSY}}$
7	$\overline{\text{ERROR}}$
8	CAP FAIL

Data bits 9 through 15 are ignored and normally set to 0. The power-up default mask is 01FF (no lines masked).

A Read at this address returns the current mask.

High Address Drivability Error Mask (ADDRESS F000 0068) 3-48.

A Write at address F000 0068 masks drivability errors on the high byte (A16 through A23) of the UUT's address bus. The data bits written to D8 through D15 are ignored. The address bits are assigned to the data in this Pod Function address as follows:

```
Address bits:                            A23 ............ A16
Data bit mask:      D15 ......... D8     D7 ............ D0
                    |_____________|
                           |
                           |__________ Ignored
```

The Default value is 00FF.

Low Address Drivability Error Mask (ADDRESS F000 006A) 3-49.

A Write at address F000 006A masks drivability errors on the low word (A0 through A15) of the UUT's address bus. The address bits are assigned to the data at this Pod Function address as follows:

```
Address bits:      A15 ............... A0
Data bit mask:     D15 ............... D0
```

For example, a Write at F000 006A = FDFF instructs the Pod and Mainframe not to report drivability problems on address line A9 if they occur.

The Default value is FFFF.

Data Drivability Error Mask (ADDRESS F000 006C) 3-50.

A Write at address F000 006C masks the map of the UUT's data bus bits. The data bits are assigned to the data in this Pod Function address as follows:

```
Data bits:         D15.............. D0
Data bit mask:     D15.............. D0
```

The power-up default condition for this mask is FFFF (errors on all lines reported). A Read at address F000 006C returns the current mask.

Miscellaneous Line Error Mask (ADDRESS F000 006E) 3-51.

The Miscellaneous Line Error Mask masks individual pseudo-status lines from being reported to the Mainframe. The lines that are masked correspond to any data bits that are specified as 0 in the bit assignments described below.

BIT	STATUS LINE
10	PWR FAIL 30
11	PWR FAIL 62
12	GND FAIL 35
13	GND FAIL 9

Data bits 0 through 9, 14, and 15 are ignored and normally set to 0. The power-up default mask is 3C00 (no lines masked).

DETERMINING ERRORS 3-52.

You may want to know whether errors have occurred after a particular operation, such as a Write, even if you have disabled the reporting of those errors (see the heading "Masking Errors" in this section). The Pod provides several "last error" Pod Function addresses that contain the results of the possible error conditions. By reading the contents of these last error addresses, you can locate specific information about the most recent error.

The information at the last error Pod Function addresses (with the exception of the Last Error Summary at address F000 0040) is not changed until another operation that causes the Pod to execute a UUT-access cycle (a Read or a Write at a UUT address, for example) is performed. The Error Summary Mask (F000 0060) disables portions of the Pod's error testing program so that the Pod executes tests faster when certain errors are not of interest. When the "Last Error Summary" address is read, the Pod temporarily reenables all error testing and recalculates errors incurred on the last UUT access. This operation also updates the other "last error" special addresses.

NOTE

Operations such as Bus Test should not rely on the "Last Error Summary" to accurately report all errors, since only errors that occurred on the last UUT access can be reported. Bus Test relies on fault information from a multitude of UUT accesses in order to accurately diagnose UUT errors.

NOTE

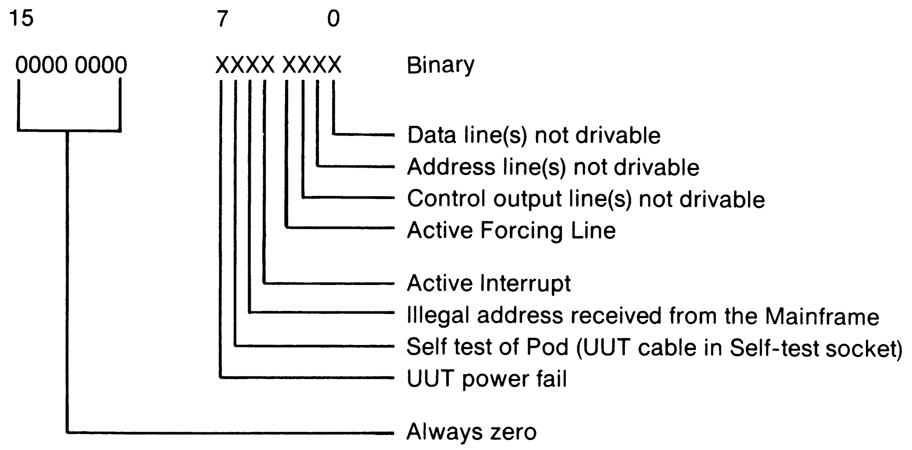
Reading the last Error Summary contents at address F000 0040 clears all its bits. None of the contents of other "last error" Pod Function addresses is changed if they are read.

The information for the individual Pod Function addresses is described in the following paragraphs.

Last Error Summary (ADDRESS F000 0040) 3-53.

After each Pod/UUT operation, the Pod sends a byte of information to the Mainframe to indicate possible error conditions or significant changes in the state of the Pod. The Mainframe uses this information to generate the correct response to each condition. For example, if the Forcing Lines Pending and Disabled bit is set, the Mainframe displays an *ACTIVE FORCE LINE* message.

A Read at address F000 0040 returns the hex value of the Error Summary byte, with the individual bits assigned as follows:


15
7
0
0000 0000
XXXX XXXX
Binary
Data line(s) not drivable
Address line(s) not drivable
Control output line(s) not drivable
Active Forcing Line
Active Interrupt
Illegal address received from the Mainframe
Self test of Pod (UUT cable in Self-test socket)
UUT power fail
Always zero

For example, READ @ F000 0040=0010 indicates that an interrupt line was asserted during the most recent UUT access.

The data returned by a Read at this address is NOT affected by the condition of the Error Summary Mask.

Last Control Errors (ADDRESS F000 0042) 3-54.

A Read at address F000 0042 returns the hex representation of the control errors for the most recent Pod/UUT operation. The bit assignments for in this address are the same as those used for the control lines in other functions. Refer to Table 4-4 in Section 4 for a description of the Control Line Bit Assignments.

Last Forcing Line Errors and Active Interrupts (ADDRESS F000 0044) 3-55.

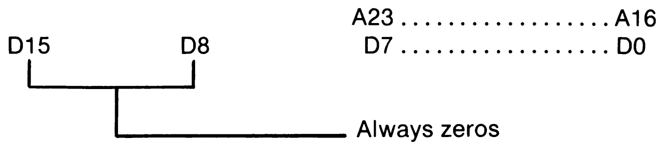
A Read at address F000 0044 returns the hex representation of any Forcing Line errors or Active Interrupts that may have occurred during the most recent operation. The lines that were active correspond to any data bits that are specified as 1 in the bit assignments described below.

BIT	STATUS LINE
0	$\overline{\text{READY}}$
1	HOLD
2	NMI
3	INTR
4	RESET
5	PEREQ
6	$\overline{\text{BUSY}}$
7	$\overline{\text{ERROR}}$
8	CAP FAIL

Data bits 9 through 15 are ignored and normally set to 0.

Last High Byte Address Drivability Errors (ADDRESS F000 0048) 3-56.

A Read at address F000 0048 returns a map of bits that identifies drivability errors on the high byte (A16 through A23) of the UUT's address bus. The address bits are assigned to the data at this Pod Function Address as follows:



A bit that is set high indicates a drivability error on the corresponding address line. For example, READ @ F000 0048=0002 shows that a drivability error occurred on address line A17.

Last Low Word Address Drivability Errors (ADDRESS F000 004A) 3-57.

A Read at address F000 004A returns a map of bits that identifies drivability errors on the low word of the UUT's address bus. The address bits are assigned to the data in this Pod Function address as follows:

Address bits:	A15 .... A0
Data bit mask:	D15 .... D0

A bit that is set high indicates a drivability error on the corresponding address line. For example, READ @ F000 004A=0020 shows that a drivability error occurred on address line A5.

Last Data Drivability Errors (ADDRESS F000 004C) 3-58.

A Read at address F000 004C returns a map of bits that identifies drivability on the UUT's data bus. The data bits are assigned to the data in this Pod Function address as follows:

Data bits:	D15 .... D0
Data bit mask:	D15 .... D0

A bit that is set high indicates a drivability error on the corresponding data line. For example, READ @ F000 004C=0401 shows that drivability errors occurred on data lines D0 and D10.

Last Status (ADDRESS F000 0052) 3-59.

A Read at address F000 0052 returns the status signal values for the most recent Pod operation. The normal status bit assignments as shown in Table 4-3 are used.

ACTIVE FORCING LINES 3-60.

Forcing lines are a special category of status lines that, when asserted, could force the UUT's microprocessor into some specific state or action. When the Pod is plugged into the UUT's microprocessor socket in place of the microprocessor, the Mainframe reports activity on these lines with an active Forcing line error message.

The forcing lines of the 80286H Pod consist of CAP FAIL, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, PEREQ, RESET, HOLD, and $\overline{\text{READY}}$.

You may disable the reporting of active forcing lines by selecting the Mainframe Setup function message *SET-TRAP ACTIVE FORCE LINE? NO.* The Pod still monitors the lines, but the Mainframe does not interrupt its operation to display the *ACTIVE FORCE LINE* error message on the display. Sometimes it is useful to disable the reporting of active forcing lines, particularly if the information is not needed by the operator.

USER-ENABLEABLE FORCING LINES 3-61.

The 80286 has two status lines that are individually enabled or disabled using the Mainframe's Setup function: $\overline{\text{READY}}$ and HOLD. Pressing the YES key on the Mainframe enables the status line; pressing the NO key disables the status line.

If these user-enableable lines are disabled (using the Mainframe Setup function), their inputs to the Pod microprocessor are disabled, but the Pod continues to monitor their condition. If the user-enableable lines are asserted, the Pod reports to the Mainframe that a forcing line is active.

During Mainframe Setup, selecting the message *SET-ENABLE xxxxxx? NO* prevents the designated line from affecting the operation of the Pod (although the Pod still detects whether the line is high or low). This differs from selecting the Mainframe Setup message *SET-TRAP ACTIVE FORCE LINE? NO* which does not prevent an enable line from affecting the operation of the microprocessor, but does prevent the active condition from being reported on the Mainframe display.

NOTE

While you generally want to test your UUT with the enableable lines enabled (so that wait states and alternate bus masters are properly handled), you may improve Pod execution speed in some cases if the enableable lines are disabled (eliminating unnecessary wait states or hold acknowledge states).

WRITING CONTROL LINES 3-62.

The 80286H Pod has three control lines that the Mainframe can drive high or low with the WRITE CTL function: $\overline{\text{LOCK}}$, $\overline{\text{PEACK}}$, and HLDA. This feature is used by BUS TEST to check lines that cannot be toggled by normal read and write operations. Writing to control lines also helps troubleshoot the control lines.

The Write Control and Data Toggle Control require the entry of binary digits to define user-writable control lines. When using either of these two functions, the Mainframe display prompts the operator for a binary number to identify the control line(s) to be written.

For example, to perform a Write Control operation that writes all three user-writable control lines high, the operator enters WRITE @ CTL = 111. To write any of the lines to the low state, the operator enters a 0 in place of 1 at the bit position that corresponds to the particular control line.

NOTE

Two of the writable control lines are active low (bit 0, $\overline{LOCK}$ and bit 2, $\overline{PEACK}$), and one is active high (bit 1, HLDA). To drive the three lines to their active states during the WRITE @ CTL operation, use WRITE @ CTL = 10.

NOTE

The Write Control function sets a line high or low for only one UUT access, just long enough to verify that the line can be driven.

For more information on the control lines, see the heading "Control Lines" in Section 4 of this manual.

Section 4
Pod Reference Data

INTRODUCTION 4-1.

This section describes how to configure the Pod for use with a specific UUT and how to use Pod functions that do not involve accesses to the UUT.

The Pod provides Pod Function addresses to allow you to set up the Pod to the same configuration as the microprocessor that it replaces. Occasionally, some functions (such as interrupt) need to be set before you can access components on the UUT with Read and Write operations.

Other Pod Function addresses allow you to get information from the Pod about the most recent UUT access, such as status and error information.

Section 4 contains the following topics:

Addresses	Contains information on the address configuration of the Pod.
Information About Pod Signals	Lists all the 80286 microprocessor signals and provides a description of how the Pod handles each line.
Status and Control Lines	Describes the Status and Control lines and explains how they provide information for diagnosing UUT faults.

NOTE

In this manual, the terms Status and Control differ in meaning from the same terms used in the microprocessor manufacturer's literature. The Mainframe considers input lines to the microprocessor to be Status lines and output lines to be Control lines.

USING THE POD 4-2.

Once the Pod is connected to the Mainframe and the UUT, and the Pod has been configured to your UUT as described in the previous sections, you can begin using the Mainframe/Pod team to investigate 80286 UUTs.

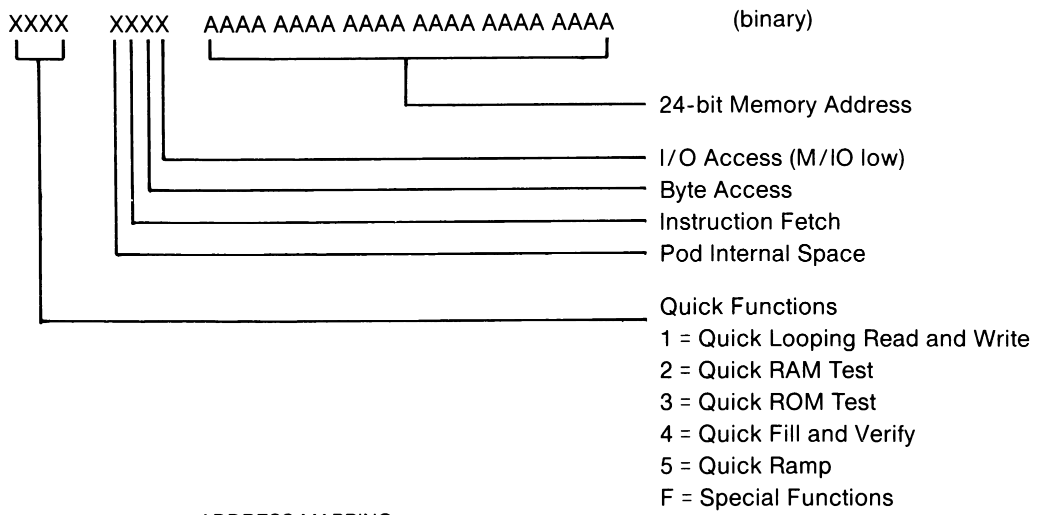
All of the standard diagnostic procedures that are built into the Mainframe work in the usual manner (such as Bus Test, Read, Write, etc.). In addition to these functions, there are many additional functions that are built into the Pod. These Pod functions allow you to do such things as perform high-speed memory tests, set up internal registers and masks, check for details about specific errors, alter default values of various parameters, and manipulate interrupt activity.

Together with the Mainframe's tests, Pod functions provide an extensive capability for diagnosing UUT defects. Some Pod functions are implemented in the form of Pod Function addresses. Pod Function addresses are read/write locations that are not in the memory space of the UUT; they access diagnostic addresses within the Pod instead. The Pod Function Addresses are constructed with the usual 24-bit address information, plus additional bits to indicate specific Pod functions. More detailed information is located further on in this section under the heading "Pod Function Addresses".

Many of the Pod functions use and return information about the microprocessor's Status and Control lines. The Pod has all of these lines (and a few Pod-generated status and control lines) mapped into standard Status and Control words. The bit assignments for these words are described in this section, and are also listed on the decal on the bottom of the Pod. You'll probably use the information frequently, so it might pay to take the time to become familiar with it.

ADDRESSES 4-3.

Two types of addresses are used with the 80286H Pod: UUT access addresses and Pod Function addresses. UUT-access addresses are used to Write data to or Read data from memory or I/O devices on the UUT. Pod Function addresses allow you to use the many additional functions that the Pod provides in addition to those that are built into the Mainframe. Figure 4-1 shows the general address structure that is used with the Pod. Information about accessing UUT addresses and Pod Function addresses is contained in the following paragraphs.



ADDRESS MAPPING

NAME	WORD ADDRESSES	BYTE ADDRESSES
MEMORY	0- FF FFFE	200 0000 - 2FF FFFF
I/O	100 0000 - 100 FFFE	300 0000 - 300 FFFF
INSTRUCTION	400 0000 - 4FF FFFE	—

Figure 4-1. Addresses Used with the 80286 Pod

UUT Addresses 4-4.

UUT addresses consist of two parts: a single hex digit to determine the UUT address space to be accessed, and a 24-bit (6 hex digit) physical address. The individual bits of the address modifier denote whether the address is Memory or I/O, and Byte or Word access. Any time that a Mainframe Read or Write operation is used with a UUT address, the operation must consist of all seven digits of information.

PHYSICAL ADDRESSES 4-5.

Physical address locations on the UUT may be either memory devices or I/O devices.

Memory Addresses 4-6.

The 80286 can address either words (16 bits of data) or bytes (8 bits of data). Each physical address is a byte address.

The 80286H Pod accepts only even addresses for word accesses; odd addresses default to the next lower (even) address. To access a word at an odd address, the operator must actually use two consecutive byte accesses. During even-word accesses, the $\overline{\text{BHE}}$ (Bus High Enable) line is low (active), which enables the high byte at the same time as the low byte.

Memory accesses are specified by an Address Control Bit (A24).

I/O Addresses 4-7.

The 80286 can address up to 64k (65,536) 8-bit ports or 32k (32,768) 16-bit ports. The I/O space is not segmented; to access a port address, the 80286 places the address on the lower 16 lines of the address bus. Address lines A16 through A23 are output as zeros. I/O addresses are specified by an Address Control Bit (A24).

ADDRESS CONTROL BITS 4-8.

Four bits (one hex digit) of additional information are added to the normal 24-bit physical memory address to denote specific information about UUT accesses. If this hex digit is not specified, the Pod provides a default value. The default value is 0000 (binary); normal, memory, word accesses.

Memory/I O Addressing 4-9.

The Memory/IO bit (A24) determines whether the address is a memory reference or an operation with an IO port. If A24 is low (the default), the access is to memory (M/$\overline{\text{IO}}$ is high). If A24 is high, the access is to an I/O device (M/$\overline{\text{IO}}$ is low).

NOTE

Only the lower 16 bits of the physical address are used for I/O operations. A16 through A23 are output low, and must be specified as zeros in the address.

For example, a Write at address 100 00C4 = XXXX specifies an I/O word access at location C4.

Word/Byte Addressing 4-10.

The Word/Byte bit (A25) indicates whether the remainder of the address is a byte access or a word access. When A25 is high, the access is to a byte address. When A25 is low (the default), the access is to a word address. For example, a Write at address 201 9AA1 = XX specifies a byte access to memory location 1 9AA1.

For byte addresses at an even address, $\overline{BHE}$ = 1 and A0 = 0; for byte addresses at an odd address, $\overline{BHE}$ = 0 and A0 = 1.

For word addresses, $\overline{BHE}$ = 0 and A0 = 0.

NOTE

Word-access addresses should be even (LSB=0). If an odd address is specified for a word access, the Mainframe displays an illegal address message. The Pod still performs the access, however, and the uneven word-access address is mapped to the next lowest word address. For example, a Write at address 00 4321 = XX actually writes to location 00 4320.

Instruction Fetch 4-11.

The instruction fetch bit (A26) returns the executable opcode from the address specified in the Read command. For example, a Read at address 400 BCDE returns two bytes of data at address BCDE. The COD/$\overline{INTA}$ line is driven high for the duration of the access.

Pod Internal Space 4-12.

A27 is used for troubleshooting the Pod and may only be enabled when self test is disabled. For more information on this Pod function address space, refer to Section 5.

Pod Function Addresses 4-13.

Pod Function Addresses are read/write locations that are not in the memory space of the UUT; they access diagnostic functions of the Pod. The Pod Function addresses themselves are not sent to the UUT; they are only a means to access the Pod functions.

These additional Pod functions are provided to allow you to set up internal registers and masks, to check for specific errors, to alter default values of various parameters, to manipulate interrupt activity, and to perform high-speed tests of memory. Table 4-1 shows the available Pod Function addresses.

QUICK TEST AND FUNCTION ADDRESSES 4-14.

Three bits of the address (A28 through A30) indicate that one of the Pod's Quick Tests is to be used with the indicated physical address.

For example, a Write at address 3XXX XXXX is an operation that is part of performing a Quick ROM Test.

If no Quick Test is wanted, the Quick Test Prefix (A28 through A30) may be left blank and the default will be 0 (no test). For example, a Write at XXX XXXX = XXXX is OK.

For more information on the the Pod's Quick Tests, see "Using the Quick Tests and Functions" in Section 3.

SPECIAL FUNCTION ADDRESSES 4-15.

When the eighth digit of the address (bits A28 through A31) is hexadecimal F, the address is a Special Function Address used by the Pod. These Special Function addresses provide a wide variety of diagnostic functions to use in a detailed investigation of a UUT.

Table 4-1. Pod Function Addresses

ADDRESS	DESCRIPTION	
Pod Control Addresses		
F000 0000	Self-Test Result Code	(Read)
F000 0002	Standby Address	(Read/Write)
F000 0006	Standby Read Enable	(Read/Write)
F000 0012	Software Revision Level	(Read Only)
Error Reporting Addresses (Read Only)		
F000 0040	Last Error Summary	
F000 0042	Last Control Errors	
F000 0044	Last Active Forcing Line and Active Interrupts	
F000 0048	Last High Address Drivability Errors	
F000 004A	Last Low Address Drivability Errors	
F000 004C	Last Data Drivability Errors	
F000 0052	Last Status	
Error Masks (Read/Write)		
F000 0060	Error Summary Mask	
F000 0062	Control Drivability Error Mask	
F000 0064	Forcing Line and Interrupt Error Mask	
F000 0068	High Address Drivability Error Mask	
F000 006A	Low Address Drivability Error Mask	
F000 006C	Data Drivability Error Mask	
F000 006E	Miscellaneous Error Mask	
Interrupt Vector and Configuration Addresses		
F000 0080	Enable Interrupt Acknowledge Cycle	(Read/Write)
F000 0082	Read Interrupt Vector and Re-enable	(Read Only)
F000 0084	Read Interrupt Vector and do not Re-enable	(Read Only)
F000 0086	Force Interrupt Acknowledge Bus Cycle	(Read Only)
F000 0088	Quick Loop on Force INTA	(Read Only)
F000 008C	Read Cascade Address (low word)	(Read Only)
F000 008E	Cascade Address (high byte)	(Read Only)
Overlay RAM Addresses (Read/Write)		
F000 00A0	Enable Overlay RAM	
F000 00A2	Overlay RAM Base Address	
F000 00A4	Enable $\overline{\text{READY}}$ from the UUT to Overlay RAM	
Breakpoint Addresses		
F000 00B0	Enable Breakpoint	(Read/Write)
F000 00B2	Breakpoint Address (low word)	(Read/Write)
F000 00B4	Breakpoint Address (high byte)	(Read/Write)
F000 00B6	Break confirmation	(Read Only)
Segment Register Contents for RUN UUT (Read/Write)		
F000 0130	CS Register Contents	
F000 0132	DS Register Contents	
F000 0134	SS Register Contents	
F000 0136	ES Register Contents	

Pod Function Addresses are used by writing data to them to do such tasks as loading registers, defining parameters, or initiating Pod actions, or by reading them to retrieve information about the Pod and UUT. Writing data to Pod Function Address F000 0136, for example, defines the contents of the CPU's ES Register. The operation does not affect the contents of memory location 000 0136 in the UUT's memory.

All of the Pod Function Addresses are listed numerically in Table 4-1. Details of the Pod Function Addresses are contained in Sections 2 and 3 of this manual.

INFORMATION ABOUT POD SIGNALS 4-16.

Table 4-2 lists all of the 80286 microprocessor signals and provides a brief description of how each signal is handled by the Pod. Some of the lines are classified by the Pod as Status Lines, Forcing Lines, Control Lines, or User-Enableable Lines. Figure 4-2 shows the 80286H Pod and microprocessor pin assignments.

Refer to the microprocessor manufacturer's literature for detailed design-level information about the various microprocessor signals.

STATUS AND CONTROL LINES 4-17.

Pod status lines are input signals that are applied by the UUT to the UUT's microprocessor socket. Status lines are inputs that are not data, clock, nor power supply signals.

Pod control lines are output signals that are applied by the Pod to the UUT at the UUT's microprocessor socket. Control lines are not address or data signals.

The Pod creates several additional pseudo-status lines to augment the information that the Pod normally reports to the Mainframe.

The Mainframe can report activity on the status lines and drivability errors on the control lines after the Pod performs any UUT access.

The status and control lines provide much of the information that is used to diagnose defects in a UUT. These lines may be used to determine the results of tests and to simulate microprocessor operations. You should become familiar with how the status and control lines work with the Pod before using them to troubleshoot your UUTs.

NOTE

In this manual, the terms Status and Control differ in meaning from the same terms used in the microprocessor manufacturer's literature. The Mainframe considers input lines to the microprocessor to be Status lines, and output lines to be Control lines.

Figure 4-3 shows how the various lines interact between the Pod and the UUT.

Status Lines 4-18.

Most of the Status Lines are input lines to the UUT's microprocessor socket that indicate critical factors about system operation. Some additional status lines used are not real 80286 status lines, but pseudo-status bits that are generated within the Pod to indicate conditions that are important to the user. Information about status lines is displayed as a map of bits, with each bit corresponding to a status line. Status lines and pseudo-status lines are shown in Table 4-3 and on the decal on the back of the Pod.

Table 4-2. 80286 Microprocessor Signal Descriptions

SIGNAL NAME	PIN	DESCRIPTION
CLK	31	The 80286 derives the clocks needed internally by the processor from the Clock input line.
A0-A23	7-8, 10-28 32-34	These are the address signals of the 80286 microprocessor (and the Pod). The address bus on the Pod consists of 24 unidirectional tri-state output lines and provides the bus address for all processor operations. All 24 lines of the address bus are held in the tri-state condition when Hold Acknowledge is high.
D0-D15	36-51	These are the 16-bit, bidirectional, tri-state data signals of the 80286 microprocessor (and the Pod). During interrupt-acknowledge cycles, an external device sources the vector number to the Pod on lines D0 through D7. All 16 lines of the data bus are held in the tri-state condition when Hold Acknowledge is high.
$\overline{\text{BHE}}$	1	$\overline{\text{BHE}}$ (Bus High Enable) is an output signal of the 80286 microprocessor (and of the Pod) that, when asserted (low), denotes that data on the upper half of the data bus is valid. $\overline{\text{BHE}}$ is valid for the entire length of a bus cycle. $\overline{\text{BHE}}$ is reported by the Pod as a Control Line (see Table 4-4). $\overline{\text{BHE}}$ is asserted for all word accesses and for odd addresses entered in byte address spaces. $\overline{\text{BHE}}$ is held in the tri-state condition when Hold Acknowledge is high.
$\overline{\text{S0}}$ $\overline{\text{S1}}$	5 4	$\overline{\text{S0}}$ and $\overline{\text{S1}}$ are output signals of the 80286 microprocessor (and the Pod) that define (along with M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$) the type of bus cycle being performed. In general, $\overline{\text{S0}}$ asserted indicates a write cycle; $\overline{\text{S1}}$ asserted indicates a read cycle. Both $\overline{\text{S0}}$ and $\overline{\text{S1}}$ are asserted during an interrupt acknowledge cycle. Both $\overline{\text{S0}}$ and $\overline{\text{S1}}$ are held in the tri-state condition when Hold Acknowledge is high. $\overline{\text{S0}}$ and $\overline{\text{S1}}$ are reported by the Pod as Control Lines (see Table 4-4).
M/$\overline{\text{IO}}$	67	Memory/IO Select distinguishes between memory access and I/O access. M/$\overline{\text{IO}}$ is held in the tri-state condition when Hold Acknowledge is high. The Pod treats M/$\overline{\text{IO}}$ as a control line. M/$\overline{\text{IO}}$ is asserted or not asserted depending on A24 in the address field. If M/$\overline{\text{IO}}$ is high, a memory access is indicated by the user; if M/$\overline{\text{IO}}$ is low, an I/O access is indicated.
COD/$\overline{\text{INTA}}$	66	Code/Interrupt Acknowledge distinguishes between an instruction fetch bus cycle and an interrupt acknowledge bus cycle. COD/$\overline{\text{INTA}}$ is held in the tri-state condition when Hold Acknowledge is high. COD/$\overline{\text{INTA}}$ is decoded in conjunction with $\overline{\text{S0}}$, $\overline{\text{S1}}$, and M/$\overline{\text{IO}}$ to determine what cycle is taking place, as shown in the following table:

Table 4-2. 80286 Microprocessor Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
		COD/$\overline{\text{INTA}}$ | M/$\overline{\text{IO}}$ | $\overline{\text{S1}}$ | $\overline{\text{S0}}$ | BUS CYCLE INITIATED 0 | 0 | 0 | 0 | Interrupt acknowledge 0 | 1 | 0 | 0 | If A1 = 1 then halt, else shutdown 0 | 1 | 0 | 1 | Memory data read 0 | 1 | 1 | 0 | Memory data write 1 | 0 | 0 | 1 | I/O read 1 | 0 | 1 | 0 | I/O write 1 | 1 | 0 | 1 | Memory instruction read The Pod handles COD/$\overline{\text{INTA}}$ as a control output. A low in A26 of the Pod's address space selection bits indicates a regular memory cycle; a high in A26 indicates an instruction fetch. Interrupt acknowledge cycles are initiated by using the special addresses explained in Sections 2 and 3 under Testing Interrupt Circuitry.
$\overline{\text{LOCK}}$	68	When asserted (low), the $\overline{\text{LOCK}}$ output signal of the 80286 microprocessor (and the Pod) denotes that other system bus controllers may not have control over the address and data buses. $\overline{\text{LOCK}}$ is held in the tri-state condition when Hold Acknowledge is high. $\overline{\text{LOCK}}$ is reported by the Pod as a Control Line (see Table 4-4), and is user-writable.
$\overline{\text{READY}}$	63	Bus $\overline{\text{READY}}$ is an asserted-low input to the 80286 microprocessor (and the Pod). $\overline{\text{READY}}$ terminates a bus cycle on a falling edge. $\overline{\text{READY}}$ is used to insert wait states. $\overline{\text{READY}}$ is ignored when Hold Acknowledge is high. The Pod handles $\overline{\text{READY}}$ as an enableable forcing status input (see Table 4-3). If the $\overline{\text{READY}}$ input is enabled, and held or "stuck" high on the UUT, operation of the Pod can suspend, causing a Pod timeout. The $\overline{\text{READY}}$ line can be disabled from reaching the Pod's microprocessor using the SETUP function of the Mainframe. When disabled, the $\overline{\text{READY}}$ input is reported as a Forcing Line by the Pod. The Pod contains an internal 2.0k ohm pullup resistor to +5V on $\overline{\text{READY}}$.
HOLD	64	HOLD is an asserted-high input to the 80286 microprocessor (and the Pod). When asserted (high), HOLD denotes that another bus controller is requesting control of the local bus. The 80286 microprocessor responds to a HOLD input by asserting the HLDA output (see below) and by placing the microprocessor's bus into a high-impedance state. The HOLD input is treated by the Pod as an Enableable Forcing Status Line and is normally enabled to reach the Pod's microprocessor. When the HOLD input is enabled and becomes asserted,

Table 4-2. 80286 Microprocessor Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
		the Pod responds by asserting its HLDA output (see below) and by placing the Pod's external bus into a high-impedance state (see "Special Signal States" in this section). If the HOLD input is enabled, and held or "stuck" high on the UUT, operation of the Pod can suspend, causing a POD TIMEOUT. The HOLD line can be disabled from reaching the Pod's microprocessor using the SETUP function of the Mainframe. When disabled, the HOLD input is reported as a Forcing Line by the Pod (see Table 4-3).
HLDA	65	The HLDA output of the 80286 microprocessor (and the Pod) is asserted (high) to acknowledge that an external device may drive the microprocessor's bus. The HLDA output is asserted high in response to assertion of the HOLD input; HLDA then returns low shortly after HOLD is driven low. When HLDA becomes asserted, the microprocessor's bus is placed in a high-impedance state. (See Special Signal States in this section for a list of the Pod's signals that are placed in a high-impedance state in response to an asserted HLDA signal.) Note that if the HOLD input to the Pod is disabled (see above), the HLDA output is never asserted by the Pod in response to an asserted HOLD input. HLDA is reported by the Pod as a Control Line (see Table 4-4). HLDA is user-writable. NOTE The Pod's bus is NOT put into a high-impedance condition when HLDA is asserted by the Pod during a Write Control UUT access.
INTR	57	Interrupt Request requests the 80286 suspend its current operation and service a pending external request. The 80286 responds to an interrupt request by performing two interrupt-acknowledge cycles to read an interrupt vector that identifies the source of the interrupt. The Pod enables or disables interrupt acknowledge through a special address (F000 0080). The default is disabled. Interrupt Request is treated by the Pod as both enableable and as an interrupt line (see Table 4-3). If interrupts are enabled, the Pod executes an interrupt-acknowledge sequence. Then the Pod fetches an interrupt vector from the data bus and sets the INT VECT status bit. If interrupts are not enabled, then the interrupt is reported as active only.
NMI	59	In an 80286-based system, the Non-Maskable Interrupt Request is an asserted-high, non-maskable, edge-triggered interrupt to the microprocessor. Except during RUN UUT mode, NMI is always disabled from reaching the Pod's microprocessor. NMI is reported by the Pod as a Status line and as an Interrupt (see Table 4-3).

Table 4-2. 80286H Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
$\overline{\text{PEACK}}$	6	The Processor Extension Acknowledge output from the 80286 signals the processor extension that an operand has been transferred. $\overline{\text{PEACK}}$ is held in the tri-state condition when Hold Acknowledge is high. $\overline{\text{PEACK}}$ is treated by the Pod as a writable control line.
PEREQ	61	The Processor Extension Operand Request input from a coprocessor requests a data operand transfer for a processor extension. PEREQ is treated by the Pod as a non-enableable, forcing status input.
$\overline{\text{ERROR}}$	53	Processor Extension Error indicates a problem with the current bus cycle. When asserted, $\overline{\text{ERROR}}$ causes the 80286 to execute a processor extension interrupt. $\overline{\text{ERROR}}$ is treated by the Pod as a non-enableable forcing status input.
$\overline{\text{BUSY}}$	54	The Processor Extension Busy input is used by a coprocessor to stop execution of the 80286 while the processor extension completes an operation. $\overline{\text{BUSY}}$ is treated by the Pod as a non-enableable forcing status input.
RESET	29	When asserted, the RESET signal suspends all operations of the microprocessor and resets internal registers. Instruction execution does not begin again until the RESET signal is returned to a high logic level. The Pod treats the RESET signal from the UUT as a non-enableable forcing status line (see Table 4-3). Except during RUN UUT, the RESET input from the UUT is always disabled from reaching the Pod's microprocessor.
CAP	52	The microprocessor uses the CAP pin to connect to an external filter capacitor for its substrate bias generator. A 0.047 μF capacitor is used as the filter element. The capacitor is inserted between the CAP pin and ground (Vss). The Pod tests for the presence of a capacitor of approximately the correct value and signals an error if the test fails. The fault is communicated to the Mainframe as a forcing line asserted. The tolerance on the CAP detection circuit is +100%, -50%. NOTE The Pod has an internal capacitor for its own microprocessor and is not dependent on the UUT capacitor for operation.

Table 4-2. 80286 Microprocessor Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
VCC (+5V)	30,62	System power is provided by pins 30 and 62 on the 80286. The Pod monitors each Vcc line independently and reports a failure on pin 30 as PWR FAIL 30 or on pin 62 as PWR FAIL 62.
VSS (Ground)	9,35,60	Power return is provided by pins 9, 35, and 60 on the 80286. The Pod monitors pins 9 and 35 for an open condition, and reports a failure on pin 9 as GND FAIL 9 or on pin 35 as GND FAIL 35. Pin 60 is the ground reference for the Pod.

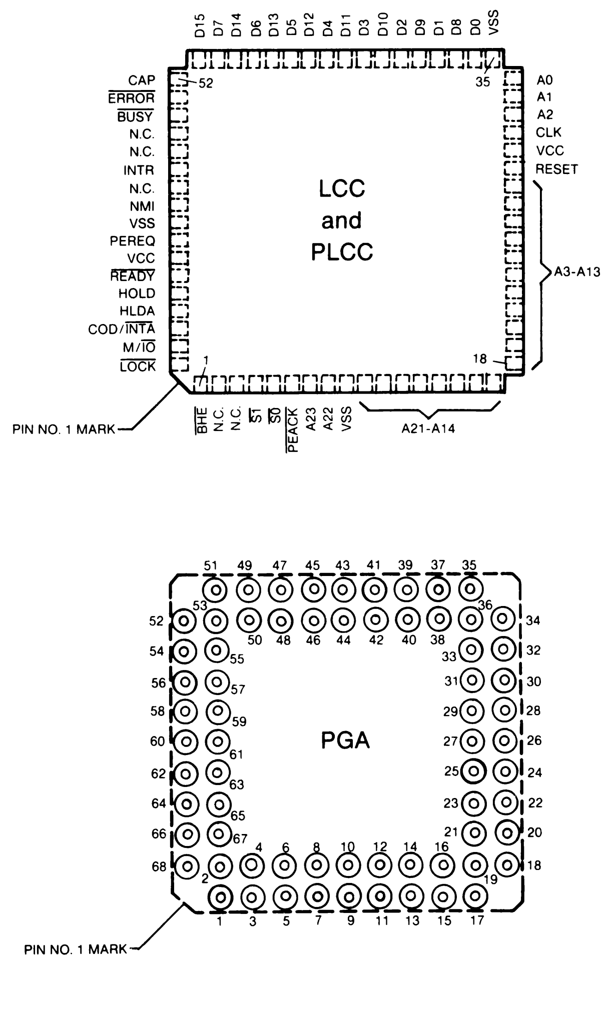


Figure 4-2. 80286H Pod and Microprocessor Pin Assignments (Top View)

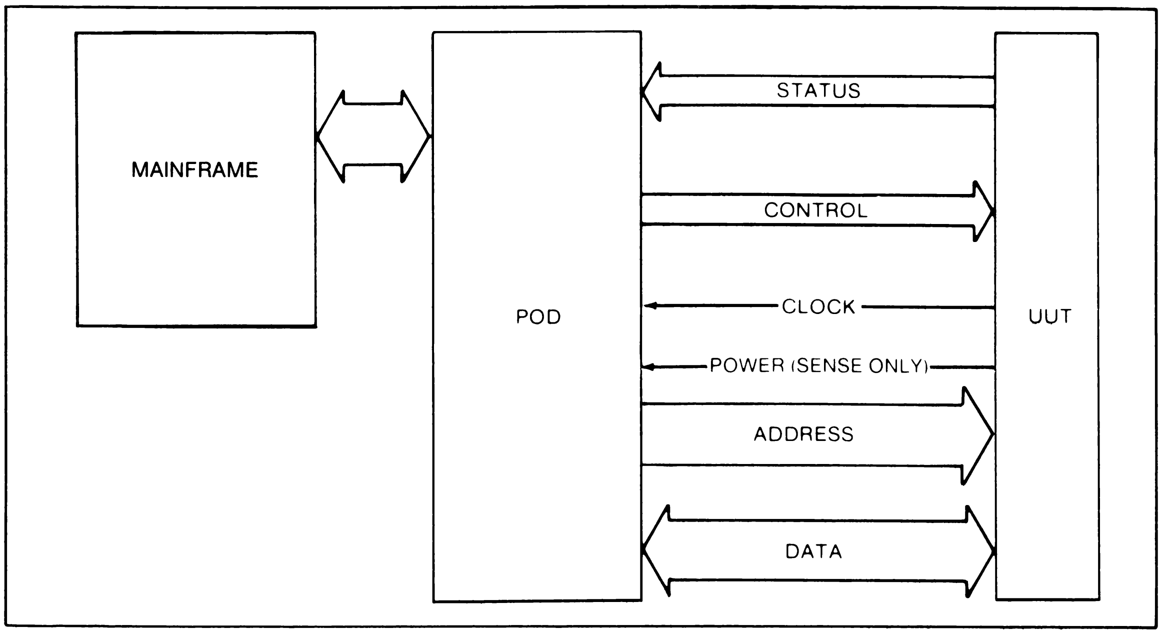


Figure 4-3. Signals Between the Pod and the UUT

FORCING LINES 4-19.

Forcing lines are a special category of status lines that, when asserted, could force the UUT's microprocessor into some specific state or action. When the Pod is plugged into the UUT's microprocessor socket in place of the microprocessor, the Mainframe reports activity on these lines with an active Forcing line error message.

The forcing lines of the 80286H Pod consist of CAP FAIL, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, PEREQ, RESET, HOLD, and $\overline{\text{READY}}$.

The enableable lines $\overline{\text{READY}}$ and HOLD are reported only as active Forcing lines when they are disabled from affecting Pod operation.

An active forcing line is reported as a high bit in a status error word, regardless of its actual logic level of the signal. For example, if the $\overline{\text{ERROR}}$ line (bit 7) is active (low), an active $\overline{\text{ERROR}}$ signal is reported as an active Forcing line with the status bits set to 80 (0000 0000 1000 0000 binary), while a read status operation would show the true logic levels by reporting 40 (0000 0000 0100 0000 binary). An NMI interrupt, on the other hand, (if the Mainframe's active interrupt trap is enabled), is reported as an active interrupt with the status bits set to 4 (0000 0000 0000 0100 binary), and a read status operation reports C4 (0000 0000 1100 0100 binary).

User-Enableable Forcing Lines 4-20.

The 80286 has two status lines that are individually enabled or disabled using the Mainframe's Setup function: $\overline{\text{READY}}$ and HOLD. If these user-enableable lines are disabled (using the Mainframe Setup function), their inputs to the Pod microprocessor are disabled, but the Pod continues to monitor their condition. If the user-enableable lines are asserted, the Pod reports to the Mainframe that a forcing line is active.

Table 4-3. Status Lines

BIT	STATUS LINE	SPECIAL CHARACTERISTICS
0	$\overline{\text{READY}}$	e, f
1	HOLD	e, f
2	NMI	i
3	INTR	i
4	RESET	f
5	PEREQ	f
6	$\overline{\text{BUSY}}$	f
7	$\overline{\text{ERROR}}$	f
8	CAP FAIL	p, f, m*
9	INT VECT	p
10	PWR FAIL 30	p, m
11	PWR FAIL 62	p, m
12	GND FAIL 35	p, m
13	GND FAIL 9	p, m
14	(not used)	
15	(not used)	

Special Characteristics Definitions

e -	an enableable line
f -	a forcing line
i -	an interrupt
m -	miscellaneous line
p -	a pseudo-status line. Does not reflect actual physical line condition of a line, but indicates a condition of the Pod that is important to the user.

*CAP FAIL is a forcing line for the 9000-Series Mainframe and a miscellaneous line for the 9100-Series Mainframe.

The active forcing condition for $\overline{\text{READY}}$ is a high logic level ("not" ready). The active forcing condition for HOLD is a high logic level.

When these UUT-generated lines are disabled, they are prevented from affecting Mainframe and Pod operation. For example, a HOLD line that is stuck high would cause the 80286 within the Pod to remain in a HOLD state, preventing normal Mainframe/Pod operation. When the Setup function of the Mainframe is used to disable this input to the Pod, the HOLD signal is prevented from reaching the 80286 within the Pod, allowing the Mainframe/Pod interactions to take place normally.

If the user-enableable lines are enabled, they are not reported as forcing line errors, even when they are asserted.

NOTE

Operating the Pod with the enable lines enabled may degrade Pod execution speed, especially if the UUT requires Wait states or HOLD/HLDA operations to operate correctly.

Disabled Forcing Lines 4-21.

The $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, PEREQ, and RESET lines are forcing lines that are hardware-disabled except during RUN UUT. The lines are monitored and reported as active forcing lines if they are active during non-RUN UUT operation.

INTERRUPT LINES 4-22.

Some of the status lines are interrupt signals. Interrupt lines of the 80286H Pod consist of INTR and NMI. The NMI input is disabled by hardware except during operation in the RUN UUT mode. Although disabled, the NMI input is routinely checked by the Pod software and reported to the Mainframe if held high by the UUT. NMI is also reported as a status bit (bit 2).

NOTE

Reporting of the active interrupt error message is disabled at power on. It is possible to enable the reporting of active interrupt lines by selecting the Mainframe Setup active interrupt trap. Sometimes it is useful to enable the reporting of active interrupts if you need the information.

PSEUDO-STATUS LINES 4-23.

Pseudo-status lines are Pod-generated read-only information bits that convey information about Pod or UUT operation that is not otherwise available from the microprocessor- or UUT-generated signals. The Pod creates the information solely for display on the Mainframe. The Mainframe displays pseudo-status lines along with microprocessor/UUT-generated status lines when information is requested (for example, by reading the status). Pseudo-status lines provide you with added information about the results of Pod functions. For example, PWR FAIL 30 is a Pod-generated status line that shows that power at pin 30 of the UUT's microprocessor socket is out of the range of 4.5 to 5.5 volts; information that is not obtainable using only the microprocessor's lines.

The following paragraphs describe pseudo-status lines. Bit assignments for the pseudo-status lines are shown in Table 4-3.

Power Fail (PWR FAIL 30 and PWR FAIL 62) 4-24.

The PWR FAIL 30 and PWR FAIL 62 status lines are set high by the Pod whenever UUT power supply voltage on either of the 80286's Vcc pins drops below 4.5 volts or rises above 5.5 volts. The Read Status function distinguishes between PWR FAIL 30 and PWR FAIL 62 in the status bits returned to the Mainframe.

Ground Fail (GND FAIL 9 and GND FAIL 35) 4-25.

The Pod checks for continuity on the 80286 Vss pins by using one pin (pin 60) as a reference and having the others (pins 9 and 35) connected to latches with a pull-up resistor on each. If a UUT has an open ground connection to the processor, the latch input goes high, and a powerfail condition is reported. A Read Status differentiates between GND FAIL 9 and GND FAIL 35 in the status bits returned to the Mainframe.

NOTE

An open circuit on pin 60 of the UUT socket has unpredictable effects. Pin 60 of the 80286 is used for ground reference from the UUT to the Pod, and loss of the reference may cause a power-fail condition to be reported by the Pod.

Interrupt Vector (INT VECT) 4-26.

The INT VECT pseudo-status bit reports that the Pod has performed an interrupt-acknowledge sequence in response to an interrupt. The INT VECT bit also indicates that the Pod has stored interrupt-type information (and, possibly, a cascade address) that was generated by an external Programmable Interrupt Controller. A Read Status shows if an interrupt vector has been read from the data bus.

INT VECT is reset by using appropriate Pod Function addresses (see the heading "Testing Interrupt Circuitry" in Sections 2 and 3 for more information).

Capacitor Fail (CAP FAIL) 4-27.

The CAP FAIL status line is set high by the Pod whenever the test for the value of the external capacitor fails. (The Pod has an internal capacitor for its own microprocessor and does not depend on the UUT capacitor for operation.) The tolerance for the external capacitor is +100%, -50%. A Read Status shows if the capacitor is out of tolerance.

Control Lines 4-28.

Control lines are Pod output lines that provide information to the UUT. The control lines and some of their characteristics are listed in Table 4-4.

Table 4-4. Control Lines

BIT	CONTROL LINE	SPECIAL CHARACTERISTICS
0	$\overline{LOCK}$	w
1	HLDA	w
2	$\overline{PEACK}$	w
3	$\overline{BHE}$	
4	$\overline{S1}$	
5	$\overline{S0}$	
6	M/$\overline{IO}$	
7	COD/$\overline{INTA}$	
8	(not used)	
9	(not used)	
10	(not used)	
11	(not used)	
12	(not used)	
13	(not used)	
14	(not used)	
15	(not used)	

w - writable control line

Information about control lines is displayed as a map of bits, with each bit corresponding to a control line. The bits assigned to the specific control lines are shown in Table 4-4 and on the decal on the back of the Pod. For example, when performing Bus Test or various other Mainframe functions, the Mainframe may detect that one or more control lines are not drivable. Suppose that during a BUS TEST the Mainframe detects that the $\overline{\text{LOCK}}$ line is not drivable. The Mainframe displays a control line error message. The zeros and ones in the message correspond to the bit numbers assigned to the control lines as listed in Table 4-4. The first bit, Bit 0, is set to 1 because the $\overline{\text{LOCK}}$ line was detected as not drivable.

The 80286H Pod has three control lines that the Mainframe can pulse high or low with the Write Control function: $\overline{\text{LOCK}}$, $\overline{\text{PEACK}}$, and HLDA. This feature is used by BUS TEST to check lines that cannot be toggled by normal read and write operations. Writing to control lines also helps troubleshoot the control lines.

The Write Control and Data Toggle Control require the entry of binary digits to define user-writable control lines. When using either of these two functions, the Mainframe display prompts the operator for a number to identify the control line(s) to be written. For the 9100-Series Mainframe, the number is hex. For the 9000-Series Mainframe, the number is binary.

NOTE

The Write Control function sets a line high or low for only one UUT access, just long enough to verify that the line can be driven. Following the UUT access, the lines return to their inactive state.

SPECIAL SIGNAL STATES 4-29.

Special signal states are caused when the HOLD input is asserted (active high) or by abnormal UUT power supply levels.

The following lines are set to high-impedance states when the HOLD input is high:

A0 through A23
D0 through D15
$\overline{\text{BHE}}$
$\overline{\text{S0}}$
$\overline{\text{S1}}$
M/$\overline{\text{IO}}$
COD/$\overline{\text{INTA}}$
$\overline{\text{PEACK}}$
$\overline{\text{LOCK}}$

If the Pod detects a voltage below 3.5 volts on 80286 Vcc pins 30 or 62, the Pod drives selected signal lines to a high-impedance state. This protects UUT circuitry from damage by signals that might be present when the UUT cannot operate normally. The Mainframe then indicates UUT power failure. The Mainframe tries to reset the Pod until the UUT supply lines rise above 3.5 volts.

All of the lines listed above and HLDA are set to a high-impedance state when the voltage drops below 3.5 volts.

POD DRIVE CAPABILITY 4-30.

As a driving source on the UUT bus, the Pod provides equal to or greater than normal 80286 current drive capability. All Pod inputs (except CLK) and all outputs are TTL-compatible.

PROBLEMS DUE TO A MARGINAL UUT 4-31.

The Pod is designed to approximate, as closely as possible, the actual characteristics of the microprocessor 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 microprocessor 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.

UUT Operating Speed and Memory Access 4-32.

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 become apparent.

Many high-speed 80286 UUTs sample the $\overline{S0}$ and $\overline{S1}$ signals immediately after the signals are generated, with very little timing margin. Because of the Pod's slight time delay on $\overline{S0}$ and $\overline{S1}$, marginal timing on these signals can be more readily detected.

UUT Noise Levels 4-33.

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 the Mainframe.

Bus Loading 4-34.

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

For the 80286H Pod, the open-collector $\overline{\text{READY}}$ signal is pulled to a logic high more slowly because of this added capacitance, making any initial $\overline{\text{READY}}$ timing or drive problems more apparent.

Clock Loading 4-35.

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.

Section 5
Theory of Operation

INTRODUCTION 5-1.

The theory of operation of the Pod is described on two levels in this section. The first level is a general description of the major sections of the Pod and how they relate to each other, to the UUT, and to the Mainframe. The second level is a detailed description of each Pod section. The descriptions are supported by block diagrams in this section and by complete instrument schematics in Section 8 of this manual.

GENERAL POD OPERATION 5-2.

The Pod is essentially a complete microprocessor system by itself, with the capability of switching between the buses and signals of the UUT and those internal to the Pod. It is usually in an internal "housekeeping" mode, waiting for instructions from the Mainframe. Under these conditions, it functions like any normal microprocessor-controlled system. When the Pod receives an instruction, it switches buses and performs an operation or series of operations on the UUT microprocessor bus. When the Pod accesses the UUT this way, the bus is momentarily (for the duration of a memory access cycle or an I/O cycle) switched to the UUT by disabling the components in the Pod and connecting all lines to the UUT, buffered in the appropriate direction.

In the RUN UUT mode, the components within the Pod are permanently disabled, and the Pod microprocessor is effectively permanently connected to the UUT.

For the purposes of description, the Pod may be divided into four major functional areas:

- Processor Section
- UUT Interface Section
- Timing and Control Section
- UUT Power Sensing Section

The general operation of each section is described in the following paragraphs.

Processor Section 5-3.

The Processor Section, shown in Figure 5-1, is made up of a microprocessor, RAM, ROM, an I/O interface to the Mainframe, and various latches and buffers. These elements, along with timing components, constitute a small microsystem that receives Mainframe commands and directs all Pod operations during execution.

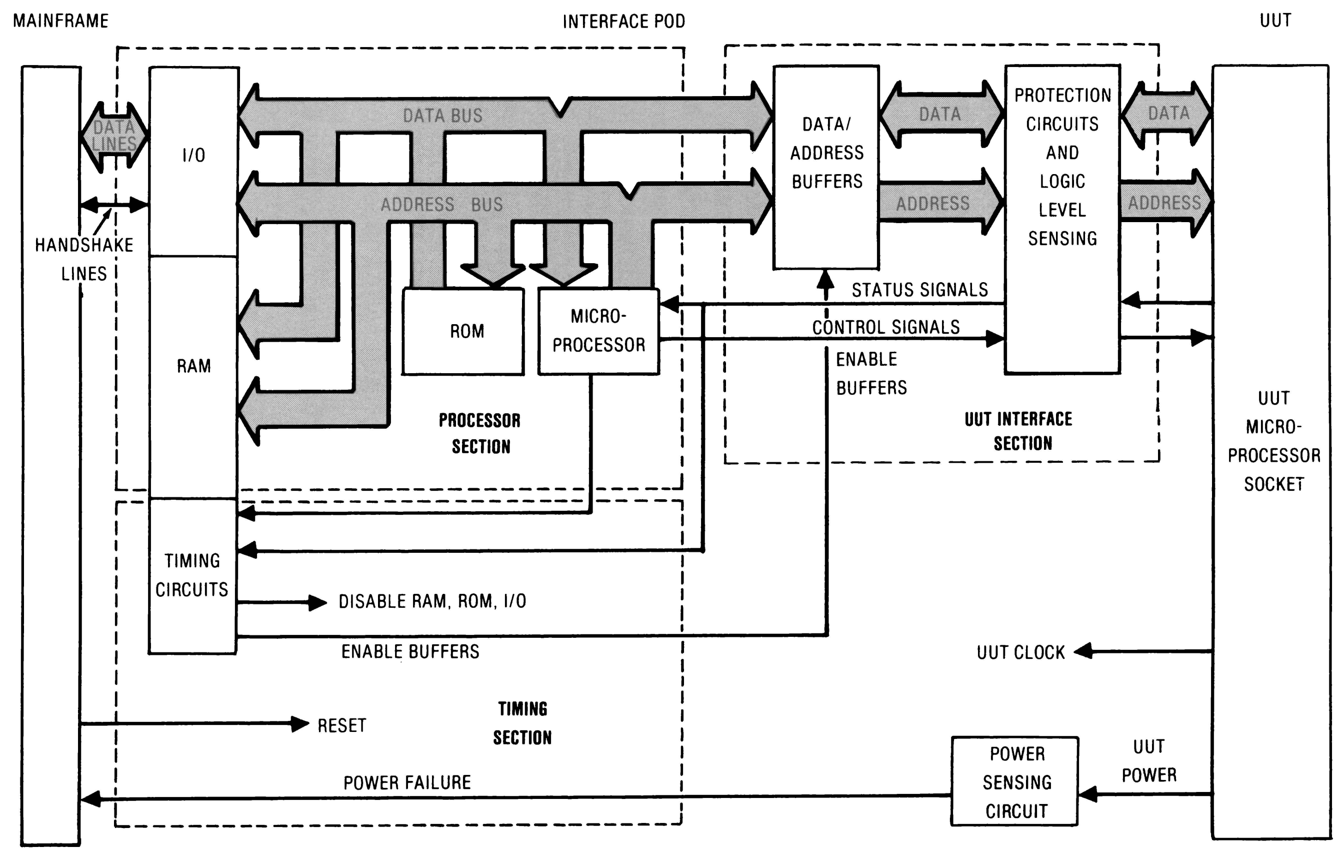


Figure 5-1. 80286H Interface Pod General Pod Block Diagram

For the 80286H Pod, as well as all other Pods, the microprocessor inputs from the UUT are referred to as Status Lines, and the outputs to the UUT are referred to as Control Lines. This nomenclature is not always in agreement with the manufacturer's literature (80286 manufacturers, for example, refer to line $\overline{S1}$, which is a microprocessor output, as a status line). This convention, however, allows consistency between Pods when implementing the Mainframe functions that involve status or control lines, such as read status or write control. Refer to the definition of status and control lines in Section 4 for more information about these signals.

All Pod status lines that could adversely affect the Pod operation are either automatically disabled by the Pod or may be disabled by the operator using the Mainframe's Setup function. Disabling these status lines allows the Pod to operate in UUT environments in which malfunctioning status lines such as HOLD could prevent the Pod from performing any tests. The one microprocessor input that may not be disabled, of course, is the UUT clock. The clock signal must always be present for Pod operation. All the status lines are enabled in the RUN UUT mode.

The Processor Section also contains circuitry for Pod self test. When the UUT cable plug is inserted into the self test socket, part of the Pod circuitry becomes a simplified pseudo-UUT. During Pod self test, certain tests are performed on this pseudo-UUT, and any failures are reported to the Mainframe.

UUT Interface Section 5-4.

The Interface Section, shown in Figure 5-1, consists of buffers and drivers, protection circuits, logic level detection circuits, and a clock buffer. The buffers and drivers switch the UUT to the microprocessor or to the standby control and address signals, as dictated by the Timing and Control Section.

Each UUT signal is protected from overvoltage or short-circuit conditions that might damage Pod components. Resistors in series with the inputs of the detection circuit latches limit the input current, and resistors in series with the output drive lines limit output current. A pair of clipping diodes connected to ground and power protect the Pod's circuitry against damaging voltages.

The detection circuits consist of latches connected through the protection circuitry to the UUT. The latches are clocked during a UUTON cycle, when the signals are expected to be at a known level.

If a signal cannot be driven through the output-current-limiting resistor, an error is detected when the latches are individually read by the Processor Section, and the values are compared with the expected values.

The clock buffer module at the end of the plug cable takes the clock signal used on the UUT and amplifies it to a suitable level to overcome the load of the cable and Pod components.

Timing and Control Section 5-5.

The Timing and Control Section, shown in Figure 5-1, consists of a state machine and internal timing and control logic. The Timing and Control Section receives inputs from the Processor Section and the Interface Section. When a UUT access occurs, the internal data bus is disabled and the buffers to the UUT are enabled. At other times, the UUT buffers are disabled and the internal data bus is enabled. This bus switch is accomplished by buffer control signals and chip enable signals generated by the

Timing and Control Section in response to inputs from the state machine, the control register outputs as set by the microprocessor, the status lines from the UUT, and the control lines from the microprocessor.

A single bus switch lasts one microprocessor memory access cycle or I/O cycle. The bus is switched to communicate with the UUT between the last internal operation and the start of the intended UUT operation. The bus is switched back to communicate with the Mainframe at the end of the cycle.

If the microprocessor has sent the RUN UUT command to the Pod's control register (or port), the bus switch is started in the normal fashion, but is then held on indefinitely until RUN UUT is terminated by the Mainframe.

During the time that the Pod is not communicating with the UUT, the UUT needs the proper signals so that it can perform normal dynamic memory refresh operations and other similar tasks. In order to provide these signals to the UUT, the Pod performs what are called transparent (or standby) reads. A transparent read is a read operation that is performed at a selected address. Transparent reads generate the transparent or fake control signals required to simulate a normal microprocessor read operation. This allows the UUT to maintain non-CPU operations, even when the Pod's microprocessor is not communicating with it.

UUT Power Sensing Section 5-6.

The UUT power-sensing circuit shown in Figure 5-1 constantly monitors the UUT power supply. This circuit produces an output signal to the Mainframe in the event that UUT power drops below 4.5V or rises above 5.5V.

Whenever the UUT power supply drops below about 3.4V, all active Pod outputs are changed to their high-impedance states. This feature protects UUT circuits from being damaged by Pod outputs when the UUT power supply drops below safe operating limits. The Mainframe also displays a UUT power-fail error message. When the proper operating power supplies have been restored to the UUT, the outputs of the Pod return to normal, and the Mainframe is ready for additional testing.

DETAILED THEORY OF OPERATION 5-7.

A detailed block diagram of each major Pod section is presented in Figure 5-2. Each major section is described in the following paragraphs.

Processor 5-8.

The Pod contains a complete 80286 kernel. The kernel consists of a 80286 microprocessor, RAM, ROM, and I/O. The kernel is responsible for establishing communication with the mainframe, decoding the commands of the mainframe, and performing the operations commanded.

Besides executing code to carry out the functions of the kernel, the microprocessor also generates part of the signal timing for the UUT interface, and serves as the processor for the RUN UUT function.

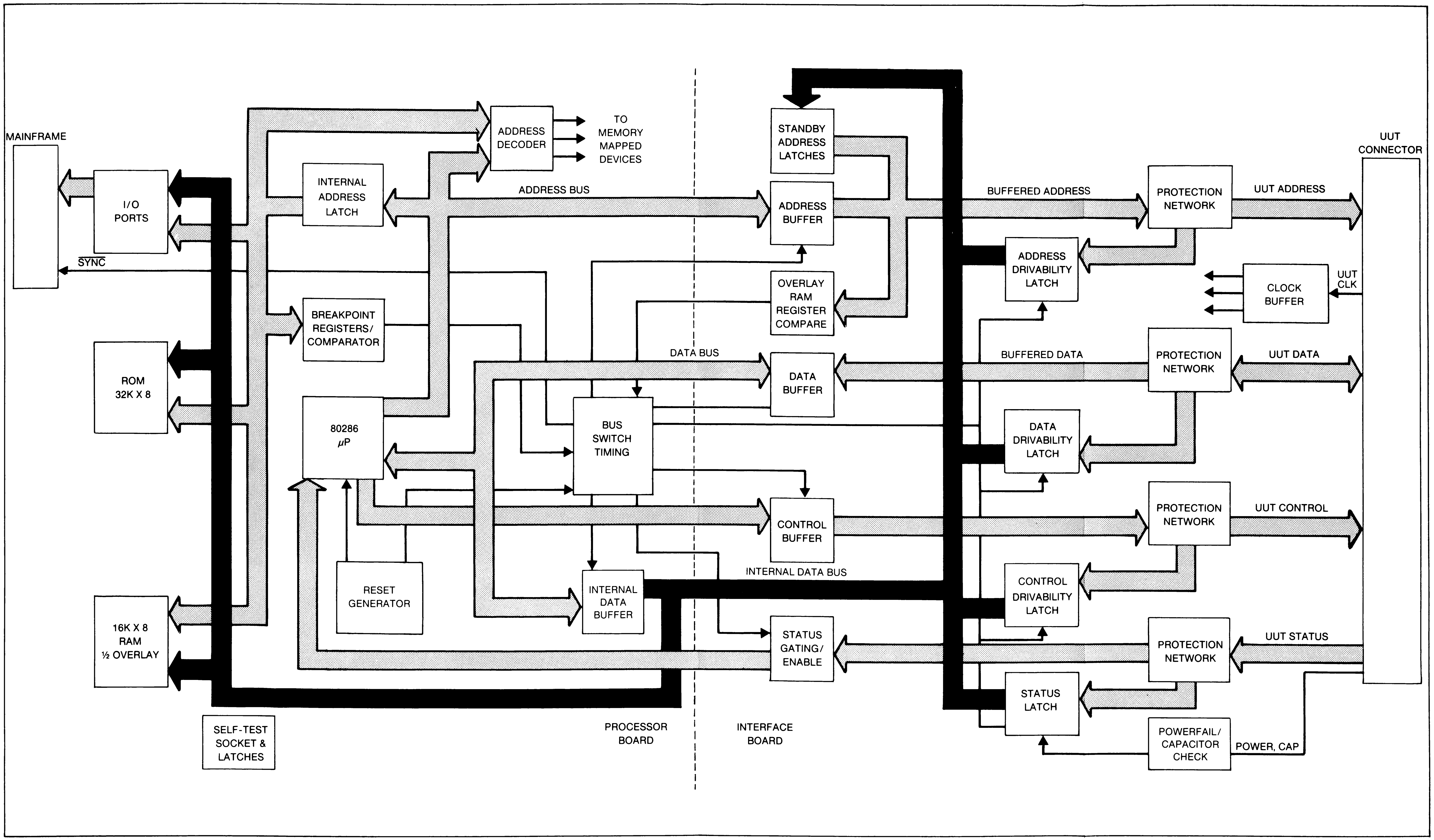


Figure 5-2. 80286H Interface Pod Block Diagram

INTERNAL BUS CONTROL 5-9.

The 80286's address and data buses are latched and buffered to improve the fanout and timing beyond what the microprocessor provides. The Internal Bus Control circuit block shown in Figure 5-2 controls the latch and buffer timing and also provides a $R/\overline{W}$ signal to the RAMs. The internal bus controller takes microprocessor signals $\overline{S0}$, $\overline{S1}$, and CLK as inputs and forms signals ALE (address latch enable), $R/\overline{W}$ (read, not write), and $DT/\overline{R}$ (data transmit, not receive) as outputs.

The internal bus controller consists of gates from U56, U61, U55, and flip flop U13A. The gates are configured as an asynchronous state machine for maximum speed. Gate U51B forms the primary output $R/\overline{W}$, while U55A forms its complement $W/\overline{R}$ in parallel, faster than a serially connected inverter. Flip flop U13A delays $W/\overline{R}$ by one clock cycle to define the data period and the data buffer control signal $DT/\overline{R}$. ALE is formed by ORing $\overline{S0}$ and $\overline{S1}$.

ADDRESS LATCHES AND DATA BUFFERS 5-10.

The internal address latches are the three octal transparent latches U58, U54, and U49. These latches are necessary because the 80286 provides valid addresses for only a short period during the beginning of the bus cycle. However, most devices require addresses for the full cycle. The address latches capture the address at the beginning and hold it valid for the full bus cycle.

The internal data bus buffer consists of bidirectional tri-state buffers U57 and U52. Their direction is controlled by the $DT/\overline{R}$ signal. When $DT/\overline{R}$ is low (as in a memory read operation), the buffers take input from the internal data bus and output to the microprocessor. When $DT/\overline{R}$ is high (as in a memory write operation), the buffers take input from the microprocessor and output to the internal data bus. The buffers are tri-stated during UUT accesses and RUN UUT, but not during UUT accesses that involve Overlay RAM. The signal DATA-BUF-EN from the interface board controls the tri-stating of these buffers during UUTON.

ADDRESS DECODER 5-11.

The kernel address decoding is performed by decoders U19, U39, and U51, latch U27, and some miscellaneous gates. The address decoder takes the 80286's address and data as inputs and produces chip selects and strobes as outputs.

The address decoder has an architecture that serves to increase the decoding speed for RAM and ROM. The 80286 chip produces the address for the current bus cycle well ahead of the ALE pulse that marks the beginning of the cycle. The Pod takes advantage of this by placing the decoder ahead of a latch (U27) of its own, so the propagation delay of the decoder has elapsed before ALE opens the decoder's latch. In this way, the delay due to the RAM and ROM decoder is effectively hidden. This latch also latches A0 and $\overline{BHE}$ to provide signals that select single-byte transfers. The remaining decoding for the rest of the chip selects is not as speed-critical and is more conventionally driven by the latched address bus.

The address decoder structure defines the memory map. See the heading "Memory Map" in Section 6 for more information.

MEMORY 5-12.

The Pod contains 8k 16-bit words of RAM in U40 and U41, and 16k 16-bit words of ROM in U48 and U37.

The lower half of the RAM space (4k 16-bit words) is dedicated to Pod software use. The upper half is dedicated to overlay RAM. RAM chip input A12 controls which half is accessed.

INPUT AND OUTPUT PORTS 5-13.

Mainframe Data Interface 5-14.

The Mainframe data interface consists of buffer U5 and register U4. The buffer allows the kernel to sample the byte sent by the Mainframe, and the register allows the kernel to send bytes to the Mainframe. When the Pod is receiving, the output drivers of the register are turned off via an output bit from register U36.

Miscellaneous I/O 5-15.

The Pod has a variety of internal functions to sample and control. Registers U31, U36, and U28 provide outputs, and buffer U33 provides inputs to handle these miscellaneous I/O functions.

UUT Interface 5-16.

The UUT interface circuitry consists of buffers to and from the UUT on all microprocessor lines, drivability latches to check the logic level on each output during a UUT access, and protection of the circuitry from faults on the UUT. In addition there are circuits to buffer the clock input, powerfail detection circuits, a capacitor check circuit, and the registers and comparators for overlay RAM.

ADDRESS AND DATA BUFFERS 5-17.

The Data bus is buffered by the bidirectional, tri-state buffers U25 and U30. The direction of the lines though these buffers is controlled by DT/R. These buffers are enabled only during a UUT access or in RUN UUT. The Address bus buffers are the output-only buffers U16, U20, and U28. The upper four bits of an address are treated differently. The latch U36 provides the high nibble of address for UUT accesses, and half of U28 is enabled only for RUN UUT. During standby, (when the Pod is not accessing the UUT), the latches U17, U22, and U26 output an address whether or not standby reads are enabled. CMOS PROM U1 controls all these buffers. U1 tri-states all the buffers if a hold-acknowledge cycle or a low power condition occurs.

DRIVABILITY LATCHES AND PROTECTION 5-18.

A1 through A4 are Fluke-designed hybrid circuits that protect the Pod circuitry from over-voltage conditions and contain source and sense resistors to detect output drivability faults. All output buffers are connected with a clamping diode to +5V. Any positive overvoltage fault turns on this diode. Similarly, a diode to ground protects Pod circuitry from negative voltages. Drivability faults are detected by latching the output level when a valid level should be present (for the address, at the end of Ts; for data, at the end of Tc). An 80 ohm resistor in series with the output both limits the current through the diode and produces a voltage drop for the drivability check. Software compares the output value to the value latched during the access. Any time these values do not match, a fault is reported. A 500 ohm resistor protects the CMOS inputs on the drivability latches from damage.

CONTROL LINE BUFFERS 5-19.

U6, U7, U14, and U40 are individually controlled tri-state buffers. For each control line there are two buffers. In general, one buffer is used during UUT accesses and the other is turned on in standby. Both are tri-stated during a hold acknowledge or a low power condition. CMOS PROMS U2 and U3 control the output enables on the buffers. The Schottky diodes CR2 and CR3, along with R39, are a fast OR gate that generates the standby strobe $\overline{\mathrm{ST}}$. CR4 is a diode that pulls $\overline{\mathrm{ST}}$ high to disable standby reads when the two buffers are tri-stated. Additional control logic associated with U4 disables standby reads during a UUT-active reset to prevent problems in UUTs with an 82288 Bus Controller IC.

COD/$\overline{\mathrm{INTA}}$ is not enabled directly from the processor to the UUT when a UUT access occurs. The state of COD/$\overline{\mathrm{INTA}}$ is simulated by an output port of the Pod that sets the level of the line during the access. The line is ORed with $\overline{\mathrm{ADDR\text{-}BUF\text{-}EN}}$ so the level during a standby read is always high.

Writable control lines are handled in the following way. An output port is set to the desired level by software, and the correct timing signals are encoded to create the signal sent to the UUT. The writable control lines are inactive during standby and the actual processor signals are used only during RUN UUT.

The HLDA signal is a special case. Since a hold cycle can occur in standby, the actual signal is sent to the UUT. In addition, HLDA is the only signal not tri-stated in hold acknowledge.

STATUS GATING/ENABLE 5-20.

The status inputs to the microprocessor are disabled except during RUN UUT. The enableable lines HOLD and $\overline{\mathrm{READY}}$ are exceptions. These two lines can be enabled to the processor under software control via the port outputs ENREADY and ENHOLD. INTR is enabled in a similar manner for the purpose of performing an interrupt acknowledge cycle.

The most complex input is $\overline{\mathrm{READY}}$. $\overline{\mathrm{READY}}$ is disabled when UUT reset occurs (during standby reads, for example) and when overlay RAM is enabled. The source of $\overline{\mathrm{READY}}$ can be the UUT, or the Pod inserts zero wait states if $\overline{\mathrm{READY}}$ is disabled. The signal PODREADY ensures that $\overline{\mathrm{READY}}$ to the processor does not go low during Ts.

Other status lines used by the Pod include: BUSY, which controls part of the UUTON timer; NMI, which occurs when LOWPOWER is active; and RESET, which is either a mainframe or power-up LONG-RESET, or a self-generated SHORT-RESET. During a UUT access, U39 latches the state of each status bit, allowing active forcing lines to be reported or a read status operation to be performed.

CLOCK BUFFERS 5-21.

The clock from the UUT is conditioned in the UUT cable to prevent degradation and reflection problems. A high speed comparator converts the clock signal from TTL- to-ECL-level signals. The input is attenuated 2 to 1 because of the input range of the comparator. The switching level is set by the resistor divider R4 and R5; the effective switch point is 2.1V. In the Pod, U42 converts the signal back to TTL level. Several gates in IC U42 supply opposite phases of the clock and limit fan-out.

CAPACITOR CHECK CIRCUIT 5-22.

Transistor Q1 is connected through a 681 ohm resistor to +12V. A 32 Khz square wave is applied to the base of Q1; the capacitor is charged through the 681 ohm resistor and discharged through the collector of Q1. The cap is within +100%, -50% of 0.047 μF if, at the rising edge of the clock, the voltage is within the window set by R24, R25, and R26 (3V at the junction of R24 and R25, and 9V at the junction of R25 and R26). A 51.1

ohm resistor and clamping diodes to +12V and ground provide protection from faults. The crystal Y1 and IC U46 make a simple oscillator. The Schmitt trigger U44 conditions the slow rise and fall times of the HCU04 for input to the ALS flip-flops (U41).

POWERFAIL DETECTION CIRCUIT 5-23.

The powerfail sensing circuit consists of a pair of window detectors to monitor the VCC lines from the UUT and a single comparator on each line to detect voltage below 3.5V (lowpower). The ground lines (35 and 9) are monitored by tying the line to a latch input (U27). If they are pulled high by R30 or R31, an open condition is reported. Pin 60 at the UUT is used as a reference for the Pod as a whole and for powerfail detection. An open on Pin 60 is detected as a powerfail on pins 62 and 30.

To ensure powerfail detection, the comparator references are set to approximately 5.3V and 4.6V. This reference allows for resistor tolerances, comparator offset voltages, and other environmental variations while still guaranteeing that 5V, ±5% is acceptable and that greater than ±10% is unacceptable. 50 mV of hysteresis on each comparator ensures that errors are not reported intermittently. A capacitor filter to each input of the comparator eliminates noise.

Low power detection is implemented as a Schmitt trigger centered on 3.5V with 0.2V hysteresis. The hysteresis prevents the comparator on this line from oscillating.

Timing and Control 5-24.

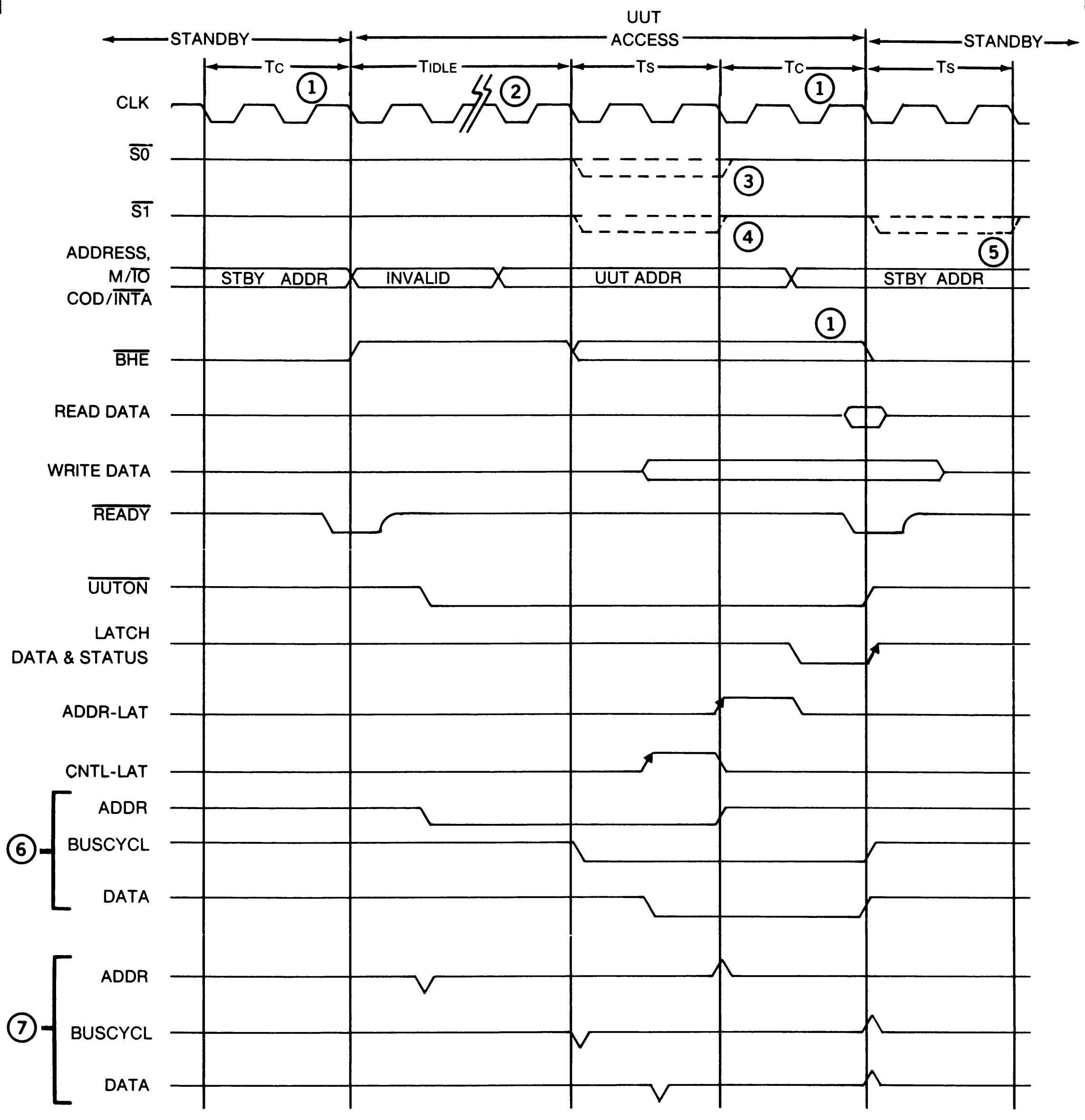
The components that compose the timing and control section are located on the processor board. The timing and control section is divided into functional subsections for the purposes of discussion, although some of the subsections are closely interrelated. Timing relationships of several important timing and control Pod signals are shown in Figures 5-3 and 5-4.

The function of the timing and control section is to switch the Pod buffers from internal to UUT context. The basic method is to put the Pod in a wait condition, and then a timer asserts and de-asserts the BUSY line to the processor to synchronize the bus switch with the execution of the opcode of interest. The programmable logic devices U34 and U29 are state machines tracking the processor bus cycle states. From the inputs of M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, and ALE, the Programmed Logic Array (PLA) outputs BUS-END, TC, and DATA-BUF-EN to control kernel functions.

BUS-END is used for the Output Enable on the ROM and RAM. TC along with UUTON-REQ enables the counter (U16) that asserts BUSY, to start the UUTON cycle.

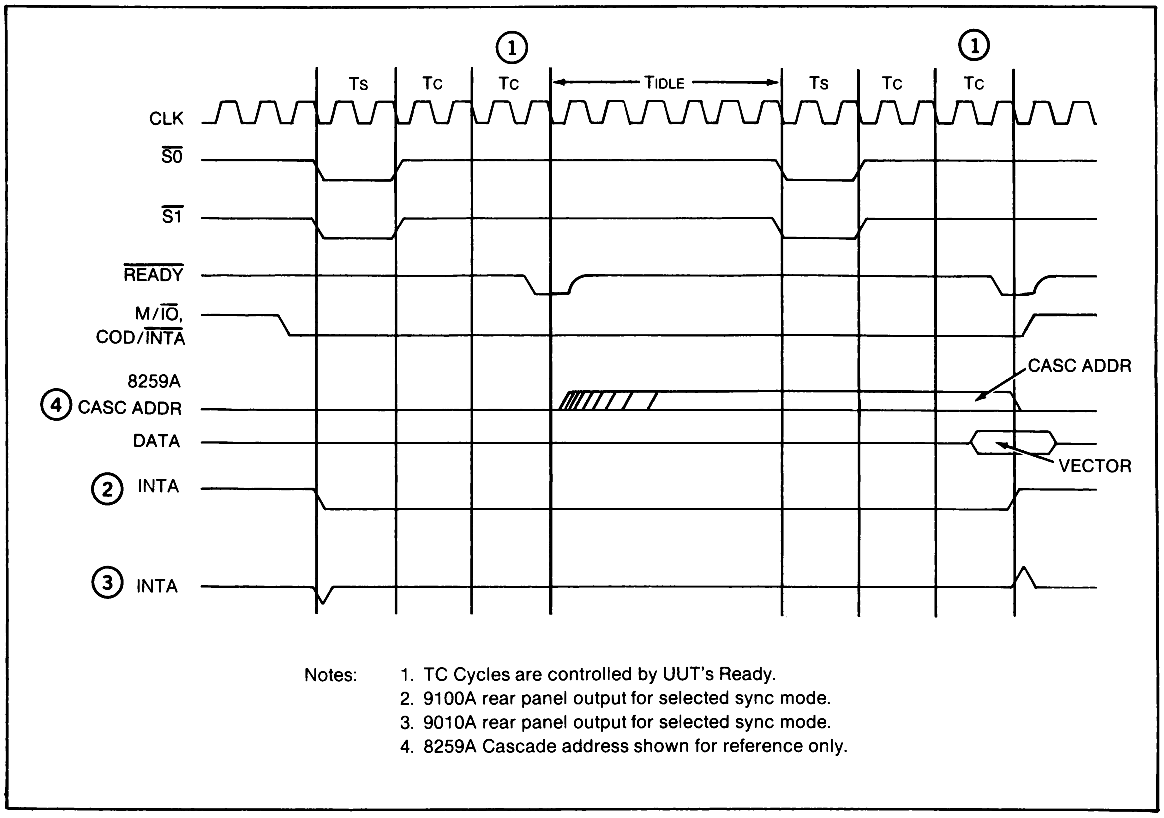
UUTON GENERATOR 5-25.

The UUTON cycle is started by a read at FA00. This is decoded as the UUTON-REQ signal. The UUTON-REQ signal sets the flip-flop U2A, which in turn outputs $\overline{\text{Q}}$ as PRE-UUTON. The Busy Counter then starts counting down from 15 to zero (4-bit binary counter). When the output BUSY-CTR-RCO is de-asserted, $\overline{\text{BUSY}}$ is asserted to the processor (via U19 on the interface board). After the Read at FA00 by the software, a WAIT instruction is executed that puts the processor in an inactive state until $\overline{\text{BUSY}}$ is de-asserted. When the 4-bit binary counter counts up to 15, BUSY-CTR-RCO goes high, as well as $\overline{\text{BUSY}}$. Note that the Busy Counter does not count during Tc states because the Enable input (connected to U14D) is controlled by TC. The PLAs U34 and 29 also monitor BUSY-CTR-RCO and begin switching the buses to UUT context; first the address bus (ADDR-BUF-EN), then control and data. The PLAs generate the drivability latch timing signals and the various sync mode timing signals (that are multiplexed via U22 to the Mainframe).



NOTES:
1. Only one Tc cycle is shown. Additional Tc cycles are controlled by UUT's READY signal. Standby address appears on the first Tc cycle. BHE is valid to end of first Tc cycle.
2. Exact length of TIDLE period will depend on type of UUT access performed.
3. S0 will go low for a write access.
4. S1 will go low for a read access.
5. S1 will go low if standby reads are enabled.
6. 9100A rear panel output for selected sync mode.
7. 9010A rear panel output for selected sync mode.

Figure 5-3. 80286 Pod Read and Write Cycle Timing



1
1
Ts
Tc
Tc
TIDLE
Ts
Tc
Tc
CLK
S0
S1
READY
M/IO,
COD/INTA
8259A
4 CASC ADDR
CASC ADDR
DATA
VECTOR
2 INTA
3 INTA
Notes:
1. TC Cycles are controlled by UUT's Ready.
2. 9100A rear panel output for selected sync mode.
3. 9010A rear panel output for selected sync mode.
4. 8259A Cascade address shown for reference only.

Figure 5-4. 80286 Pod Interrupt-Acknowledge Cycle Timing

RUN UUT is started the same way as UUTON, but the PLAs never switch the buffers back to Pod context. RUN UUT is exited by a reset of the processor and the timing and control circuit only (a short reset of 32 clock cycles). An abort signal from the Mainframe, a breakpoint reached, or a self-induced reset can cause this short reset. U15 counts for 32 clock cycles, then the Pod runs again internally from the reset address. To allow the software to interpret what caused the reset, a reset monitor composed of three latches (U46A, U46B, and U50) is read and the cause determined.

RESET CIRCUITS 5-26.

A power-up reset generator and power supply monitor chip (U3) assures that the proper reset pulse is applied at power-on and that the Pod is reset if there is any power supply interruption. A counter (U60) is clocked by CAPCLK (at approximately 32 Khz) and continuously cleared by the UUT clock. When the UUT clock is missing, the counter asserts the UUT-CLK-FAIL signal and a reset occurs. A long reset can be for any length of time, depending on the source. U2B and U44 flip-flops assure that a PODRESET from the mainframe remains synchronized with the UUT's PCLK (a clock half the frequency of the main clock).

BREAKPOINTS 5-27.

The Breakpoint circuit consists of a set of three octal latches, three octal comparators, and associated hardware. The basic function of the Breakpoint circuit is to stop RUN UUT on a set address. The break address is loaded into the registers by software, and when the UUT address matches that in the registers, the comparator's output goes true, causing a short reset. The enables of the comparators are controlled by M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$, so that a break only occurs on an instruction fetch. The latches U62A and U62B hold M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$ valid for the entire bus cycle.

SELF TEST 5-28.

The self-test socket provides access to a very simple UUT inside the Pod. The address lines are divided into two groups and looped back to the data bus via latches U24, U25, and U26. First the eight higher-order address bits are enabled by software using the port bit STOE1 while the lower 16 bits of address, controlled by $\overline{\text{STOE0}}$, are off. Then the lower sixteen address bits are connected to the data bus while the upper eight bits are off. $\overline{\text{STOE0}}$ is ORed with DT/$\overline{\text{R}}$ and $\overline{\text{S0}}$ so that a write operation does not find data drivability errors (for the 9000A Pod self test). The control and status bits are looped back directly by using different types of accesses (I/O, byte, etc.), the status bits are read in different states. $\overline{\text{S0}}$ and $\overline{\text{S1}}$ are special cases because of their timing; no status bit is latched during the time they are driven low, so they run directly to the interface board latch U24. A0 also goes directly to the interface board latch U24. Several ports control status inputs (the enableable lines HOLD and $\overline{\text{READY}}$ are controlled in this way).

The powerfail circuit is tested by setting the output port bits STGND0 and STGND1 high. Setting these bits high allows groundfail conditions to be detected on the interface board. In addition, two open-collector inverters, U7A and U7C, pull the power supply pins 30 and 62 low. An open collector is used so that a resistor divider from +12V can produce +5V, ±5%. To test the capfail circuit, an open-collector inverter forces a failure to be detected when STOE1 is low.

The Pod detects that it is in self test by means of a latch input on the interface board connected to pin 60 of the self-test socket. When the UUT cable is inserted into the self-test socket on the Pod, the latch input is pulled low.

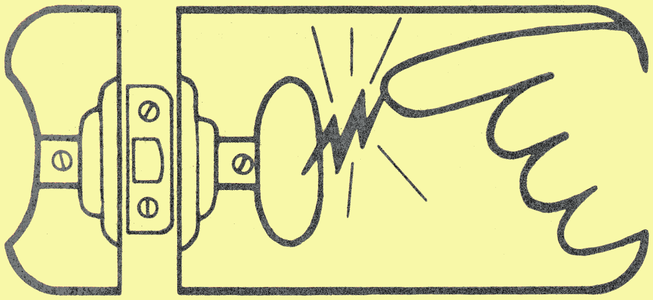
Several other circuits in the Pod are used for self-test. Latches U24 and U27 on the interface board have $\overline{\text{PODSYNC}}$ as an input. The Pod selects different sync modes and reads back from this latch if the correct levels are recorded. U24 has NMI and INTR as inputs because they are connected to M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$ in the self-test socket. These control lines are only valid through Ts so the timing must be control-latch, not data-latch timing.

On the processor board U55B ANDs ALE and STOE1 so the address drivability latches capture the standby address to test the standby address circuit. Flip-flop U44B allows the Pod to tell if the breakpoint circuitry is working without entering RUN UUT.

static awareness

A Message From

John Fluke Mfg. Co., Inc.



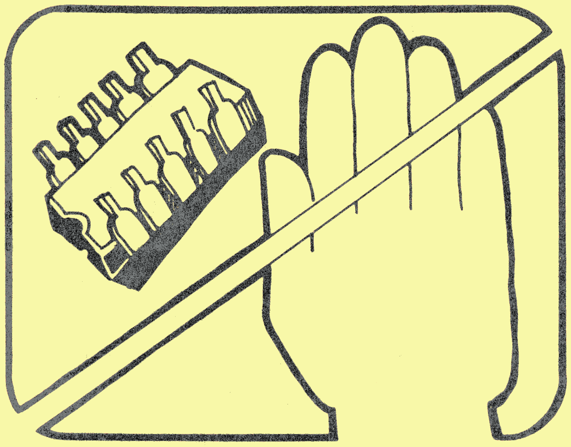
Some semiconductors and custom IC's can be damaged by electrostatic discharge during handling. This notice explains how you can minimize the chances of destroying such devices by:

1. Knowing that there is a problem.
2. Learning the guidelines for handling them.
3. Using the procedures, and packaging and bench techniques that are recommended.

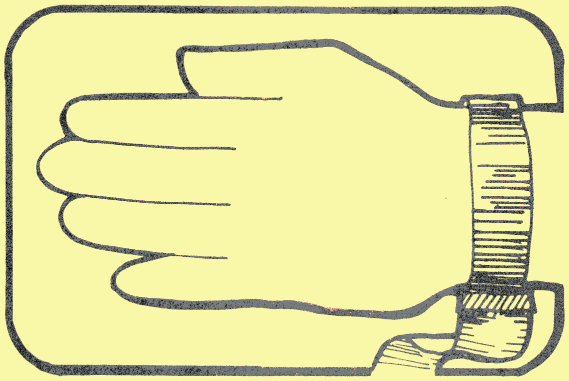
The Static Sensitive (S.S.) devices are identified in the Fluke technical manual parts list with the symbol

" ⊗ "

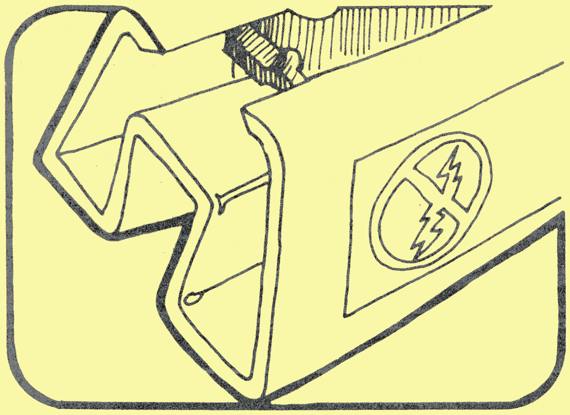
The following practices should be followed to minimize damage to S.S. devices.



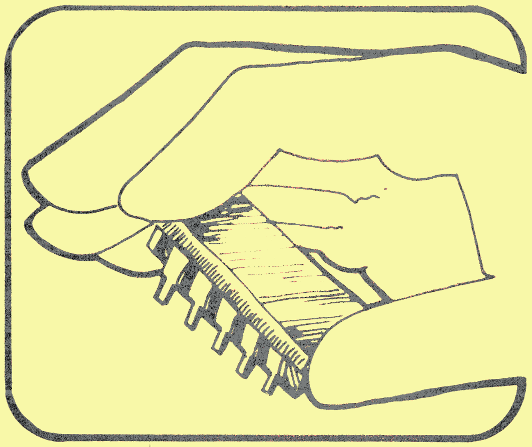
1. MINIMIZE HANDLING



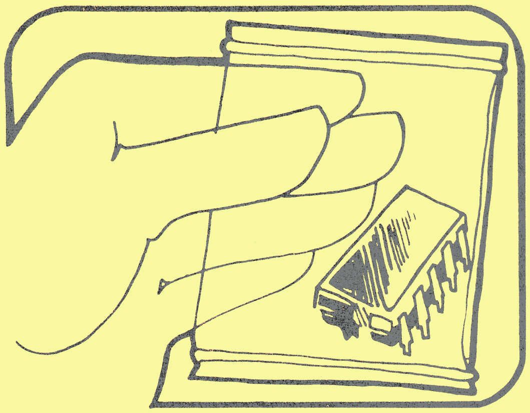
3. DISCHARGE PERSONAL STATIC BEFORE HANDLING DEVICES. USE A HIGH RESISTANCE GROUNDING WRIST STRAP.



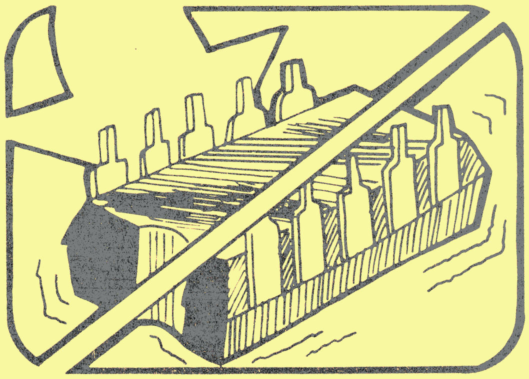
2. KEEP PARTS IN ORIGINAL CONTAINERS UNTIL READY FOR USE.



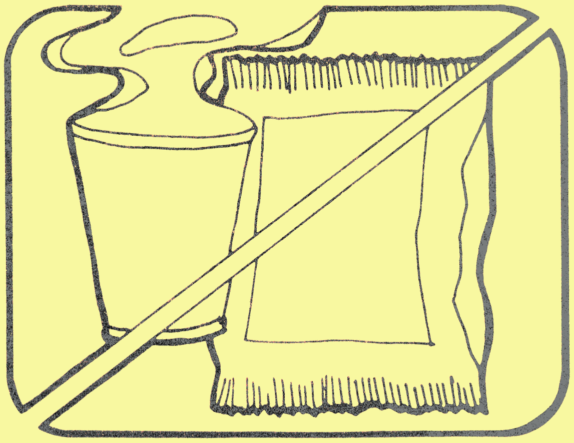
4. HANDLE S.S. DEVICES BY THE BODY



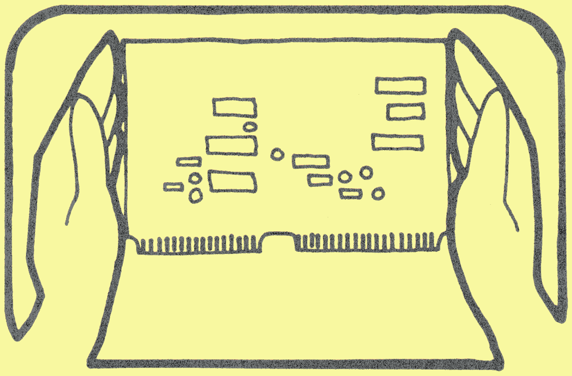
5. USE STATIC SHIELDING CONTAINERS FOR HANDLING AND TRANSPORT



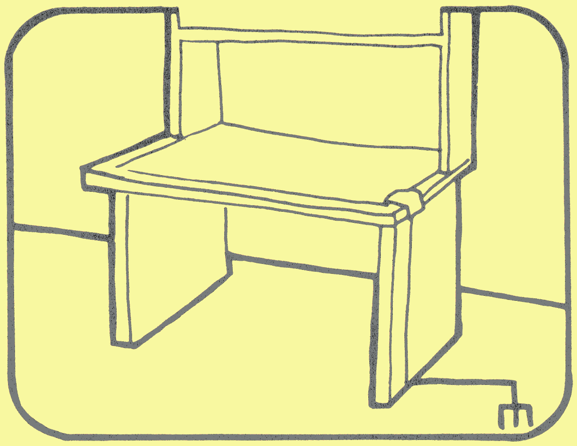
6. DO NOT SLIDE S.S. DEVICES OVER ANY SURFACE



7. AVOID PLASTIC, VINYL AND STYROFOAM® IN WORK AREA



8. WHEN REMOVING PLUG-IN ASSEMBLIES, HANDLE ONLY BY NON-CONDUCTIVE EDGES AND NEVER TOUCH OPEN EDGE CONNECTOR EXCEPT AT STATIC-FREE WORK STATION. PLACING SHORTING STRIPS ON EDGE CONNECTOR HELPS TO PROTECT INSTALLED SS DEVICES.



9. HANDLE S.S. DEVICES ONLY AT A STATIC-FREE WORK STATION

10. ONLY ANTI-STATIC TYPE SOLDER-SUCKERS SHOULD BE USED.

11. ONLY GROUNDED TIP SOLDERING IRONS SHOULD BE USED.

A complete line of static shielding bags and accessories is available from Fluke Parts Department, Telephone 800-526-4731 or write to:

JOHN FLUKE MFG. CO., INC.
PARTS DEPT. M/S 86
9028 EVERGREEN WAY
EVERETT, WA 98204

® Dow Chemical

J0089D-07U8604/SE EN Litho in U.S.A.
Rev. 1 MAR 86

Section 6
Troubleshooting

WARNING

THESE INSTRUCTIONS ARE FOR USE BY QUALIFIED PERSONNEL ONLY. TO AVOID ELECTRIC SHOCK, DO NOT PERFORM ANY INSTRUCTIONS UNLESS YOU ARE QUALIFIED TO DO SO.

INTRODUCTION 6-1.

This section provides troubleshooting information for the Pod and includes repair precautions and disassembly procedures.

The built-in Pod self test (described in Section 1 of this manual) detects most Pod malfunctions. Whenever the Mainframe displays a message indicating a self test error, or whenever the Pod appears to be defective or inoperative, you should make a note of the message or symptoms. If the Pod is still covered under the Warranty, or if you want to have the Pod repaired by Fluke, send the Pod to a Fluke Technical Service Center for repair as described below. If you plan to troubleshoot and repair the Pod yourself, continue to "Getting Started" further on in this section.

CAUTION

The Pod's Processor and Interface PCAs (Printed Circuit Assembly) use numerous surface-mounted parts. Replacement of these parts requires a reflow soldering technology. Attempting to replace surface-mounted components using a soldering iron or other conventional means will damage the PCA.

WARRANTY AND FACTORY SERVICE 6-2.

Troubleshooting and repair during the Warranty period should be done by a Fluke Technical Service Center. (See the Warranty statement at the front of this manual for details of the Warranty.) If the Pod is still covered under the Warranty, send the Pod, along with the description of the symptoms to a Fluke Technical Service Center. The Mainframe Operator Manual or Service Manual contains a list of Fluke Technical Service Centers.

After the Warranty period, if you do not want to service the Pod yourself, or if attempted troubleshooting fails to reveal the Pod fault, you may ship the Pod to a Fluke Technical Service Center for repair at a reasonable cost. If requested, a free cost estimate will be provided before any repair work is performed.

INSPECTING A SHIPMENT 6-3.

If you receive an original delivery of this Pod, thoroughly inspect the Pod immediately upon arrival for damage and missing items.

- If the Pod is damaged in any way, file a claim with the carrier. You are responsible for negotiations and final claims with the carrier. If you want to have shipping damage repaired by Fluke, contact the nearest Fluke Technical Center for a quotation.

- Check all material in the container against the enclosed packing list. Fluke will not be responsible for shortages unless you notify us immediately.

SHIPPING THE POD TO FLUKE FOR REPAIR OR ADJUSTMENT 6-4.

Ship the Pod to Fluke prepaid by a package delivery service or "Best Way" (Air Freight from outside North America). Ship the Pod in its original packing carton, or, if that is not available, use a suitable container that is rigid and of adequate size. If you use a substitute container, wrap the Pod in paper and surround it with at least four inches of shock-absorbing material.

GETTING STARTED 6-5.

Troubleshooting the Pod is similar to troubleshooting any other microprocessor-based UUT, and requires the equipment listed in Table 6-1. You should use the Theory of Operation in Section 5 and the schematic diagrams in Section 8 to augment the troubleshooting procedures that are outlined below.

Table 6-1. Required Test Equipment for Pod Troubleshooting

EQUIPMENT TYPE	REQUIRED TYPE
Mainframe	Fluke 9000-Series or Fluke 9100-Series
Interface Pod	Fluke 9000A-80286
Digital Multimeter	Fluke 77
Oscilloscope	Tektronix 485 or equivalent
DC Power Supply	+5V, -5V, and +12V regulated output

NOTE

All references to data and addresses in the following paragraphs are in hexadecimal notation unless otherwise noted.

CAUTION

Static discharge can damage MOS components contained in the Pod. To prevent this possibility, take the following precautions when troubleshooting and/or repairing the unit.

- **Never remove, install, or otherwise connect or disconnect PCAs without disconnecting the Pod from the Mainframe.**
- **Perform all repairs at a static-free work station.**
- **Do not handle ICs or PCAs by their connectors.**
- **Wear a static ground strap when performing repair work.**
- **Use conductive foam to store replacement or removed ICs.**
- **Remove all plastic from the work area (including vinyl and expanded foam, such as Styrofoam®).**
- **Use a grounded soldering iron.**
- **Always place the Pod in a static-free plastic bag for shipping.**

DETERMINING WHETHER THE POD IS DEFECTIVE OR INOPERATIVE 6-6.

The first task of troubleshooting the Pod is to determine whether the Pod is defective or inoperative. This determination is based on the results of the Pod self test described in Section 1. If you have not performed the self test, refer to Section 1 and perform the self test before proceeding with the troubleshooting.

The results of the Pod self test and the behavior of the Pod when it is connected to a known good UUT categorizes the problem into one of the three following groups:

• Defective Pod:	The Pod fails the Pod self test and the Mainframe displays a self test failure code. Refer to "Troubleshooting a Defective Pod" in this section.
• Inoperative Pod:	The Pod is unable to complete the Pod self test. The 9000-Series Mainframe displays an *ATTEMPTING RESET* message. The 9100-Series Mainframe displays a *no database loaded (please reset)* message or a *pod timeout bad UUT clock* message. Refer to "Troubleshooting an Inoperative Pod" in this section.
• Suspected Defective Pod:	The Pod passes the Pod self test but exhibits abnormal behavior when it is connected to a known good UUT. Refer to "Extended Troubleshooting Procedures" in this section.

® Styrofoam is a registered trademark of Dow Chemical Co., Inc.

NOTE

(For 9000-Series Mainframe users)

The 80286H Interface Pod is designed to be used only with 9000-Series Mainframes that have been updated with improved delay lines and probes. Earlier models use a slow TTL part as a delay line, which may provide unstable probe readings at the high clock frequencies (possibly greater than 6 MHz) used with the 80286 CPU. If your Mainframe's Serial Number is lower than 3194000 and is demonstrating such symptoms, contact a Fluke Technical Service Center for advice.

TROUBLESHOOTING A DEFECTIVE POD 6-7.

The following paragraphs describe what to do if the Mainframe displays a self test failure message when the Pod self test is performed. For other error messages, refer to the previous section, "Determining whether the Pod is Defective or Inoperative".

The procedures for troubleshooting a defective Pod are based on the information reported by the self test failure codes (shown in Table 6-2). These self test failure codes provide information that can enable the operator to locate the cause of the Pod failure.

NOTE

The fact that the self test was completed (but still reported an error) is a good indication that the problem is probably located in the UUT Interface Section of the Pod. If the self test was completed, the Processor Section and the Timing Section are probably functioning normally. Those sections are essential for accepting the self test commands and communicating the results to the Mainframe.

Information is available from two self test sequences: the standard self test that is controlled by the Mainframe and performed during the self test procedure, and the enhanced self test that is built into the Pod and performed during power-up and reset sequences. The enhanced self test provides more thorough evaluation of the Pod than is provided by the standard self test alone. The results of both self tests are available after doing a normal self test operation on the Pod (as described in Section 1 of this manual).

NOTE

(For 9020 Mainframe users)

The 9000-Series failure codes described in Table 6-2 are used with the 9005A and 9010A Mainframes. Different error codes are displayed when using the Pod with a 9020A. See Appendix D in the 9020A Operator Manual for a description of the 9020A error codes.

Table 6-2. Self Test Failure Codes

9000-SERIES POD SELF TEST

FAILURE CODE	DESCRIPTION
00	UUT read access failed or enhanced self test failed (to determine which enhanced self test failed, do a READ @ F000 0000 for enhanced self test failure code).
01	UUT write access failed. (See "Preparation for Troubleshooting a Defective Pod.")
02	Control lines cannot be driven. (See "Preparation for Trouble-shooting a Defective Pod.")
03	Enableable Status line(s) failed. (See "Preparation for Trouble-shooting a Defective Pod.")

9100-SERIES POD SELF TEST

FAILURE CODE	DESCRIPTION
2001	Not in self test socket.
2002	Unexpected powerfail.
2003	No powerfail when expected.
2004	Pod does not report ABORT when ABORT is asserted.
2005	Pod reports ABORT unexpectedly.
2007	Pod responds unexpectedly when disabling enableable lines.
2008	Pod responds unexpectedly when setting fault mask.
2009	Pod fails SYNC test/select.
3001	Enableable line $\overline{\text{READY}}$ defective.
3002	Enableable line HOLD defective.

NOTE

The following 9000-Series failure codes pertain to the enhanced self test. These codes are displayed on the 9000-Series Mainframe after a READ @ F000 0000. The 9100-Series failure codes are a continuation of the 9100 self test codes.

9000 FAILURE CODE	9100 FAILURE CODE	DESCRIPTION
00	NA	No enhanced self test failures.
01	1001	ROM test failure.
02	1002	RAM test failure.
03	1003	Self reset failure.
04	1004	UUTON timer failure.
05	1005	Low order address buffer failure.

Table 6-2. Self Test Failure Codes (cont)

06	1006	Address drivability latch failure.
07	1007	Low order address line or data line in UUT cable open.
08	1008	Data buffer input failure.
09	1009	Data drivability latch failure.
0A	1010	High order address buffer failure.
0B	1011	High order address line in UUT cable open.
0C	1012	Data output buffer failure.
0D	1013	Control buffer failure.
0E	1014	Control latch failure.
0F	1015	Status latch failure, or status or control line in UUT cable open.
10	1016	Enableable line failure.
11	1017	Sync circuit failure.
12	1018	Standby read address failure.
13	1019	Overlay RAM read/write failure.
14	1020	Overlay RAM comparator/decoder failure.
15	1021	Breakpoint comparator failure.
16	1022	Self interrupt failure.
17	1023	INTR open or stuck low near processor (U53).
18	1024	Power/Ground monitor failure.
19	1025	NMI open or stuck low near processor (U53).
1A	1026	Capacitor monitor failure.
1B	1027	Low power tri-state failure.

Preparation for Troubleshooting a Defective Pod 6-8.

CAUTION

Any adjustment, maintenance, or repair of the opened Pod under voltage shall be avoided as far as possible and, if done, shall be carried out only by a skilled person who is aware of the hazards involved.

Prepare to troubleshoot your defective Pod as follows:

1. Disassemble the Pod, referring to the material under "Disassembly" later in this section. It is not necessary to separate the two PCAs at this point. The two PCAs should remain securely fastened together with screws to avoid possible problems with electrical connections between them.

2. Look for any obvious problems such as burned components or ICs that are loose in their sockets. Replace components if necessary.

3. Connect the Pod to the Mainframe, and insert the UUT cable plug into the self test socket as shown in Figure 6-1.
4. To initiate the self test from the 9000-Series Mainframe, press the BUS rest key, then press the STOP key. To initiate the self test from the 9100-Series Mainframe, press the MAIN MENU key, select SELFTEST, move the cursor to the right, select POD, and press ENTER.
5. Perform a Write at special address F000 0060 = FFBF to disable the Pod self test. (The Pod self test may be re-enabled by cycling Pod power off and then on, or by a Write at special address F000 0060 = FFFF.)
6. Disable all enableable lines (HOLD and $\overline{\text{READY}}$), and turn off the reporting of active forcing lines.

You are now ready to explore the possible causes of Pod failure that are indicated by the Pod self test failure codes.

Troubleshooting a Defective Pod 6-9.

When the Pod and the Mainframe are prepared for troubleshooting, the test and troubleshooting functions of the Mainframe can be applied to the Pod, much like any other UUT. For example, you can perform read or write operations on the UUT (which is actually the self test socket). The Mainframe no longer knows that the Pod is plugged into its self test socket.

Use the failure code data that was reported as the starting point for subsequent troubleshooting. Use standard troubleshooting techniques to investigate possible sources of the problem.

The internal addresses of the pod can be addressed using the 8XX XXXX address space. This address space is intended only for troubleshooting the Pod and is enabled when self test is disabled. Avoid writing to the Pod's variable and stack spaces, otherwise unpredictable behavior may result.

While investigating the problem, you may find it useful to recreate the self test routine that detected the failure. Procedures for recreating the tests are listed in Table 6-3.

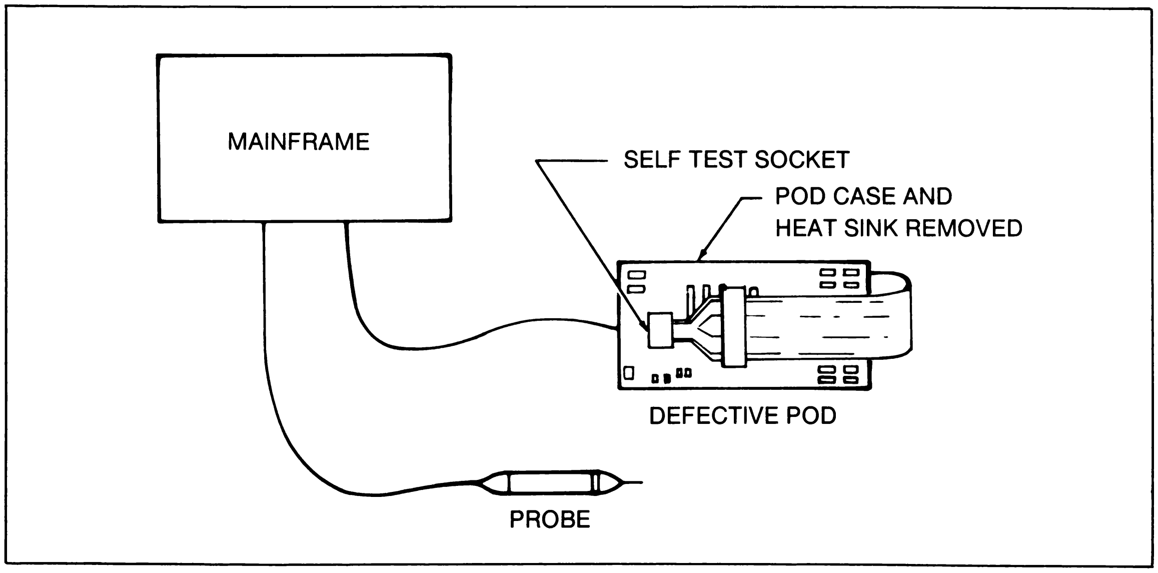


Figure 6-1. Troubleshooting a Defective Pod

Table 6-3. Recreating Self Test Routines

9000-SERIES POD SELF TEST

FAILURE CODE	POD OPERATION	OPERATOR ACTIONS TO RECREATE TEST
00	Reset Pod	Cycle power.
	READ @ 0FF0 = FOOF	Write @ Special address F000 0060 = FFBF. READ @ 0FF0 (If a powerfail error message occurs, check the power detection circuits.)
01	WRITE @ 0FF0=F00F	WRITE @ 0FF0=F00F.
02	Test control lines	Bus Test.
03	Send command to enable all Enableable lines and verify that a Pod timeout occurs (this timeout is transparent to the user).	Enable $\overline{\text{READY}}$ and HOLD; verify that time-out occurs.

9100-SERIES POD SELF TEST

FAILURE CODE	POD OPERATION	OPERATOR ACTIONS TO RECREATE TEST
2001	N/A	None.
2002	Set the VCC pins on self test socket to high (no power-fail).	None.
2003	Set the VCC pins on self test socket to low (induce power-fail).	None.
2004	Mainframe asserts ABORT line, Pod does not respond with PODINT.	None.
2005	Pod asserts PODINT, erroneously reporting an aborted operation.	None, check ABORT and PODINT.
2007	HOLD and $\overline{\text{READY}}$ are disabled.	Disable enableable lines. Pod should not report timeout.
2008	Various fault reports are enabled and disabled.	Disable enableable lines. Use Mainframe Setup to selectively turn on and off bits of the fault mask. Pod should not report timeout.
2009		Check suspect Sync mode opertion on a known good UUT.
3001	$\overline{\text{READY}}$ is enabled.	
3002	HOLD is enabled.	

NOTE

The rest of the 9000-Series failure codes in this table pertain to the enhanced self test. These codes are displayed on the 9000-Series Mainframe after a READ @ F00 0000. The 9100-Series failure codes are a continuation of the 9100 self test codes.

Table 6-3. Recreating Self Test Routines (cont)

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION	OPERATOR ACTIONS TO RECREATE TEST
00	NA	No enhanced self test failures.	
01	1001	ROM test at FF 8000 through FF FFF8.	See Table 6-4, but prefix an "8" to each address from the table.
02	1002	RAM test 0000 through 3FFE.	A Pod should not perform a RAM test over its own variable data area 800 0000 through 800 1FFE. Overlay RAM at 800 2000 through 800 3FFE can be tested.
03	1003	Pod resets itself.	None.
04	1004	UUTON occurs.	UUT READ access, e.g. READ @0.

NOTE

The following five failure codes are derived from a series of Read operations. These Read operations compare expected data with the data returned and the value of the drivability latches. To recreate these tests, perform the following operations, observing the returned data and drivability errors (these are memory word accesses).

OPERATION	EXPECTED RESULT
READ @ 000000	0000
READ @ FFFFFE	FFFF
READ @ 555554	5455
READ @ AAAAAA	ABAA
READ @ CCCCCC	CDCC
READ @ 333332	3233

NOTE

The address drivability latches are automatically checked and address drivability faults reported if their values do not match the address accessed. However, the data latches must be explicitly read and verified by a READ @ 800 F600. Refer to Table 6-5.

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION
05	1005	Data returned, address drivability, and data drivability do not match the address sent.
06	1006	Address drivability latch does not match address sent, but data and data latch do match the address sent.
07	1007	Address latch is correct, data and data latch are different. (Alternative: see next page.)
08	1008	The data drivability latches match the address sent, but the data read is different from the address sent.
09	1009	Data drivability latch does not match either address sent or data received.
0A	1010	High order address drivability latch does not match the address sent.

Table 6-3. Recreating Self Test Routines (cont)

NOTE

The high order address lines are tested by disabling the two low order address latches, and enabling the high order address latch. To do this, Write @ special address F000 0000 = 0A (high order address buffer test. At this point, STOE1 is active and $\overline{\text{STOE0}}$ is inactive until a reset or another test is specified. Disable address drivability reporting during this test (see 9000-Series error code 12).

The following two failure codes are derived from a series of Read operations. These operations are performed with byte addresses:

OPERATION	EXPECTED RESULT
READ @ 000000	XX00
READ @ FFFFFF	XXFF
READ @ 555555	XX55
READ @ AAAAAA	XXAA
READ @ CCCCCC	XXCC
READ @ 333333	XX33

The address drivability latch is explicitly verified by a READ @ 800 F400. The data drivability latches are verified by a READ @ 800 F600. Refer to Table 6-5.

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION
07	1007	Data on latch does not match A0 values.
0B	1011	Data returned does not match high byte of address.

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION
0C	1012	Data output buffer tested by WRITE operations: WRITE @ 0 = 0000 WRITE @ 0 = FFFF WRITE @ 0 = 5555 WRITE @ 0 = AAAA WRITE @ 0 = CCCC WRITE @ 0 = 3333

Table 6-3. Recreating Self Test Routines (cont)

NOTE

The following operations diagnose control and status problems. Use the Probe to investigate these lines.

OPERATION	OPTION
READ @ 0	memory word
READ @ 0	memory byte
READ @ 0	I/O word
READ @ 0	I/O byte
WRITE @ 0	memory word
WRITE @ 0	memory byte
WRITE @ 0	I/O word
WRITE @ 0	I/O byte
WRITE CONTROL	$\overline{\text{LOCK}}$ active
WRITE CONTROL	HLDA active
WRITE CONTROL	$\overline{\text{PEACK}}$ active

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION
0D	1013	Control buffer failure is diagnosed when neither status nor control latches match the expected state. (Read last status at special address F000 0052 for the status of the access.)
0E	1014	Control latch failure is diagnosed when status matched the expected state, but control errors are reported from the control latch.
0F	1015	Status latch failure, or status or control line in cable open, is indicated when control latch has the correct value, but status latch is incorrect.

9000 FAILURE CODE	9100 FAILURE CODE	POD OPERATION
10	1016	The Pod can set its enableable lines to both states and check that the status latch caught both. To recreate, disable the enableable lines and force the lines to each state by using a jumper at the self test socket, then Read Status.
11	1017	Sync circuit failure is detected by selecting sync modes and latching the $\overline{\text{PODSYNC}}$ line on U24 and U27. Use an oscilloscope to check each sync mode.
12	1018	Standby read address failure. The Pod can check its standby read latches by setting STOE1 active. With STOE1 active, the drivability latches capture Standby address rather than UUT access address. Read the address latches at 800 F400 and 800 F000 to verify that the correct standby read address was captured.
13	1019	Overlay RAM read/write failure. (See 9100-Series Fluke code 1002.)
14	1020	Overlay RAM comparator/decoder failure.
15	1021	Breakpoint comparator failure. Cannot be recreated by the user.

Table 6-3. Recreating Self Test Routines (cont)

16	1022	Self-interrupt failure is detected. When an interrupt acknowledge cycle is complete, the control bits are expected to be in a certain state. The user can force an interrupt acknowledge cycle. If no control drivability errors are reported, interrupt acknowledge is working correctly.
17	1023	INTR open or stuck low is checked by enabling interrupts and setting COD/$\overline{\text{INTA}}$ high (an instruction fetch cycle). An interrupt acknowledge cycle should occur.
18	1024	Power/Ground monitor failures are induced by port bits. Cannot be recreated by the user.
19	1025	NMI open or stuck low. (A powerfail cycle causes an NMI.) Cannot be recreated by the user.
1A	1026	Capacitor monitor failure is checked by a pass condition with a 0.047 μF capacitor. An open collector buffer pulls it low to fail. Cannot be recreated by the user.
1B	1027	Low power tri-state failure. The Pod's outputs float high. Since the Pod in troubleshooting mode stops communicating with the Mainframe if low power is detected, this test cannot be recreated by the user.

NOTE
(For 9020 Mainframe users)

The same basic techniques may be used for the additional failure codes that may be obtained when using the 9020A Micro-System Trouble-shooter mainframe in the remote mode.

To recreate any enhanced self test routine, perform a Write at special address F000 0000 = test number (in hex), where "test number" is the number of the 9000-Series failure code shown in Table 6-2. When the Write is entered, the test is performed once. To perform a loop of the Write operation, press the LOOP key.

See Table 6-5 for the Pod's memory map and bit definitions of selected Pod addresses.

TROUBLESHOOTING AN INOPERATIVE POD 6-10.

The following paragraphs describe what to do if the Pod is inoperative. An inoperative Pod does not respond to the Mainframe.

If you correct a problem while using the procedures provided in this section, try the Pod self test again. If the Pod is still inoperative, continue with the procedures in this section. However, if the Mainframe displays a Pod self test failure message, refer to "Troubleshooting a Defective Pod" earlier in this section. When the Pod is again communicating with the Mainframe, you may use the Pod to help troubleshoot itself with the information contained in "Troubleshooting a Defective Pod".

Preparation for Troubleshooting an Inoperative Pod 6-11.

An inoperative Pod is like any other microprocessor-based UUT that is not operating properly; the easiest way to fix an inoperative Pod is by using a Mainframe and a good Pod. Preparation instructions also apply to troubleshooting without a good Pod, but note that the detailed troubleshooting steps that follow only apply to using the second Mainframe and Pod. Prepare to troubleshoot the inoperative Pod by performing the following steps:

1. Disassemble the Pod, referring to "Disassembly" later in this section, but do not separate the two PCAs.
2. Look for any obvious problems, such as burned components or ICs that are loose in their sockets. Replace components if necessary. After such defects are found and repaired, try the self test again.
3. Remove the Pod microprocessor from its socket.
4. If a second Mainframe is available, connect the Pod cable plug from the inoperative Pod to the second Mainframe to supply the inoperative Pod with power. If a second Mainframe is not available, connect a +5V (2 amp.), -5V (150 mA), and +12V (100 mA) power supply to the Pod as shown in Figure 6-2. An easy place to make the power connections is at the connector that normally connects the Pod to the Mainframe.

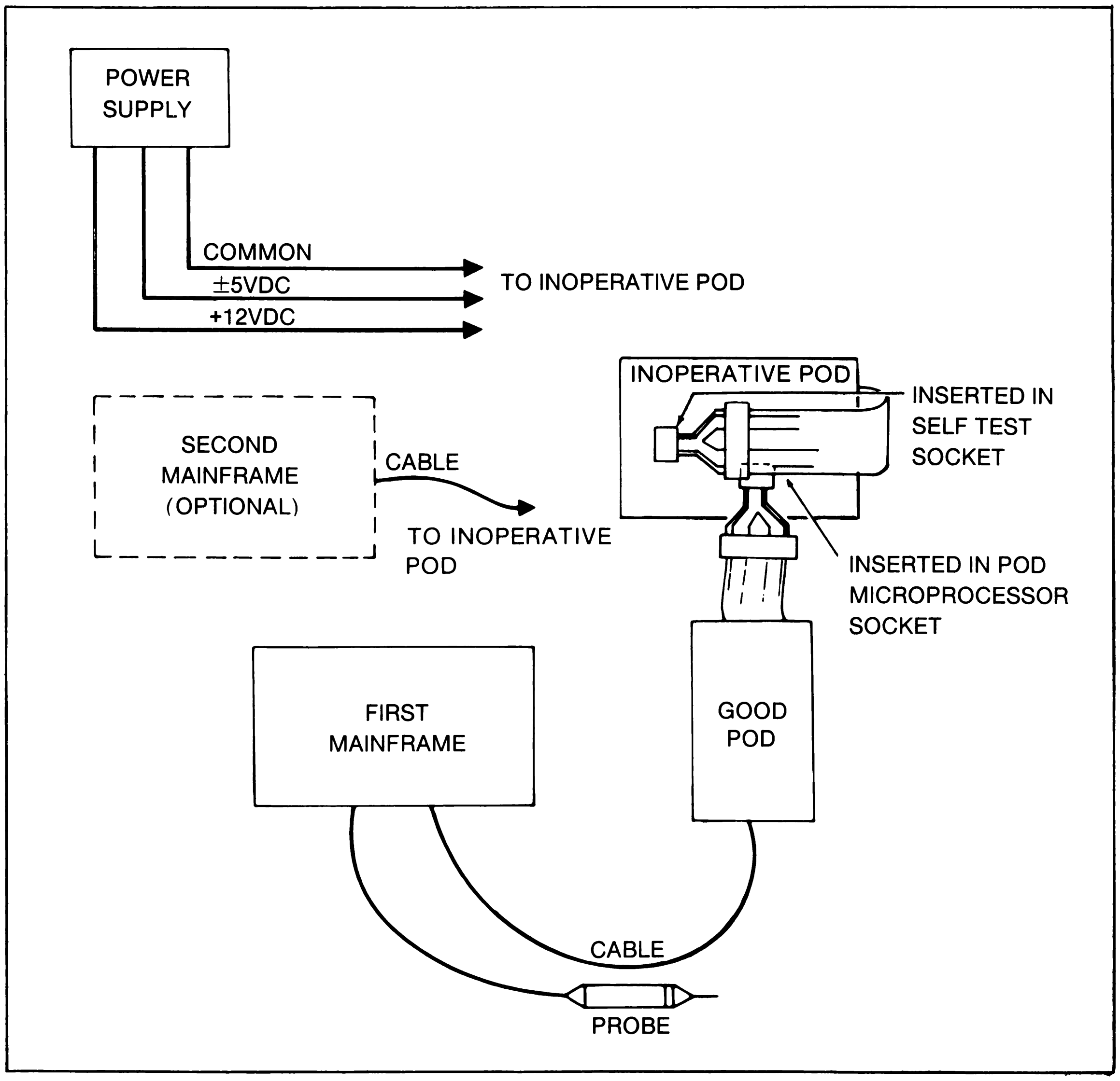


Figure 6-2. Troubleshooting an Inoperative Pod

CAUTION

Do not operate the Pod with the voltage supply exceeding 5.25 volts, or damage to the Pod could result.

5. Provide a clock signal for the inoperative Pod by inserting the UUT cable of the inoperative Pod into its own self test socket. (Make sure the clock is working properly.) An alternative source for a clock signal is a known good UUT or frequency generator.

6. Connect the Mainframe to the good Pod as shown in Figure 6-2. Apply power to the Mainframe, then install the UUT cable plug of the good Pod into the microprocessor socket of the inoperative Pod.

CAUTION

Do not apply or remove power to the good Pod with the UUT cable connected between the good Pod and the inoperative Pod.

CAUTION

Do not separate the PCAs of the inoperative Pod with power applied to the inoperative Pod. Failure to comply with this can damage components in the Pod. The PCAs should be securely fastened together with the proper screws before applying power.

Procedure for Troubleshooting an Inoperative Pod 6-12.

Use the following steps as a guide for using a good Pod to troubleshoot an inoperative Pod. The circuits and components mentioned in these steps appear in the schematic diagrams in Section 8, and the circuits are described in the Theory of Operation in Section 5.

The following procedures are intended only to direct the technician toward the source of a problem. In each case, once an irregularity is detected, it is up to the technician to pursue the problem further. When a problem is identified, standard troubleshooting procedures should be used to locate the source of the trouble. This involves such things as checking and verifying activity of enabling and timing signals, input and output signals, and logic.

NOTE

When performing looping read or write operations with a synchronized oscilloscope connected to the Mainframe, use the Quick-Looping Read or Write feature described in Sections 2 and 3 to obtain a brighter signal trace on the scope.

1. Prepare the Pod as outlined previously in this section (Preparation for Troubleshooting an Inoperative Pod). Position the inoperative Pod so that the processor board (the one with the microprocessor socket) is upwards. Apply power to the inoperative Pod. If a second Mainframe is used to supply power to the Pod to be tested, press the STOP key on the second Mainframe to prevent repetitive Pod resets.

2. Check that the microprocessor's clock signal is present at U53, pin 31. If it is not, trace the clock signal to the fault.

3. Check that RESET is not being asserted (high) at U53, pin 29.

4. If the Pod is plugged into a second 9000-Series Mainframe, press the second Mainframe's BUS TEST key, then press the STOP key. The Bus Test resets the inoperative Pod. Pressing the STOP key prevents the Mainframe from continuing to reset the Pod. If the second 9000-Series Mainframe is not available, use the first Mainframe's probe to pulse the inoperative Pod's Reset line low at pin 23 of connector P3.

5. After the inoperative Pod is initialized, try a BUS TEST with the first Mainframe. If you get a timeout message, use the Setup functions to disable the enableable lines.

6. If the Bus Test works now, check the $\overline{\text{READY}}$ and HOLD status enablers.

7. If a timeout still occurs, check the clock circuit.

8. Do a ROM TEST and a RAM TEST. Refer to Table 6-4 for the ROM test addresses and the expected results. Test RAM over the range 0000 to 3FFE.

   If the ROM Test and RAM Test are successful, the next thing to check is the data communication with the Mainframe.

9. Check the path of data from the Mainframe to the Pod. If a second Mainframe is being used and has timed out, the data 4B or AB is probably being placed on the second Mainframe's data lines. The first Mainframe can read the data lines of the second Mainframe by performing a read operation at D800. (D800 is the address of incoming data on the Processor PCA.) The data should be XXAB (X = any value). Bit 8 of the data should be the value of $\overline{\text{MAINSTAT}}$. Bit 11 of the data should be $\overline{\text{LOWPOWER}}$ (it should be high). The rest of the bits in the data are indeterminate.

   If a second Mainframe is not being used, you may check the incoming data path by performing a looping read operation at D800. Set the probe stimulus to toggle high and low pulses, then apply the stimulus pulses to the data lines, and check whether all bits are readable high and low.

10. Check the path of data from the Pod back to the Mainframe by powering the inoperative Pod from a separate power supply as explained earlier in this section. (Do not plug the inoperative Pod into a Mainframe for this step.) Powering the inoperative Pod from a separate power supply allows the Pod to drive the Pod-Mainframe data bus without interference from the Mainframe. Leave the good Pod plugged into the inoperative Pod's microprocessor socket. Test the inoperative Pod using the following procedure with the good Pod and Mainframe:

Table 6-4. ROM Test Addresses

ROM TEST	TEST ADDRESS	READ EXPECTED RESULT AT:
9000-Series ROM Test	FF 8000 through FF FFF8	FF FFFA
Pod Quick Test		FF FFFC
9100-Series ROM Test		FF FFFE

a. Write at C800, data = 01XX (where XX can be any number), to set bit 0 in U36. This bit enables the outputs of U4, which drives data to the Mainframe. The lower byte XX is latched into U4 and should appear at the outputs of U4. Check the outputs with the logic probe. The byte should also appear at connector P3 and at the Mainframe end of the Pod-to-Mainframe cable (on the same pin numbers as those of P3). Test the data bus with the lower byte XX equal to 00, 55, AA, and FF to discern stuck, open, or tied bits.

b. Write at C800, data = 0155, to set the data from U4 to hex 55. Toggle the $\overline{\text{PODRESET}}$ signal (P3-23 or P2-7) with the Mainframe's probe in pulser mode. Perform a looping Read Status operation with the good Pod; the probe pulses should toggle the RESET status bit. The outputs of U4 should now be tri-stated, and all output port bits should be set low. If this does not occur, check the Long Reset Generator section of the inoperative Pod. The outputs of the Reset Monitor should all be high (RST-WAS-SELF, RST-WAS-ABORT, RST-WAS-BREAK). If not, there is a fault in the Reset Monitor.

c. Use the probe to determine that BUSY-CTR-RCO (U16-15) is high, $\overline{\text{UUT-ON-REQ}}$ (U2-3) is high, $\overline{\text{RUNUUTON}}$ (U8-12) is high, and $\overline{\text{UUTON}}$ (U34-16) is high. If not, the fault is related to the Bus Switch Timing Circuit.

d. Write at C800, data = 0000 (then data = 0800), to toggle the $\overline{\text{PODSTAT}}$ signal. Use the logic probe to check its activity all the way though to the end of the Pod-to-Mainframe cable.

NOTE

Table 6-5 contains a description of the bit definitions of selected Pod addresses, including the addresses listed in the previous paragraphs.

If all these tests are good, you should no longer have an inoperative Pod, but you may still have a defective Pod. If the Pod is still defective, refer to "Troubleshooting a Defective Pod" earlier in this section.

EXTENDED TROUBLESHOOTING PROCEDURES 6-13.

The troubleshooting procedures provided in this section supplement the circuit checks performed on the Pod during the Pod self test. These procedures are appropriate for use with a Pod that passes the Pod self test but does not appear to function normally when used with a Mainframe and a good UUT. If a Pod fails self test, it would be better to begin troubleshooting with the procedure provided in "Troubleshooting a Defective Pod" earlier in this section.

Potential problems that may exist in a Pod that passes the Pod self test may be divided into two categories:

- Unchecked circuits
- Timing and noise problems

Unchecked Circuits 6-14.

Some circuits on the Pod are not checked by the Pod self test. Use the schematics located in Section 8 to locate potential problems in the following circuits:

- Overlay $\overline{\text{READY}}$ source
- RUN UUT

- Reset monitor Abort and Break sources
- Status line gating
- Mainframe, missing clock, and power-up resets
- BUS-SYNC and INTR-ACK-SYNC signals stuck low at input to sync mux
- Control line outputs during standby reads
- Shorted or out of tolerance drivability resistors
- Standby read enable

Timing and Noise Problems 6-15.

Timing or noise problems are usually caused by components that are still functioning but not within the component specifications. The best way to check this problem is to look at suspect signals using an oscilloscope synchronized to either address or data. Look for slow rise times or signals driven to marginal levels. If the part is too slow, it might fail on the UUT but pass the Pod self test because of narrower design margins on the UUT.

If a part has marginal drive capabilities, the added load of a UUT environment might cause it to fail. Be sure to note that inputs can malfunction (they may leak, perhaps) and put too much load on an output causing either low levels, slow rise times, or both.

DISASSEMBLY 6-16.

To gain access to the two PCAs in the Pod, perform the following steps:

1. Remove the Pod's round shielded connector from the Mainframe socket.
2. Remove the UUT cable plug from the self test socket.
3. Turn the Pod over on its top (with the large Pod decal facing up). Remove the four Phillips screws that hold the case halves together, and remove the top and bottom case halves. Place the PCAs so that the self test socket (on the processor PCA) is facing up.
4. Remove the five Phillips screws that fasten the shield to the PCA. Remove the shield.
5. Remove the four Phillips screws that connect the heat sink to the processor PCA, and remove the heat sink.

NOTE

The heat sink is not needed for heat dissipation unless the Pod is fully assembled. If the two PCAs are removed from the top and bottom cases, the heat sink may be removed during Pod operation and troubleshooting.

NOTE

When the shield and the heat sink are removed, all the components are exposed. It should not be necessary to separate the two PCAs while troubleshooting except to replace components.

6. To separate the two PCAs, turn them over so that the self test socket is facing down. Remove the four Phillips screws at the corners of the PCAs, and carefully pull the boards apart at the two connectors along the sides.

Table 6-5. Memory Map and Bit Definitions of Selected Pod Addresses

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	READ/WRITE
0000-1FFE	Kernel RAM																R/W
2000-3FFE	Overlay RAM																R/W
C000	A23	A22	A21	A20	A19	A18	A17	A16 Standby Read Address	A15	A14	A13	A12	A11	A10	A9	A8	W
C800	$\overline{\text{STGND1}}$	$\overline{\text{STGND0}}$	$\overline{\text{MASK-NMI}}$	INTR-SELF	PODSTAT	PODINT	STREADY	ENABLE	POD7	POD6	POD5	POD4	POD3	POD2	POD1	POD0	W
D000	$\overline{\text{STBYREADEN}}$	RUNUUT-REQ	ENINTR	ENREADY	ENHOLD	SYNCSEL2	SYNCSEL1	SYNCSEL0	STOE1	$\overline{\text{STOE0}}$	STHOLD	$\overline{\text{SLOW-SYNC-MODE}}$	WRITE COD/INTA	WRITEPEACK	WRITEHLDA	WRITELOCK	W
D800	BREAK-TEST	RST-WAS-ABORT	RST-WAS-SELF	RST-WAS BREAK	$\overline{\text{LOWPOWER}}$	—	$\overline{\text{ABORT}}$	$\overline{\text{MAINSTAT}}$	POD7	POD6	POD5	POD4	POD3	POD2	POD1	POD0	R
E000	—	—	—	—	—	—	—	—	—	$\overline{\text{PODSYNC}}$	INTR-CTL-TIM	NMI-CTL-TIM	$\overline{\text{SELFTEST}}$	STAO	$\overline{\text{STS1}}$	$\overline{\text{STS0}}$	R
E800	—	—	—	—	—	—	—	—	A23	A22	A21	A20	—	—	—	—	W
F000	A15	A14	A13	A12	A11	A10	A9	A8 Address Drivability	A7	A6	A5	A4	A3	A2	A1	A0	R
F200	—	—	GNDFAIL9	GNDFAIL35	POWERFAIL62	POWERFAIL30	$\overline{\text{PODSYNC}}$	$\overline{\text{CAPFAIL}}$	$\overline{\text{ERROR}}$	$\overline{\text{BUSY}}$	PEREQ	RESET	INTR	NMI	HOLD	$\overline{\text{READY}}$	R
F400	COD/$\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S0}}$	$\overline{\text{S1}}$	$\overline{\text{BHE}}$	$\overline{\text{PEACK}}$	HLDA	$\overline{\text{LOCK}}$	A23	A22	A21	A20 Address Drivability	A19	A18	A17	A16	R
F600	D15	D14	D13	D12	D11	D10	D9	D8 Data Drivability	D7	D6	D5	D4	D3	D2	D1	D0	R
F800	SELF RESET*																R/W
F900	CLEAR_RESET_MONITOR*																R/W
FA00	UUTON REQUEST*																R/W
FB00	A23	A22	A21	A20	A19 Overlay Base Address	A18	A17	A16	A15	A14	A13	—	—	ENABLE-BKPT	OVLY-RDY-EN	ENABLE-OVLY	W
FC00	—	—	—	—	—	—	—	—	A23	A22	A21	A20 Breakpoint Address	A19	A18	A17	A16	W
FD00	A15	A14	A13	A12	A11	A10	A9	A8 Breakpoint Address	A7	A6	A5	A4	A3	A2	A1	—	W
FF8000-FFFFFE	Kernel ROM																R

*No data exists at these addresses. Writing to these addresses produces a Pod processor reset, a clear reset monitor, and a request for a UUTON cycle.

Section 7
List of Replaceable Parts

INTRODUCTION 7-1.

This section contains an illustrated parts list for the instrument. Components are listed alphanumerically by assembly.

Parts lists include the following information:

1. Reference Designation.
2. Static Warning marker.

CAUTION

Devices indicated by an asterisk (*) in the listing are subject to damage by static discharge.

3. Description of Each Part.
4. Fluke Stock Number.
5. Federal Supply Code for Manufacturers (see the 9000-Series Troubleshooter Service Manual for Code-to-Name list).
6. Manufacturer's Part Number.
7. Total Quantity of Components Per Assembly.
8. Recommended spares quantity. This entry indicates the recommended number of spare parts necessary to support one to five instruments for a period of two years. This list presumes an availability of common electronic parts at the maintenance site. For maintenance for one year or more at an isolated site, it is recommended that at least one of each assembly in the instrument be stocked.

HOW TO OBTAIN PARTS 7-2.

Components may be ordered directly from the manufacturer by using the manufacturer's part number, or from the John Fluke Mfg. Co., Inc. or an authorized representative by using the Fluke Stock Number.

In the event the part ordered has been replaced by a new or improved part, the replacement will be accompanied by an explanatory note, and installation instructions if necessary.

To ensure prompt and efficient handling of your order, include the following information.

1. Quantity.
2. Fluke Stock Number.
3. Description.
4. Reference Designation.
5. Printed Circuit Board Part Number and Revision Letter.
6. Instrument Model and Serial Number.

A Recommended Spare Parts Kit for your basic instrument is available from the factory. This kit contains those items listed in the RSQ column of the parts lists in the quantities recommended.

Parts price information is available from the John Fluke Mfg. Co., Inc. or its representative. Prices are also available in a Fluke Replacement Parts Catalog, which is available upon request.

MANUAL STATUS INFORMATION 7-3.

Table 7-5 lists the PCAs and their revision levels documented in this manual. To identify the revision level of the PCAs used in your instrument, refer to the revision letter on the component side of each PCA.

As changes and improvements are made to the instrument, they are identified by incrementing the revision letter marked on the affected PCA. These changes are documented on a supplemental change/errata sheet which, when applicable, is inserted at the front of the manual.

Table 7-1. 9000A-80286H Final Assembly
(See Figure 7-1.)

REFERENCE DESIGNATOR -A>	-NUMERICS---->	S	----DESCRIPTION----	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	R S -Q	N O T -E-
		*	PROCESSOR PCA	818799	89536	818799	1		
		*	INTERFACE PCA	818807	89536	818807	1	1	
H	1		SCREW,MACH,PH,P,STL,4-40X0.750	115063	89536	115063	4		
H	2		SCREW,MACH,PHP SEMS,STL,4-40X1/4	185918	89536	185918	11		
H	3		SCREW,MACH,PH,P,STL,4-40X0.625	145813	89536	145813	2		
H	4		WASHER,LOCK,INTRNL,STEEL,#4	110403	89536	110403	2		
MP	1		CLIP, HEATSINK, 24 PIN	607655	89536	607655	2	1	
MP	2		HEATSINK 80286 POD	802066	89536	802066	1		
MP	3		CLIP HEATSINK 68 PIN	803205	89536	803205	1		
MP	4		SHIELD, 80286 POD	803189	89536	803189	1		
MP	5		SHELL TOP	744797	89536	744797	1		
MP	6		SHELL, BOTTOM	648881	89536	648881	1		
MP	7		DECAL, POD	844089	89536	844089	1		
MP	8		DECAL, SPEC	844092	89536	844092	1		
MP	9		WARNING DECAL 8086	659805	89536	659805	1		
MP	10		LABEL, STATIC CAUTION	605808	89536	605808	1		
MP	11		SPACER,HEX,ALUM,4-40X0.375	187575	89536	187575	2		
MP	12		BAG,STATIC SHIELDING,15"X18"	681015	89536	681015	1		
TM	1		INSTRUCTION MANUAL	804625	89536	804625	1		
TM	2		QUICK REFERENCE CARD	824409	89536	824409	1		
U	1	*	PROGRAMED V1.0	804591	89536	804591	1	1	
U	2	*	PROGRAMED V1.0	804583	89536	804583	1		
U	3	*	PROGRAMED V1.0	804575	89536	804575	1	1	
U	29	*	PROGRAMED V1.0	804617	89536	804617	1	1	
U	34	*	PROGRAMED V1.0	804609	89536	804609	1		
U	37	*	PROGRAMED V1.0	844097	89536	844097	1	1	
U	48	*	PROGRAMED V1.0	844100	89536	844100	1	1	
U	53	*	IC,NMOS,16 BIT MPU,12.5 MHZ,PGA	800946	89536	800946	1		
W	1	*	80286 UUT CABLE ASSY.	805754	89536	805754	1		
W	2		CABLE,80286 POD	803171	89536	803171	1	1	
XU	1		SOCKET,IC,CHIP CARRIER,68 PIN,0.155	758235	89536	758235	1		

An * in 'S' column indicates a static-sensitive part.

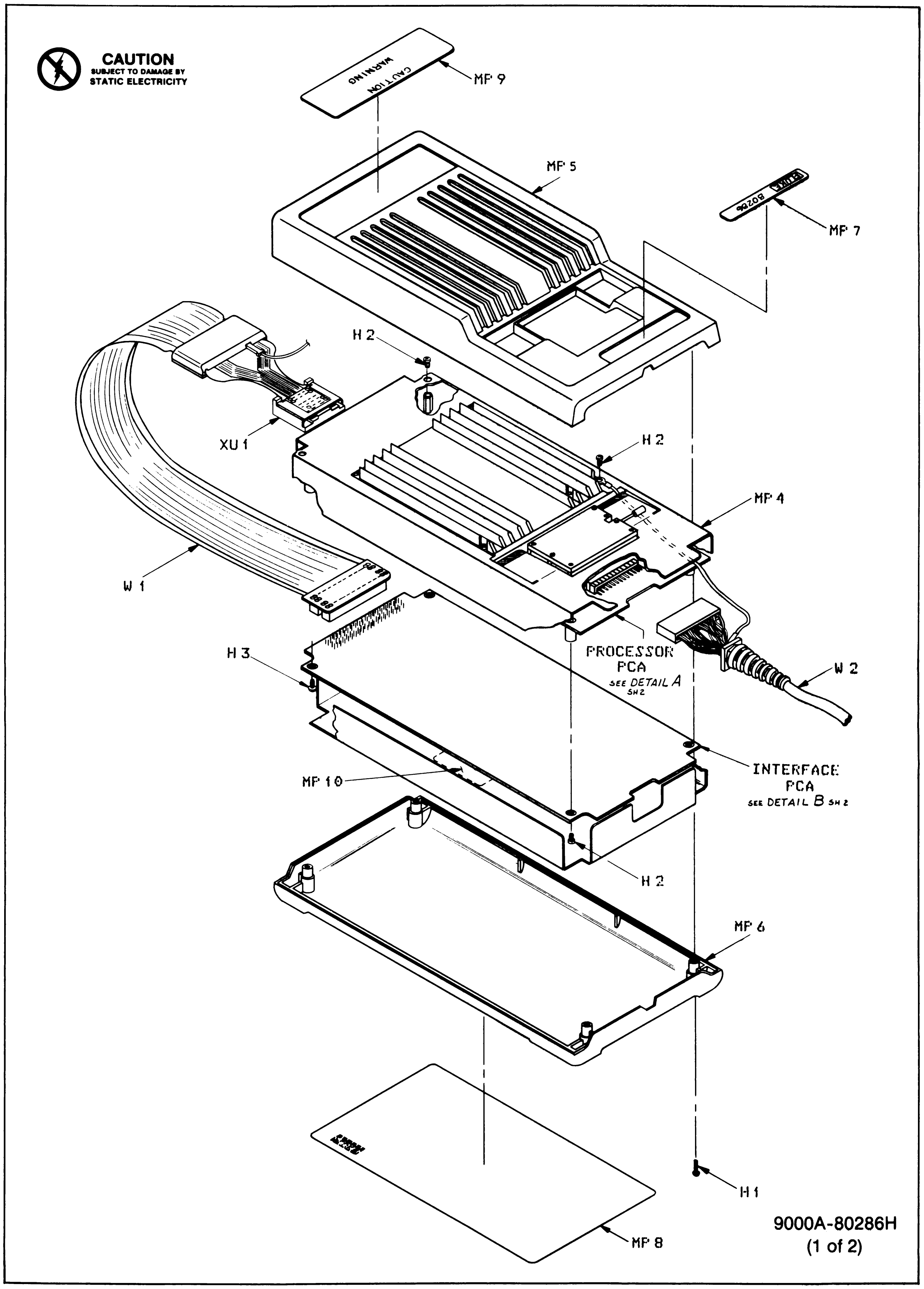

CAUTION
SUBJECT TO DAMAGE BY STATIC ELECTRICITY
CAUTION
WARNING
MP 9
MP 5
MP 7
H 2
XU 1
H 2
MP 4
W 1
H 3
PROCESSOR PCA SEE DETAIL A SH 2
W 2
INTERFACE PCA SEE DETAIL B SH 2
MP 10
H 2
MP 6
H 1
MP 8
9000A-80286H (1 of 2)

Figure 7-1. 9000A-80286H Final Assembly

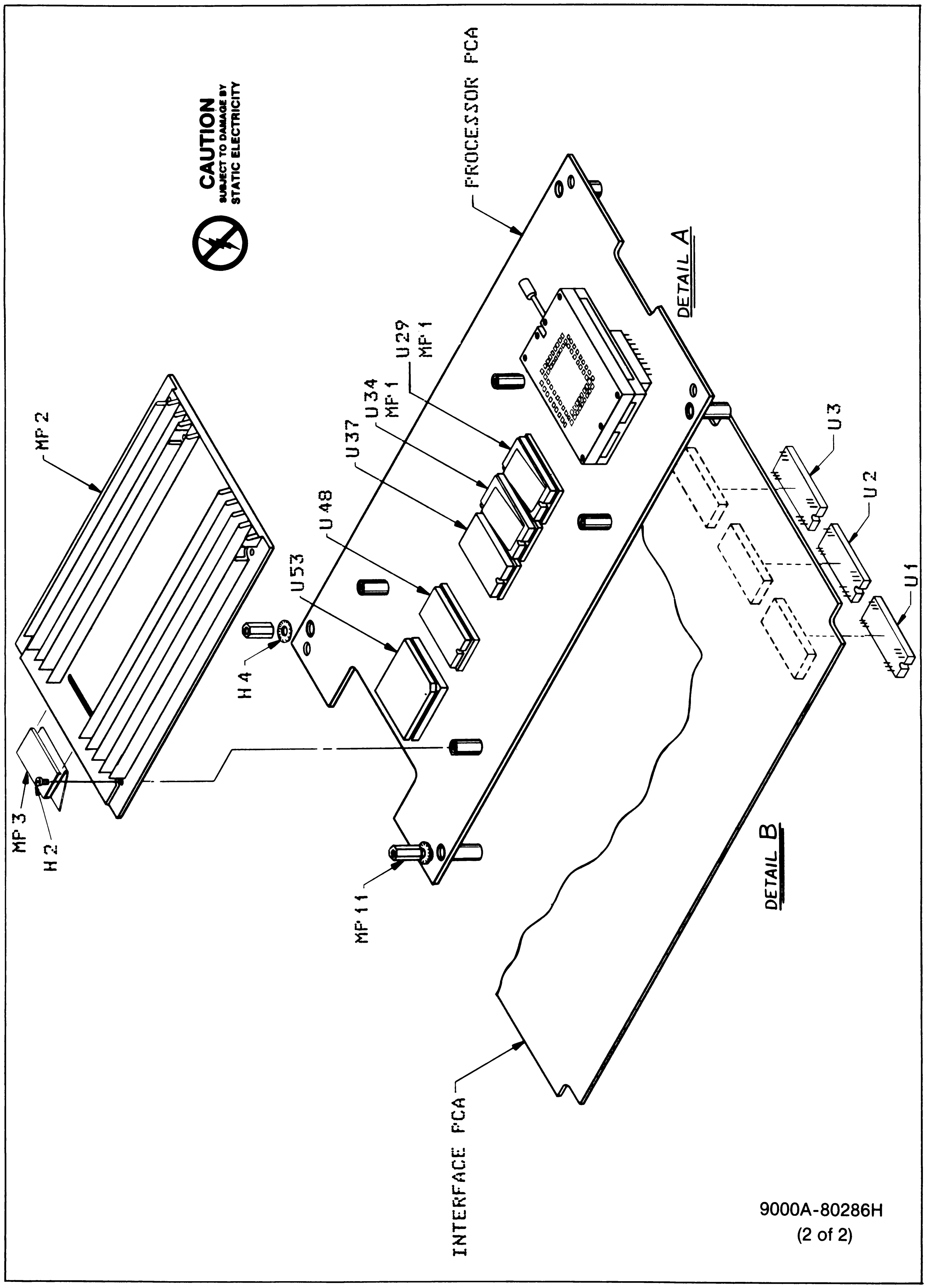
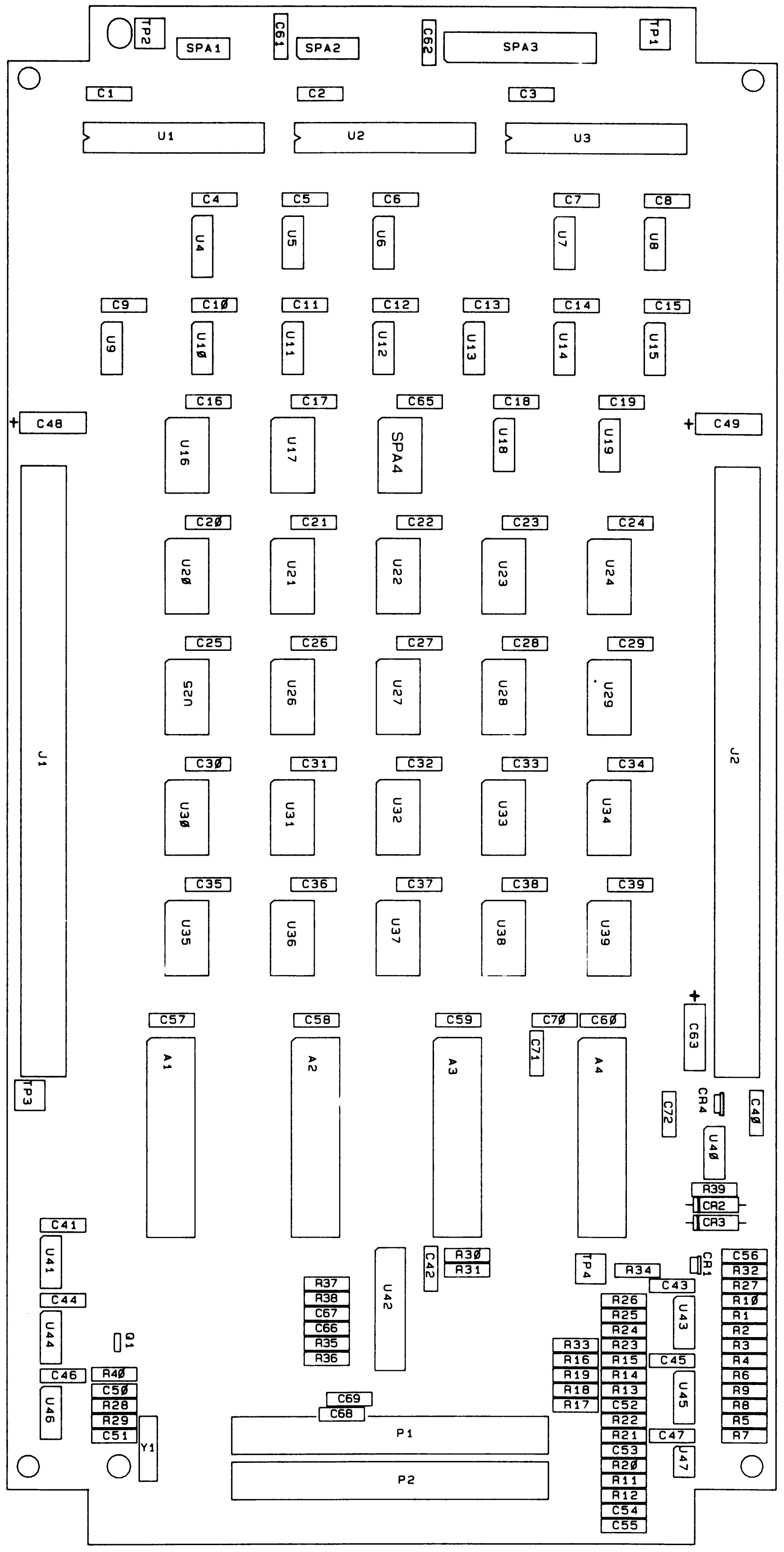


Figure 7-1. 9000A-80286H Final Assembly (cont.)

Table 7-2. Interface PCA
(See Figure 7-2.)

REFERENCE DESIGNATOR -A>	-NUMERICS---->	S	----DESCRIPTION----	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	R S -Q	N O T -E-
A	1- 4	*	HYBRID ASSY, 80286 POD, FINAL TEST	803106	89536	803106	4		
C	1- 47, 52-		CAP,CER,0.01UF,+-10%,50V,X7R,1206	747261	89536	747261	59	1	
C	62, 65			747261					
C	48, 49, 63		CAP,TA,10UF,+-20%,25V	772491	89536	772491	3		
C	50, 51		CAP,CER,27PF,+-10%,50V,COG,1206	800508	89536	800508	2		
C	66- 69		CAP,CER,5.6PF,+-10%,50V,COG,1206	782409	89536	782409	4		
C	70- 72		CAP,CER,1000PF,+-10%,50V,COG,1206	747378	89536	747378	3		
CR	1, 4		DIODE,SI,BV=70.0V,IO=50MA,DUAL,SOT23	742320	89536	742320	2		
CR	2, 3	*	DIODE,SI,SCHOTTKY BARRIER,SMALL SIGNL	313247	28484	HP5082-6264	2		
J	1, 2		SOCKET,2 ROW,PWB,0.100CTR,80 POS	806281	89536	806281	2		
P	1, 2		HEADER,2 ROW,0.100CTR,40 PIN	811554	89536	811554	2	1	
Q	1		TRANSISTOR,SI,NPN,SMALL SIGNAL,SOT23	742676	89536	742676	1	1	
R	1, 4, 8,		RES,CHIP,CERM,510K,+-5%,0.125W,1206	746800	89536	746800	4		
R	18			746800					
R	2, 3, 5,		RES,CHIP,CERM,10K,+-5%,0.125W,1206	746610	89536	746610	8		
R	7, 15, 17,			746610					
R	30, 31			746610					
R	6, 16		RES,CHIP,CERM,2.2M,+-5%,0.125W,1206	811778	89536	811778	2		
R	9, 10, 19,		RES,CHIP,CERM,2.2K,+-5%,0.125W,1206	746479	89536	746479	4		
R	27			746479					
R	11, 13		RES,CHIP,CERM,26.1K,+-1%,0.125W,1206	807685	89536	807685	2		
R	12, 14		RES,CHIP,CERM,30.1K,+-1%,0.125W,1206	801258	89536	801258	2		
R	20		RES,CHIP,CERM,267,+-1%,0.125W,1206	817411	89536	817411	1		
R	21		RES,CHIP,CERM,432,+-1%,0.125W,1206	811885	89536	811885	1		
R	22, 40		RES,CHIP,CERM,1.33K,+-1%.0.125W,1206	801423	89536	801423	2		
R	23		RES,CHIP,CERM,681,+-1%,0.125W,1206	806430	89536	806430	1		
R	24, 26, 34		RES,CHIP,CERM,2K,+-1%,0.125W,1206	807172	89536	807172	3		
R	25		RES,CHIP,CERM,4.02K,+-1%,0.125W,1206	783266	89536	783266	1		
R	28		RES,CHIP,CERM,10M,+-5%,0.125W,1206	783274	89536	783274	1		
R	29		RES,CHIP,CERM,200K,+-5%,0.125W,1206	746743	89536	746743	1		
R	32		RES,CHIP,CERM,510,+-5%,0.125W,1206	746388	89536	746388	1		
R	33		RES,CHIP,CERM,51.1,+-1%,0.125W,1206	806422	89536	806422	1		
R	35, 37		RES,CHIP,CERM,82.5,+-1%,0.125W,1206	772764	89536	772764	2	1	
R	36, 38		RES,CHIP,CERM,130,+-5%,0.125W,1206	816041	89536	816041	2		
R	39		RES,CHIP,CERM,4.7K,+-5%,0.125W,1206	740522	89536	740522	1		
TP	1- 4		TERM,UNINSUL,WIRE FORM,TEST POINT	781237	89536	781237	4		
U	4	*	IC,ALSTTL,DUAL J-K F/F,+EDG TRIG,SOIC	808113	89536	808113	1	1	
U	5, 18	*	IC,ALSTTL,QUAD 2 INPUT OR GATE,SOIC	742460	89536	742460	2	1	
U	6, 9	*	IC,ALSTTL,QUAD 2 INPUT NAND GATE,SOIC	782268	89536	782268	2		
U	7, 8, 14,	*	IC,FTTL,QUAD BUS BUFFER,SOIC	801324	89536	801324	4	1	
U	40	*		801324					
U	10, 12	*	IC,FTTL,TRIPLE 3 INPUT NAND GATE,SOIC	743435	89536	743435	2	1	
U	11	*	IC,FTTL,8 INPUT NAND GATE,SOIC	811703	89536	811703	1	1	
U	13, 19	*	IC,ALSTTL,QUAD 2 INPUT AND GATE,SOIC	741827	89536	741827	2	1	
U	15	*	IC,ALSTTL,QUAD 2 INPUT NOR GATE,SOIC	782284	89536	782284	1	1	
U	16, 20, 28	*	IC,ALSTTL,OCTAL LINE DRVR,SOIC	741744	89536	741744	3	1	
U	17, 22, 26,	*	IC,ALSTTL,OCTAL D F/F,+EDG TRG,SOIC	799569	89536	799569	4	1	
U	36	*		799569					
U	21, 33	*	IC,ALSTTL,8-BIT IDENTITY COMP,SOIC	799585	89536	799585	2	2	
U	23, 34	*	IC,CMOS,OCTAL D F/F,+EDG TRG,SOIC	799544	89536	799544	2	1	
U	24, 27, 29,	*	IC,CMOS,OCTAL D F/F,+EDG TRG,SOIC	817247	89536	817247	9	1	
U	31, 32, 35,	*		817247					
U	37- 39	*		817247					
U	25, 30	*	IC,ALSTTL,OCTAL BUS TRANSCEIVER,SOIC	799593	89536	799593	2	1	
U	41	*	IC,ALSTTL,DUAL D F/F,+EDG TRG,SOIC	742452	89536	742452	1	1	
U	42	*	IC,ECL,QUAD ECL-TTL TRANSLATOR	801274	89536	801274	1		
U	43, 45	*	IC,COMPARATOR,QUAD,14 PIN,SOIC	741561	89536	741561	2	1	
U	44	*	IC,LSTTL,HEX SCHMT-TRIG INVERTER,SOIC	742437	89536	742437	1	1	
U	46	*	IC,CMOS,HEX INVERTER,UNBUFFERED,SOIC	806893	89536	806893	1	1	
U	47	*	IC,2.5V,40 PPM T.C.,BANDGAP REF,SOIC	807040	89536	807040	1	1	
XU	1- 3		SOCKET,IC,24 PIN	807941	89536	807941	3	1	
Y	1	*	CRYSTAL,32.768KHZ,+/-0.003%	501817	89536	501817	1		

An * in 'S' column indicates a static-sensitive part.



9000A-80286-1672

Figure 7-2. Interface PCA

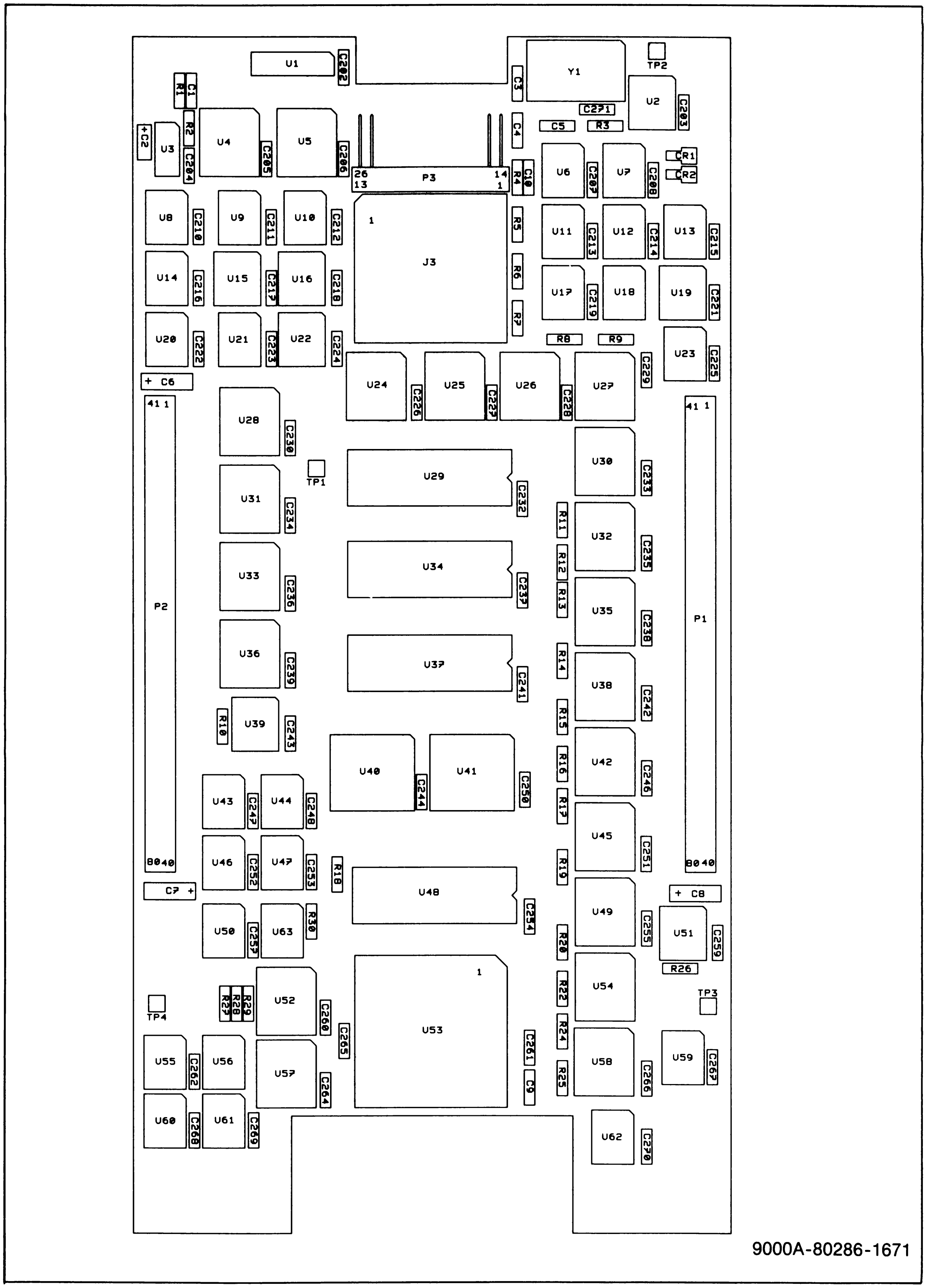
Table 7-3. Processor PCA
(See Figure 7-3.)

REFERENCE DESIGNATOR -A>-NUMERICS---->	S	--------------DESCRIPTION--------------	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	R S -Q	O T -E-
C 1		CAP,CER,0.1UF,+-10%,25V,X7R,1206	747287	89536	747287	1		
C 2		CAP,TA,0.47UF,+-20%,50V	807990	89536	807990	1		
C 4, 9		CAP,CER,0.047UF,+-20%,50V,X7R,1206	782615	89536	782615	2		
C 6- 8		CAP,TA,10UF,+-20%,25V	772491	89536	772491	3		
C 202-208,210-		CAP,CER,0.01UF,+-10%,50V,X7R,1206	747261	89536	747261	1	1	
C 202-208,210-			747261					
C 219,221-230,			747261					
C 232-239,241-			747261					
C 244,246-248,			747261					
C 250-255,257,			747261					
C 259-262,264,			747261					
C 271			747261					
CR 1, 2		DIODE,SI,BV=70.0V,IO=50MA,DUAL,SOT23	742320	89536	742320	2		
J 3		SOCKET RECEPTICAL ASSY	775981	89536	775981	1		
MP 2		BLOCK,SPACER 68 POS	773242	89536	773242	1		
MP 3		SPACER,SWAGED,RND,BRASS,4-40X0.340	380329	89536	380329	1		
MP 4		SPACER,SWAGED,RND,BRASS,4-40X0.437	442913	89536	442913	4		
P 1, 2		HEADER,2 ROW,0.100CTR,80 PIN	806273	89536	806273	2		
P 3		HEADER,2 ROW,0.100CTR,RT ANG,26 PIN	512590	89536	512590	1		
R 1, 3, 8,		RES,CHIP,CERM,10K,+-5%,0.125W,1206	746610	89536	746610	7		
R 10, 24- 26			746610					
R 2		RES,CHIP,CERM,200,+-5%,0.125W,1206	746339	89536	746339	1		
R 4, 7		RES,CHIP,CERM,732,+-1%,0.125W,1206	807677	89536	807677	2		
R 5, 6		RES,CHIP,CERM,1K,+-1%,0.125W,1206	783241	89536	783241	2		
R 9, 19, 20,		RES,CHIP,CERM,300,+-5%,0.125W,1206	746362	89536	746362	4		
R 22			746362					
R 11- 18		RES,CHIP,CERM,8.06K,+-1%,0.125W,1206	806356	89536	806356	8		
R 27, 29, 30		RES,CHIP,CERM,39,+-5%,0.125W,1206	746255	89536	746255	3		
R 28		RES,CHIP,CERM,0.05MAX,0.125W,1206	810747	89536	810747	1	1	
TP 1- 4		TERM,UNINSUL,WIRE FORM,TEST POINT	781237	89536	781237	4		
U 1	*	IC,CMOS,DELAY LINE,100 NSEC,5 TAP	807149	89536	807149	1		
U 2	*	IC,ALSTTL,DUAL J-K F/F,+EDG TRIG,SOIC	808113	89536	808113	1	1	
U 3	*	IC,VOLT SUPERVISOR,10V SENSE,SOIC	780502	89536	780502	1	1	
U 4, 24- 26	*	IC,ALSTTL,OCTAL D F/F,+EDG TRG,SOIC	799569	89536	799569	4	1	
U 5, 33	*	IC,ALSTTL,OCTAL LINE DRVR,SOIC	741744	89536	741744	2	1	
U 6	*	IC,ALSTTL,QUAD 2 INPUT NOR GATE,SOIC	782284	89536	782284	1	1	
U 7	*	IC,ALSTTL,HEX INVERTER,W/OC,SOIC	807248	89536	807248	1	1	
U 8, 18, 21	*	IC,ALSTTL,HEX INVERTERS,SOIC	782300	89536	782300	3	1	
U 9	*	IC,FTTL,QUAD DUAL AND GATE,SOIC	780957	89536	780957	1	1	
U 10, 59	*	IC,ALSTTL,TRIPLE 3INPUT NOR GATE,SOIC	782318	89536	782318	2	1	
U 11	*	IC,ALSTTL,QUAD 2 INPUT AND GATE,SOIC	741827	89536	741827	1	1	
U 12, 44, 46,	*	IC,ALSTTL,DUAL D F/F,+EDG TRG,SOIC	742452	89536	742452	5	1	
U 50, 62	*		742452					
U 13	*	IC,FTTL,DUAL D F/F,+EDG TRG,SOIC	742163	89536	742163	1	1	
U 14, 56	*	IC,FTTL,QUAD 2 INPUT NOR GATE,SOIC	807313	89536	807313	2	1	
U 15	*	IC,ALSTTL,4-BIT BINARY CNTR,SOIC	799536	89536	799536	1	1	
U 16	*	IC,FTTL,SYNC DIV BY 16 BIN CNTR,SOIC	811711	89536	811711	1	1	
U 17, 43	*	IC,ALSTTL,QUAD 2 INPUT OR GATE,SOIC	742460	89536	742460	2	1	
U 19	*	IC,ALSTTL,DUAL 2-4 LINE DECODER,SOIC	801332	89536	801332	1	1	
U 20, 47	*	IC,ALSTTL,QUAD 2 INPUT NAND GATE,SOIC	782268	89536	782268	2		
U 22	*	IC,ALSTTL,8-1 LINE MUX,SOIC	801340	89536	801340	1	1	
U 23	*	IC,ALSTTL,TRIPLE 3 INPUT NAND,SOIC	741629	89536	741629	1	1	
U 27, 49, 54,	*	IC,ALSTTL,OCTL D TRNSPRNT LATCH,SOIC	801316	89536	801316	4	1	
U 58	*		801316					
U 28, 31, 36	*	IC,ALSTTL,OCTL D F/F,+EDG TRG,SOIC	741769	89536	741769	3	1	
U 30, 35, 45	*	IC,ALSTTL,8-BIT IDENTITY COMP,SOIC	799585	89536	799585	3	1	
U 32, 38, 42	*	IC,CMOS,OCTAL D F/F,+EDG TRG,SOIC	799544	89536	799544	3		
U 39, 51	*	IC,ALSTTL,3-8 LINE DCDR W/ENABLE,SOIC	741686	89536	741686	2	1	
U 40, 41	*	IC,CMOS,8K X 8 STATIC RAM,70NSEC,SOIC	801027	89536	801027	2	1	
U 52, 57	*	IC,ALSTTL,OCTAL BUS TRANSCEIVER,SOIC	799593	89536	799593	2	1	
U 55	*	IC,FTTL,TRIPLE 3 INPUT AND GATE,SOIC	808089	89536	808089	1	1	
U 60	*	IC,LSTTL,8BIT S-IN,P-OUT R-SHFT,SOIC	742106	89536	742106	1	1	
U 61	*	IC,FTTL,TRIPLE 3 INPUT NAND GATE,SOIC	743435	89536	743435	1	1	
U 63	*	IC,FTTL,QUAD 2 INPUT OR GATE,SOIC	743237	89536	743237	1	1	
XU 29, 34, 37,		SOCKET,IC,28 PIN	448217	91506	328-AG39D	4		
XU 48			448217					
XU 53		SOCKET,PIN GRID ARRAY,0.100CTR,68 PIN	715409	89536	715409	1		
Y 1		OSCILLATOR,25.0MHZ,TTL CLOCK	756346	89536	756346	1		

An * in 'S' column indicates a static-sensitive part.

NOTES:

R28 is a 0 ohm resistor.



9000A-80286-1671

Figure 7-3. Processor PCA

Table 7-4. Clock Buffer PCA
(See Figure 7-4.)

REFERENCE DESIGNATOR -A>-NUMERICS---->	S	DESCRIPTION	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	R S -Q	N O T -E-
C 1		CAP,CER,5.6PF,+-10%,50V,COG,1206	782409	89536	782409	1	1	
C 2- 4, 7		CAP,CER,0.01UF,+-10%,50V,X7R,1206	747261	89536	747261	4		
C 5, 6		CAP,CER,0.047UF,+-20%,50V,X7R,1206	782615	89536	782615	2		
CR 1		DIODE,SI,BV=70.0V,IO=50MA,DUAL,SOT23	742320	89536	742320	1		
R 1		RES,CHIP,CERM,100,+-5%,0.125W,1206	746297	89536	746297	1		
R 2, 3		RES,CHIP,CERM,22K,+-5%,0.125W,1206	746651	89536	746651	2		
R 4		RES,CHIP,CERM,47.5K,+-1%.0.125W,1206	812222	89536	812222	1		
R 5		RES,CHIP,CERM,14K,+-1%,0.125W,1206	816033	89536	816033	1		
U 1	*	IC,COMPARATOR,HI-SPEED,ECL OUTPUTS	812081	89536	812081	1		

An * in 'S' column indicates a static-sensitive part.

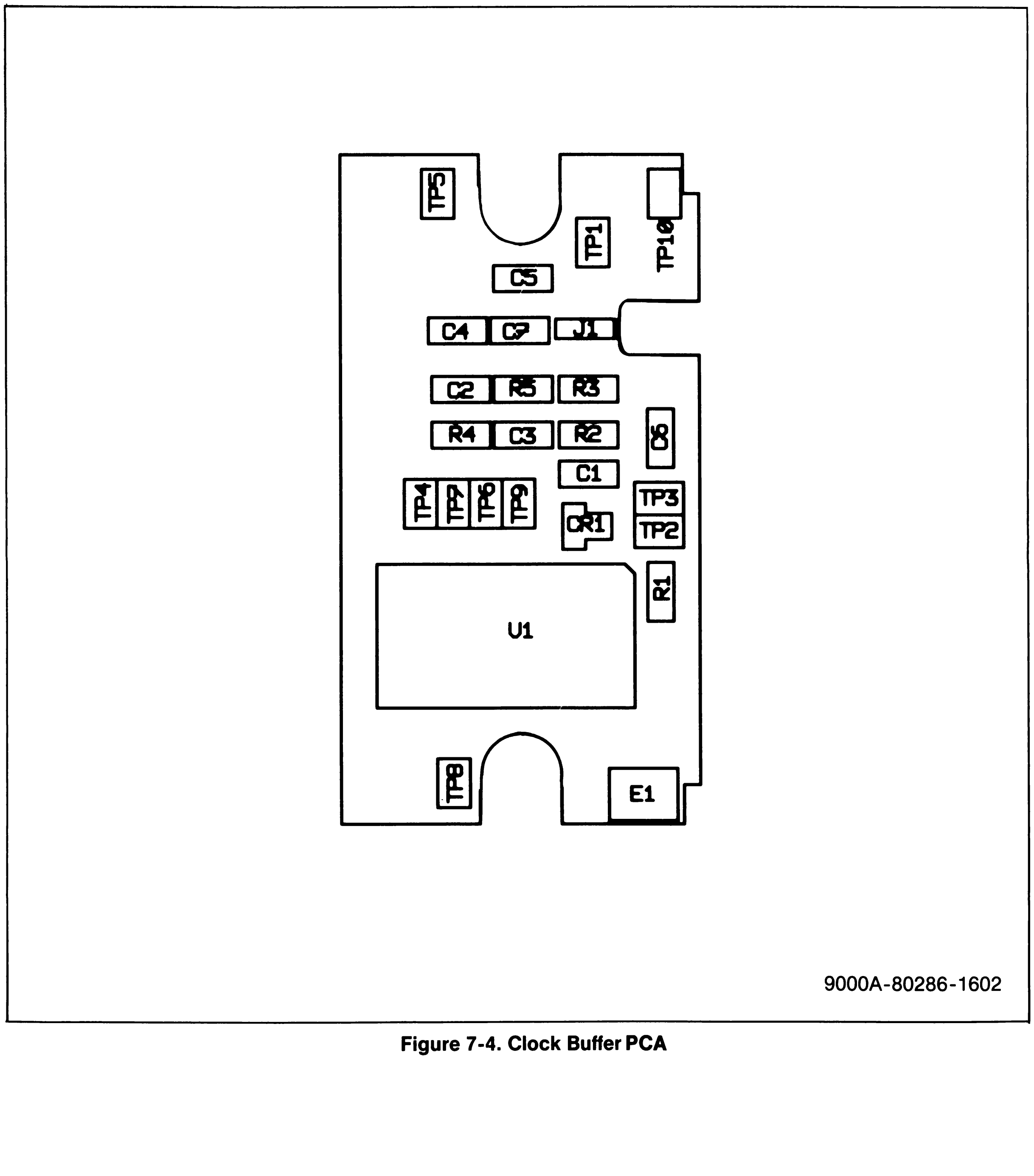


Figure 7-4. Clock Buffer PCA

Table 7-5. Manual Status Information

REF OR OPTION NO.	ASSEMBLY NAME	FLUKE PART NO.	X = The PCA revision levels documented in this manual.																					
			—	A	B	C	D	E	F	G	H	J	K	L	M	N	P							
	Processor PCA	818799	●	●	X																			
	Interface PCA	818807	●	●	X																			
	Clock Buffer PCA	818088	●	●	X																			

● = These revision letters were never used in the instrument.

Section 8
Schematic Diagrams

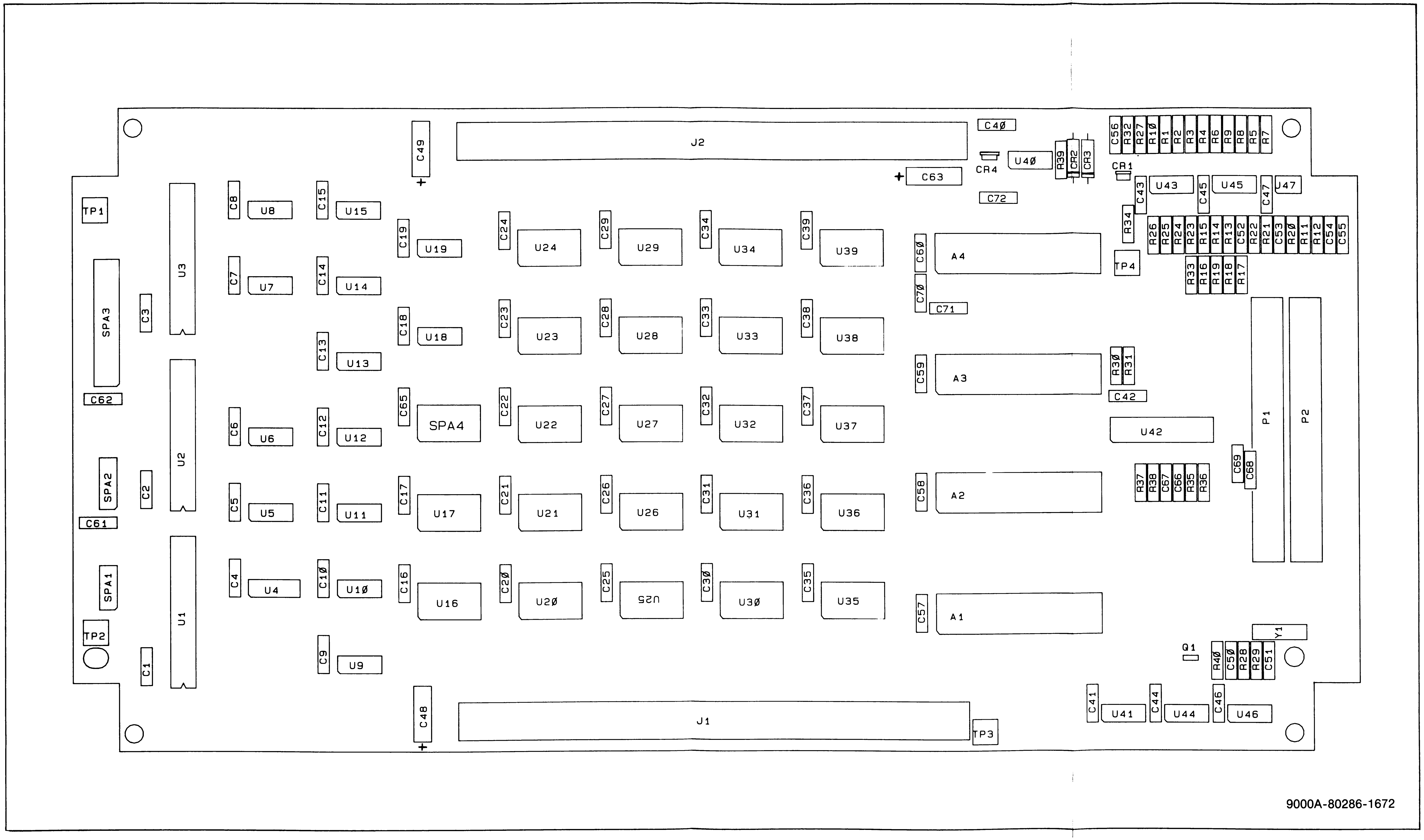
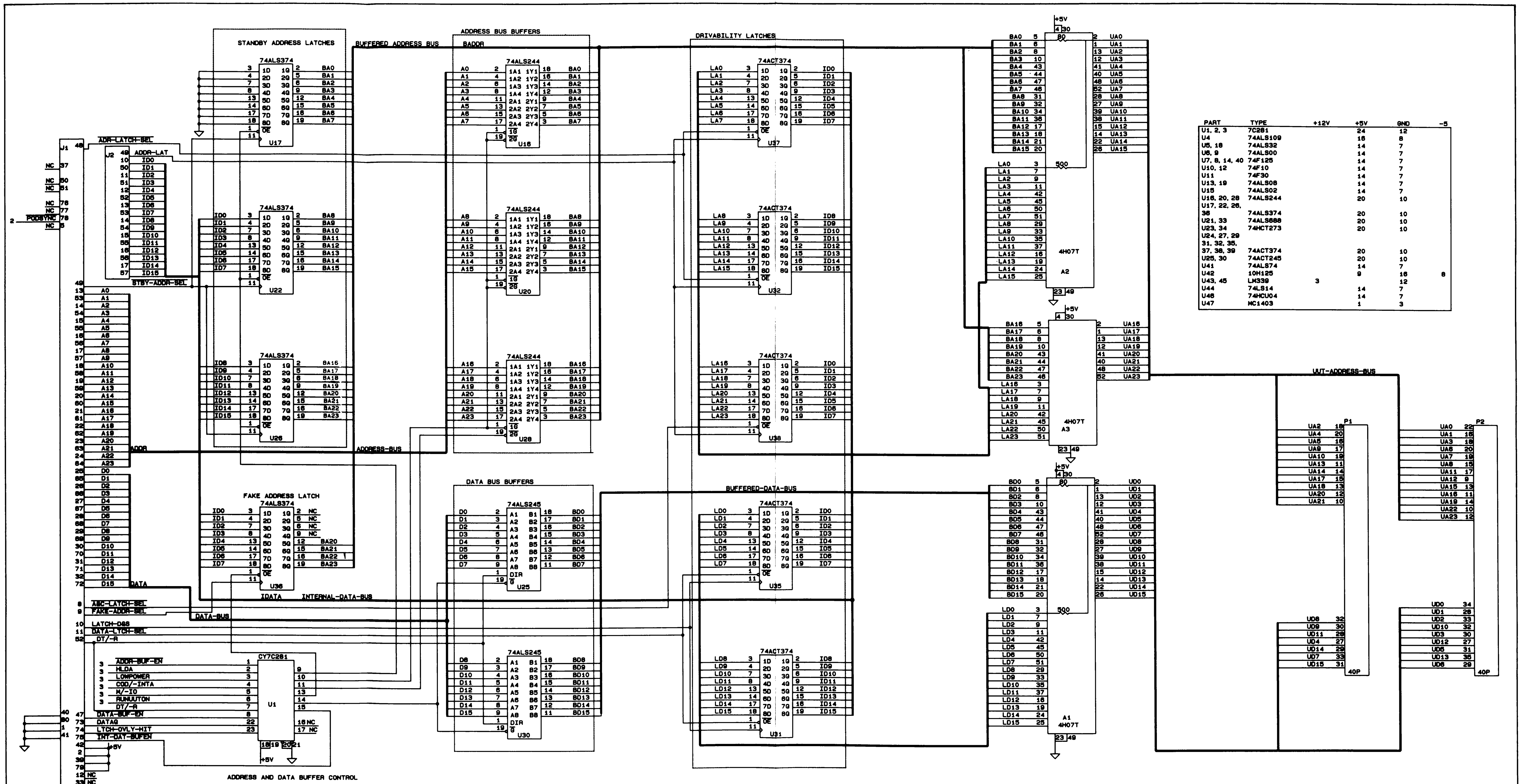


Figure 8-1. Interface PCA



PART	TYPE	+12V	+5V	GND	-5
U1, 2, 3	7C281		24	12	
U4	74ALS109		16	8	
U5, 18	74ALS32		14	7	
U6, 9	74ALS00		14	7	
U7, 8, 14, 40	74F125		14	7	
U10, 12	74F10		14	7	
U11	74F30		14	7	
U13, 19	74ALS08		14	7	
U15	74ALS02		14	7	
U16, 20, 28	74ALS244		20	10	
U17, 22, 26, 36	74ALS374		20	10	
U21, 33	74ALS688		20	10	
U23, 34	74HCT273		20	10	
U24, 27, 29 31, 32, 35, 37, 38, 39	74ACT374		20	10	
U25, 30	74ACT245		20	10	
U41	74ALS74		14	7	
U42	10H125		9	16	8
U43, 45	LM339	3		12	
U44	74LS14		14	7	
U46	74HCU04		14	7	
U47	MC1403		1	3	

NOTES: UNLESS OTHERWISE SPECIFIED.

1. WARNING: INDICATES THE USE OF MOS DEVICE(S) WHICH MAY BE DAMAGED BY STATIC DISCHARGE. USE SPECIAL HANDLING PER S.O.P. 19.1.

DES	LAST USED	NOT USED
J	J2	
U	U47	
TP	TP4	
CR	CR4	
Q	Q1	
R	R40	
C	C72	C64
P	P2	

CAUTION
SUBJECT TO DAMAGE BY STATIC ELECTRICITY

9000A-80286-1072

(1 of 3)

Figure 8-1. Interface PCA (cont.)

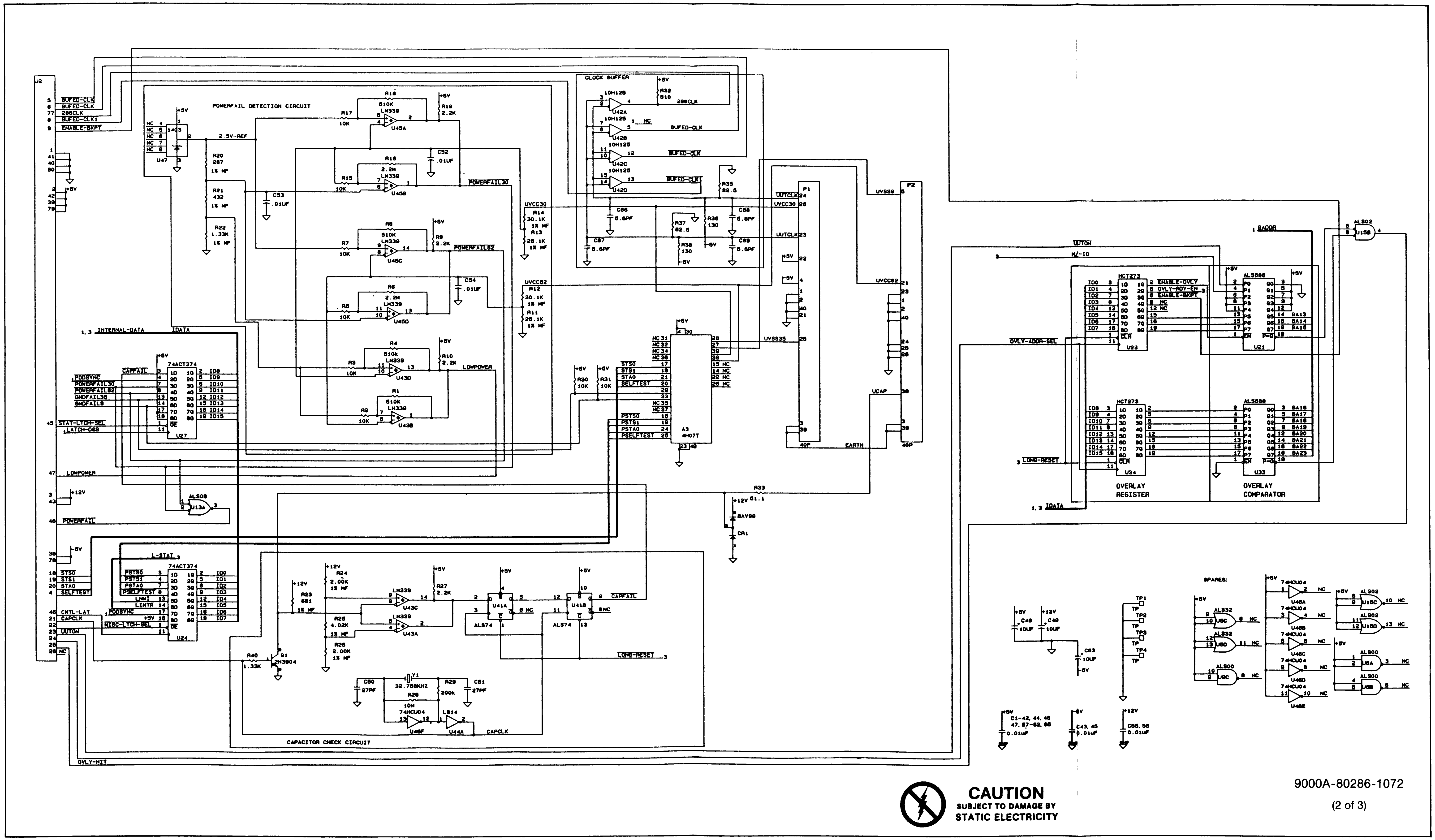


Figure 8-1. Interface PCA (cont.)

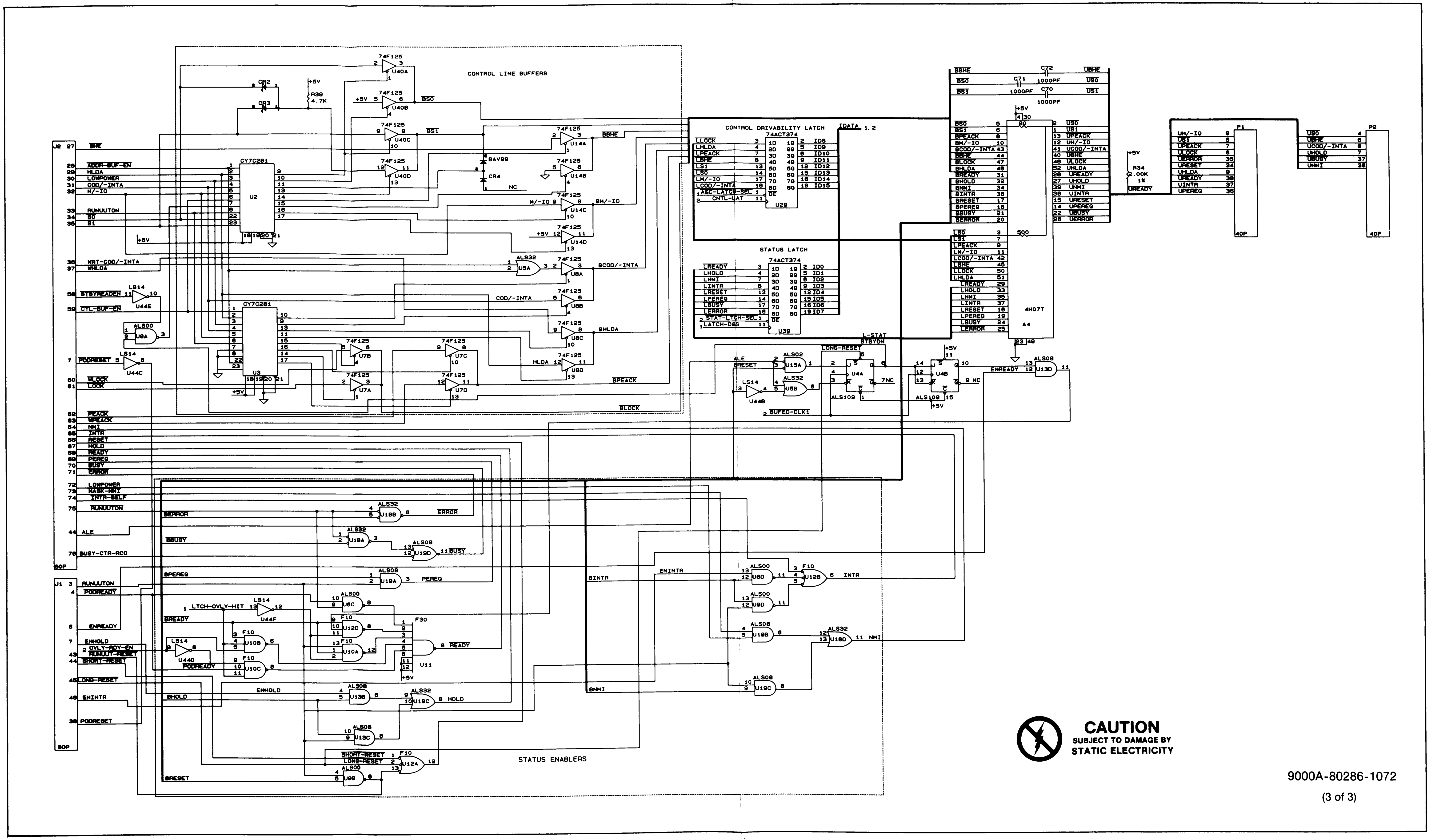


Figure 8-1. Interface PCA (cont.)

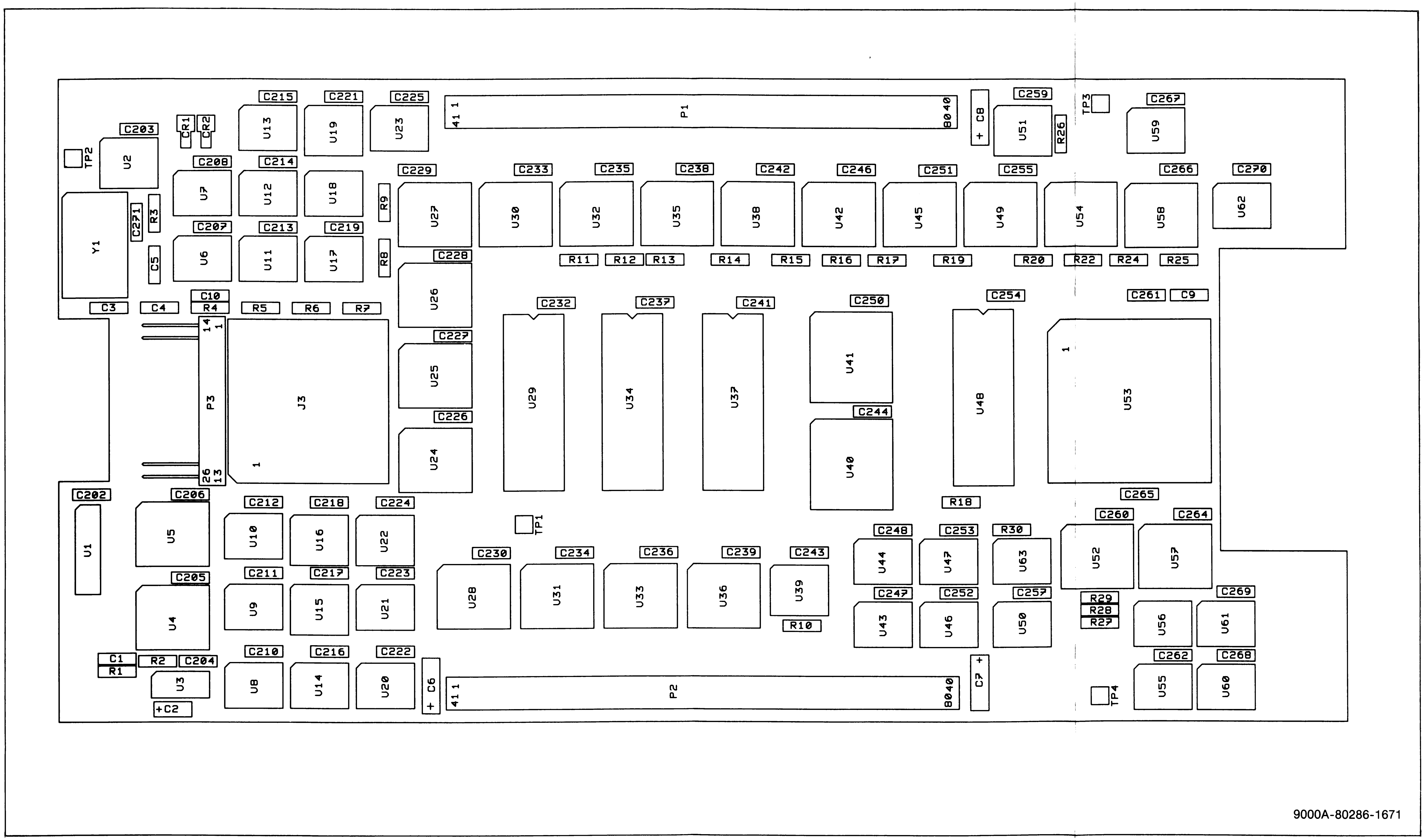
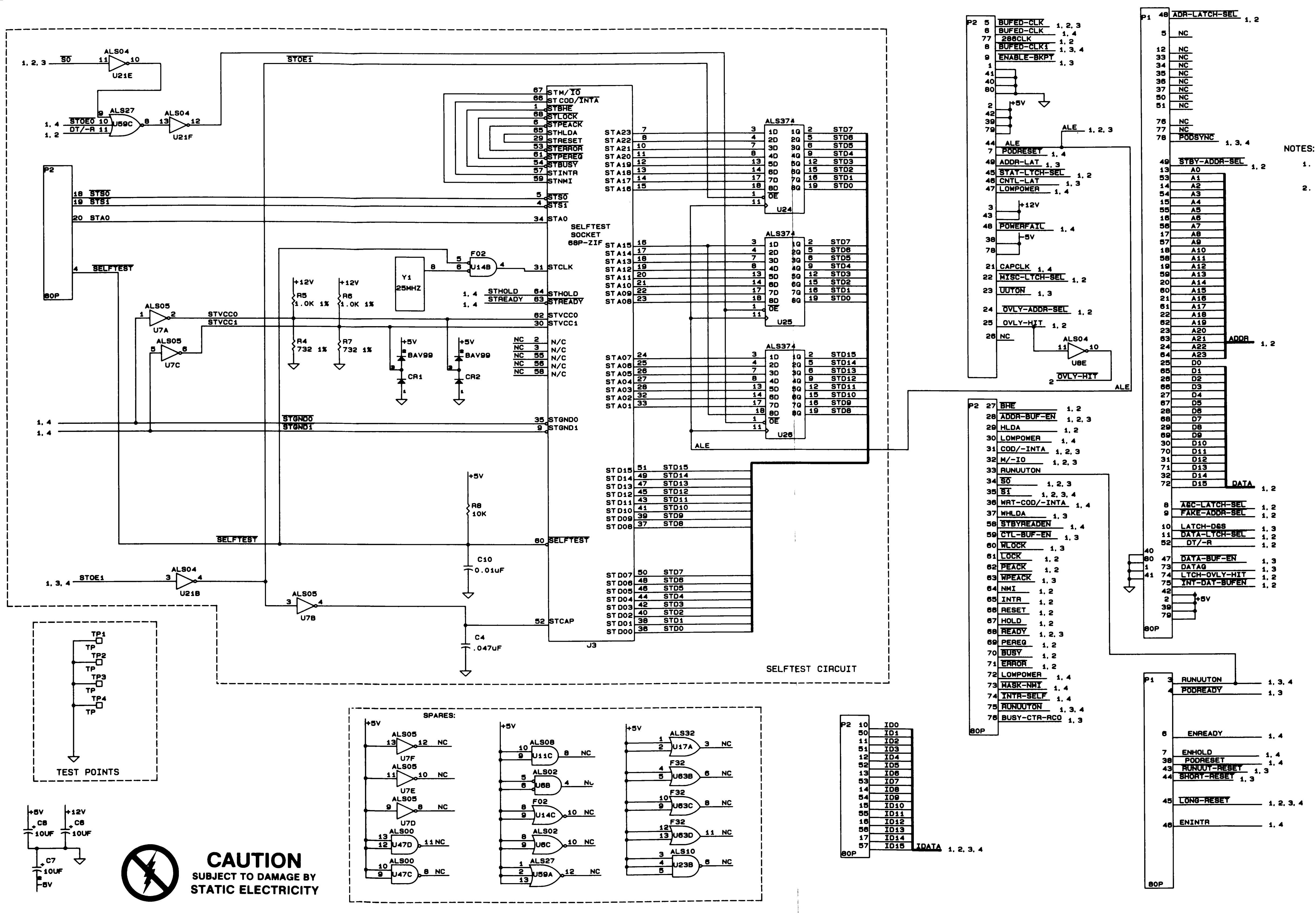


Figure 8-2. Processor PCA



NOTES: UNLESS OTHERWISE SPECIFIED.

1. ALL RESISTOR VALUES ARE IN OHMS, 1/8W, 5%. ALL CAPACITOR VALUES ARE IN MICROFARADS.

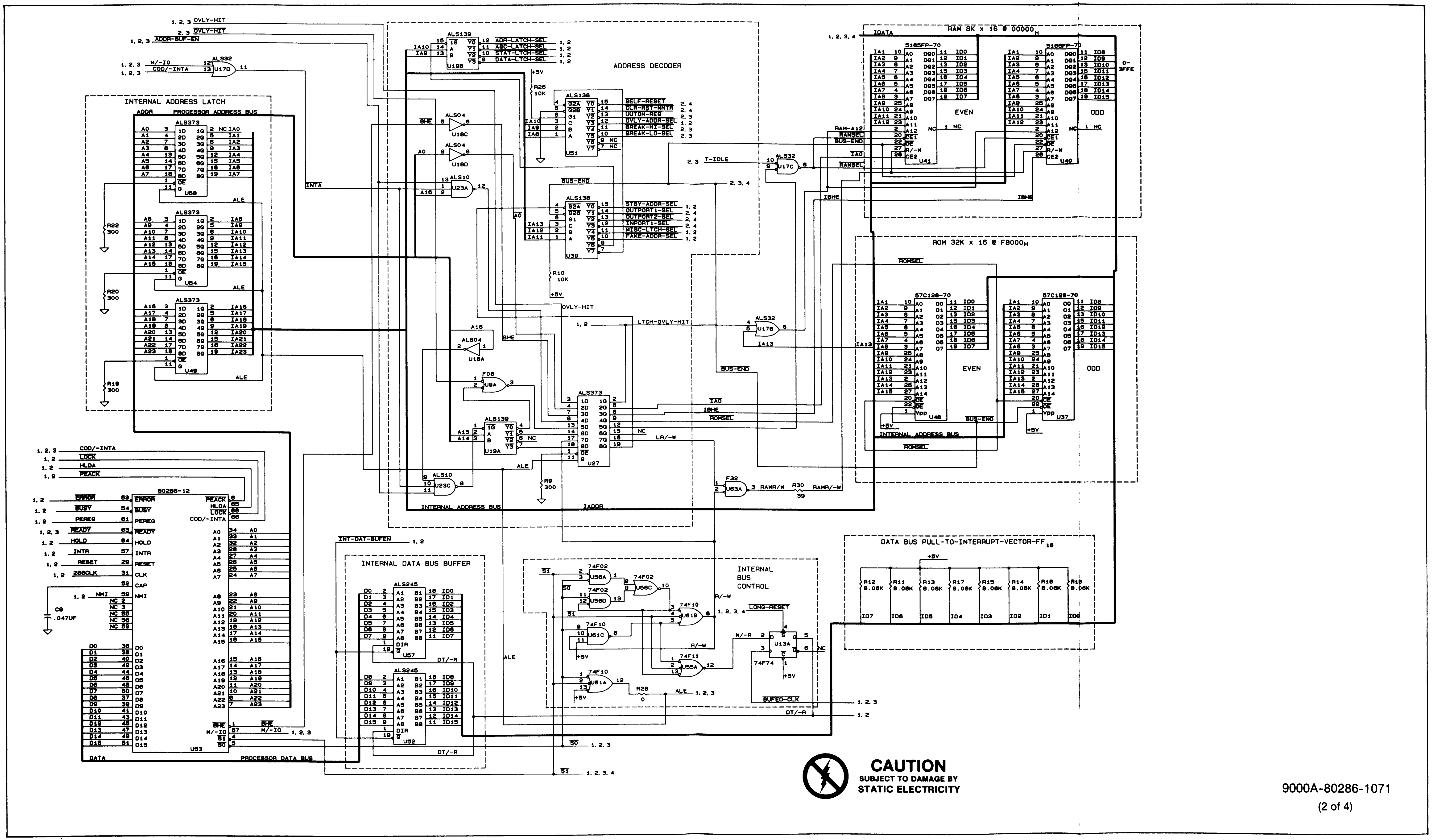
2. WARNING: ⊠ INDICATES THE USE OF MOS DEVICE(S) WHICH MAY BE DAMAGED BY STATIC DISCHARGE. USE SPECIAL HANDLING PER S.O.P. 19.1.

PART	TYPE	+5V	GND
U20, 47	74ALS00	14	7
U6	74ALS02	14	7
U14, 56	74F02	14	7
U8, 18, 21	74ALS04	14	7
U7	74ALS05	14	7
U11	74ALS08	14	7
U9	74F08	14	7
U23	74ALS10	14	7
U61	74F10	14	7
U55	74F11	14	7
U10, 59	74ALS27	14	7
U17, 43	74ALS32	14	7
U12, 44, 46, 50, 62	74ALS74	14	7
U13	74F74	14	7
U2	74ALS109	16	8
U39, 51	74ALS138	16	8
U19	74ALS139	16	8
U22	74ALS151	16	8
U15	74ALS161	16	8
U16	74F161A	16	8
U60	74LS164	14	7
U5, 33	74ALS244	20	10
U52, 57	74ALS245	20	10
U28, 31, 36	74ALS273	20	10
U32, 38, 42	74HCT273	20	10
U27, 49, 54, 58	74ALS373	20	10
U4, 24, 25, 26	74ALS374	20	10
U30, 35, 45	74ALS688	20	10
U3	7705	8	4
U40, 41	5165-70	28	14
U1	DS1000-100	14	7
Y1	25MHZ OSC	14	7
U34, 29	82S105B	28	14
U37, 48	57C128	28	14
U63	74F32	14	7

DES	LAST USED	NOT USED
J	J3	J1, J2
U	U63	
TP	TP4	
CR	CR2	
Q		
R	R30	R21
C	C271	C11-201, 209, 220, 231, 240, 245, 249, 256, 258, 263
P	P3	

9000A-80286-1071
(1 of 4)

Figure 8-2. Processor PCA (cont.)



9000A-80286-1071
(2 of 4)

Figure 8-2. Processor PCA (cont.)

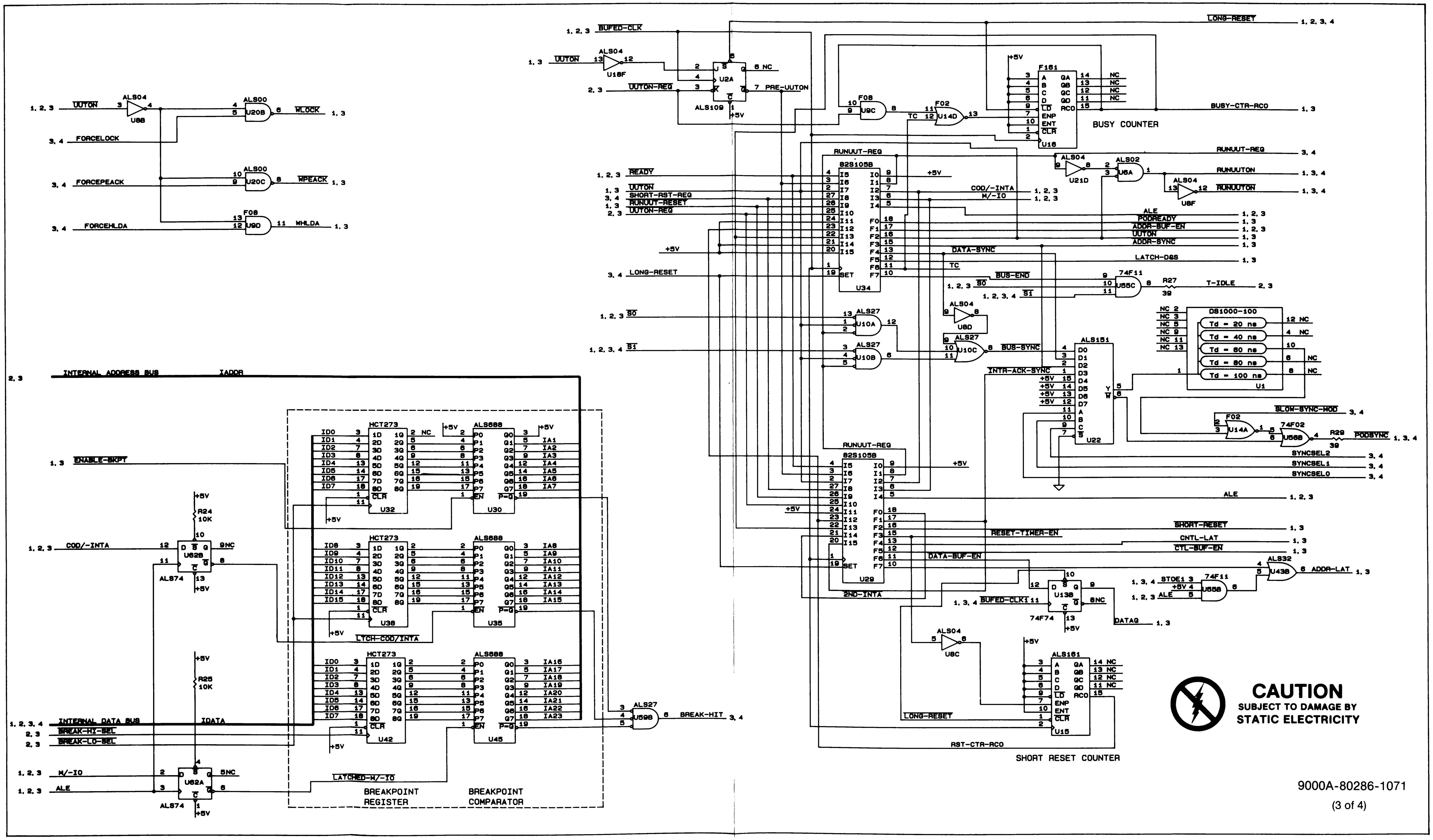


Figure 8-2. Processor PCA (cont.)

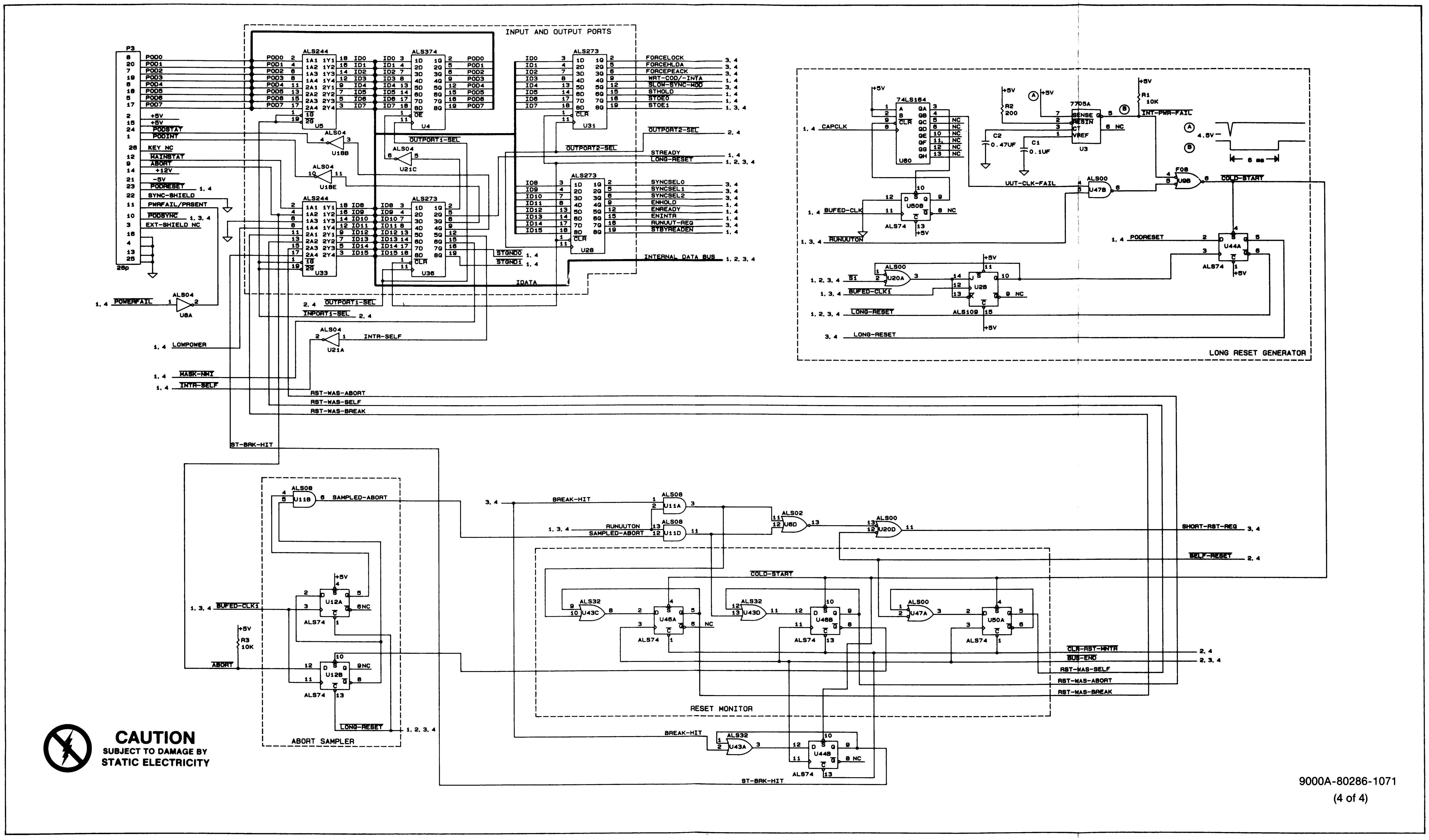
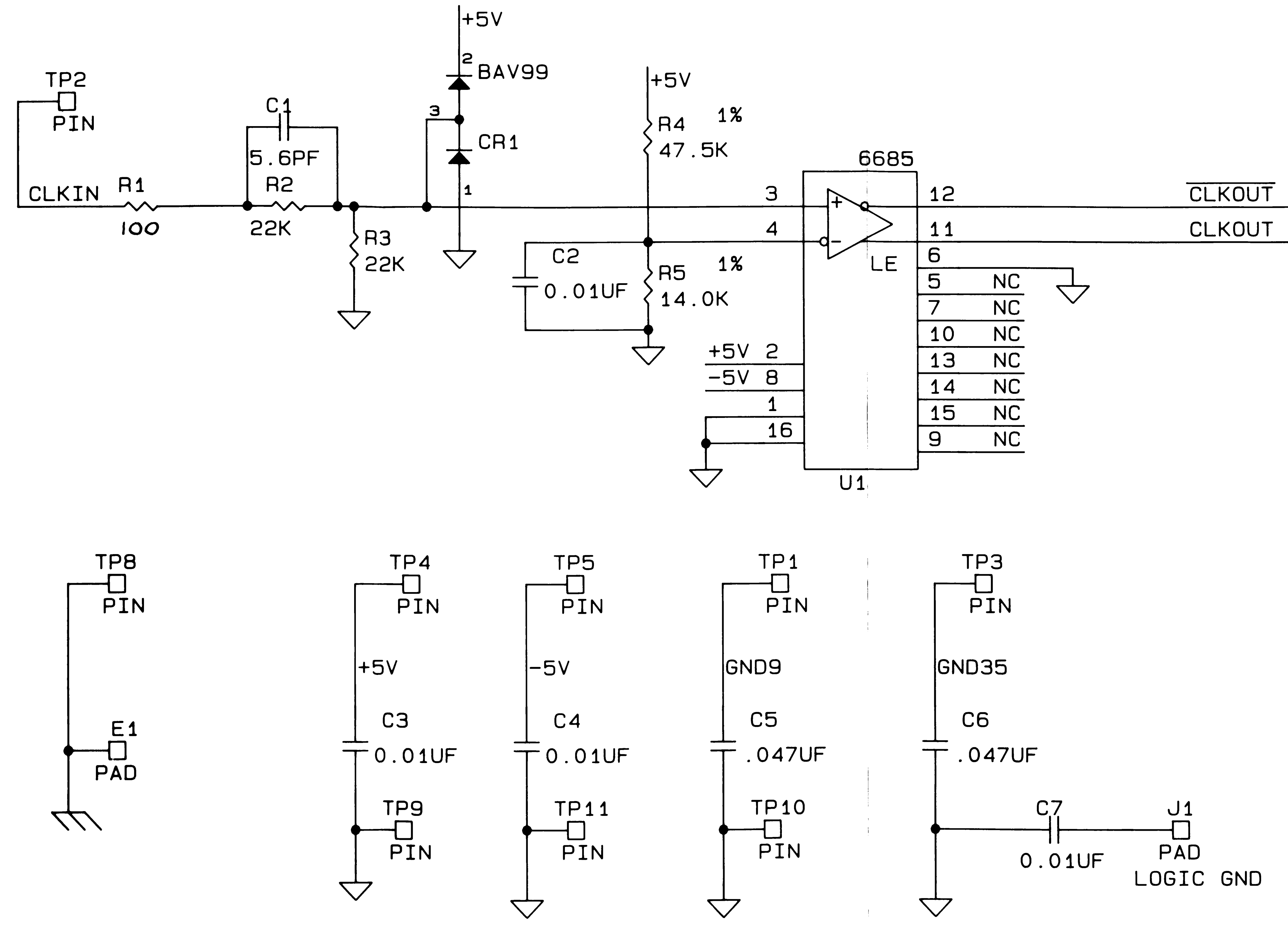
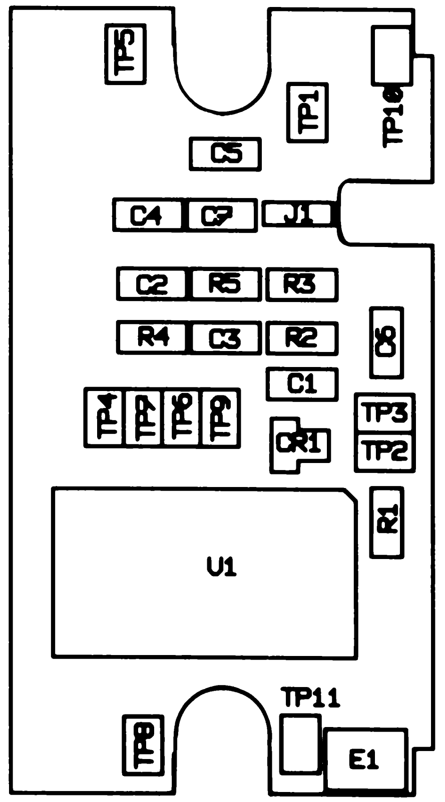


Figure 8-2. Processor PCA (cont.)





NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL RESISTOR VALUES ARE IN OHMS, 1/8 WATT, 5%.

2. ALL CAPACITORS VALUES ARE IN MICROFARADS.

3. **WARNING:** ⊘ INDICATES USAGE OF MOS DEVICE(S) WHICH MAY BE DAMAGED BY STATIC DISCHARGE. USE SPECIAL HANDLING PER S.O.P. 19.1

CAUTION SUBJECT TO DAMAGE BY STATIC ELECTRICITY

9000A-80286-1002

9000A-80286-1672

Figure 8-3. Clock Buffer

Appendix A
Creating Data Files for the 9000-Series Mainframe

USING THE 9010A LANGUAGE COMPILER PROGRAM — A-1.

The 9010A Language Compiler is a program that creates test programs for the 9000-Series Mainframe by converting source files which are created and edited on a microcomputer into programs for the Mainframe. The Compiler is available for several common microcomputers, including the Fluke 1720A and 1722A Instrument Controllers, computers with the CP/M operating system, and the IBM Personal Computer. Contact a Fluke Sales Office for information about the 9010A Language Compiler.

Before using the compiler to create 9010A programs for use with the 80286 Pod, you need to first create a new pod data file. The following procedure describes how to create the Pod data file, and how to add it to the list of files that the VERIFY program checks.

CREATING A NEW POD DATA FILE — A-2.

A pod data file is simply an ASCII file that you you create with the text editor on your host computer system. The following steps describe the creation of a new Pod data file:

1. Using the editor, create a new file named 80286.POD.

2. Copy the following lines into the file:

```
!
! 80286 Pod data file
!

FORCELN READY = 0
FORCELN HOLD = 1
busadr = 0000
uutadr = FFFFF0
```

3. Save this new file as file 80286.POD on the disk.

The new Pod data file can now be used with the compiler as described in the 9010A Language Compiler manual.

VERIFYING THE POD DATA FILE A-3.

The VERIFY program supplied with the 9010A Language Compiler on the 9LC disk verifies the integrity of files, and detects files that have been corrupted over time. VERIFY does this by calculating a checksum for each file, and comparing that checksum to the checksum contained in the VERIFY.DAT file.

If you would like to add your new Pod data file to the list of files which are checked by the VERIFY program, perform the following steps:

1. Edit file VERIFY.DAT supplied on the 9LC disk and add the following line at the end of the file:

   ```
   80286.POD        DDDD
   ```

   80286.POD is the name of the new Pod data file and DDDD is a dummy checksum for the file. (You will replace the dummy checksum with a real one later.)

2. Save the modified VERIFY.DAT file on the disk.

3. Run the VERIFY program. The last two messages that it reports should be:

   ```
   File 80286.POD error - signature is yyyy, should be DDDD

   zz files tested - 1 bad signatures, 0 missing files
   ```

   80286.POD is the name of the new Pod data file, yyyy is the correct checksum for the Pod data file, and zz is the number of files tested.

4. Write down the correct checksum for the pod data file (yyyy).

5. Re-edit the file VERIFY.DAT and replace the dummy checksum you entered before (DDDD) with the correct checksum (yyyy) as reported by the verify program.

6. Run the VERIFY program again to confirm that all changes have been made satisfactorily. The last two messages VERIFY reports should now be:

   ```
   File 80286.POD verified

   zz files tested — no errors
   ```

Appendix B
Using the Pod with a Remote 9020A

INTRODUCTION B-1.

The 80286H Pod may be used with a 9020A Micro-System Troubleshooter in its remote IEEE-488- or RS-232-controlled mode. The only Pod-specific information that you need to know to use the 80286H Pod with a remotely operated 9020A is the commands for controlling the Enableable Status Lines. (Using Status Lines in the Local Mode is described in Section 4 of this manual.)

REMOTE SETUP OF ENABLE LINES B-2.

The 9020A reserves eight Setup commands (S0,8 to S0,15) to accommodate up to eight possible UUT μP enable lines. The two Enableable Status lines (Enable Lines) of the 80286H Pod may be changed (as in the Local Mode Setup Command) by sending setup commands via the remote bus. Paragraph 6-31 in the 9020A Operator Manual gives instructions for sending these commands. The commands for the 80286H Pod are shown in Table B-1 below, which is an addendum to Table 6-9, Enable Line Setup Commands, in the 9020A Operator Manual.

Table B-1. Enable Line Setup Commands

INTERFACE POD	S0, 8	S0, 9	S0, 10	S0, 11	S0, 12	S0, 13	S0, 14	S0, 15
80286H	$\overline{\text{READY}}$	HOLD	*	*	*	*	*	*
*If this command is sent to the 9020A, it will cause a command error (status response 95).								

Appendix C
Testing UUTs With Soldered-in Microprocessors

INTRODUCTION C-1.

The 80286H Pod is normally used with UUTs that have their microprocessors installed in sockets. The microprocessor is removed from the UUT socket and replaced by the UUT plug of the Pod. Some processors, however, are soldered directly on the UUT. Soldered-in microprocessors could cause special problems when testing with the Pod. Certain testing considerations should be taken into account during UUT design.

This Appendix shows various methods of designing the UUT to simplify testing with the Pod. A description of a special PLCC (Plastic Leaded Chip Carrier) Adapter that is available from Fluke for testing soldered-in 80286 microprocessors is also included.

DESIGNING THE UUT FOR TESTING C-2.

The Pod may be connected to the UUT with an alternate test connector, or in the case of a surface-mounted PLCC, with a clip-on adapter. The connection to the UUT must be designed in the board so that all needed features are available.

To test a UUT with a soldered-in microprocessor, it is necessary to tri-state the processor in the board by asserting HOLD. If HOLD/HLDA is used as an alternate bus master, DMA device, or RAM refresh, the Pod must monitor these lines, yet the soldered-in microprocessor lines must be isolated from the rest of the UUT circuit and the Pod. Figure C-1 shows several methods for isolating HOLD and HLDA from the UUT circuit during testing.

For further information on designing for testability, see Fluke application note F0032A.

THE FLUKE CLIP-ON ADAPTER C-3.

The Fluke Y9000A-80286-7202 Clip-on Adapter (or PLCC Adapter) allows you to connect the Pod to 80286-based UUTs with soldered-in PLCC microprocessors. All processor lines are tied from the Pod to the UUT except HOLD, HLDA, and CAP. From the Pod, HOLD and HLDA are brought to clip-on leads. From the UUT side, HOLD is pulled high with a 1k ohm resistor; HLDA is not connected. This allows the Pod to tri-state the soldered-in processor, and if the design of the UUT provides for test points to connect to the isolated HOLD and HLDA lines, all UUT functions can be tested. For the clip-on adapter to work correctly, the UUT must allow the adapter to pull the soldered-in processor's HOLD input high. If the HOLD/HLDA function is not used on the UUT, disable the Clip-on Adapter's HOLD input by connecting the HOLD clip-on lead to logic ground or by using the Mainframe SETUP command.

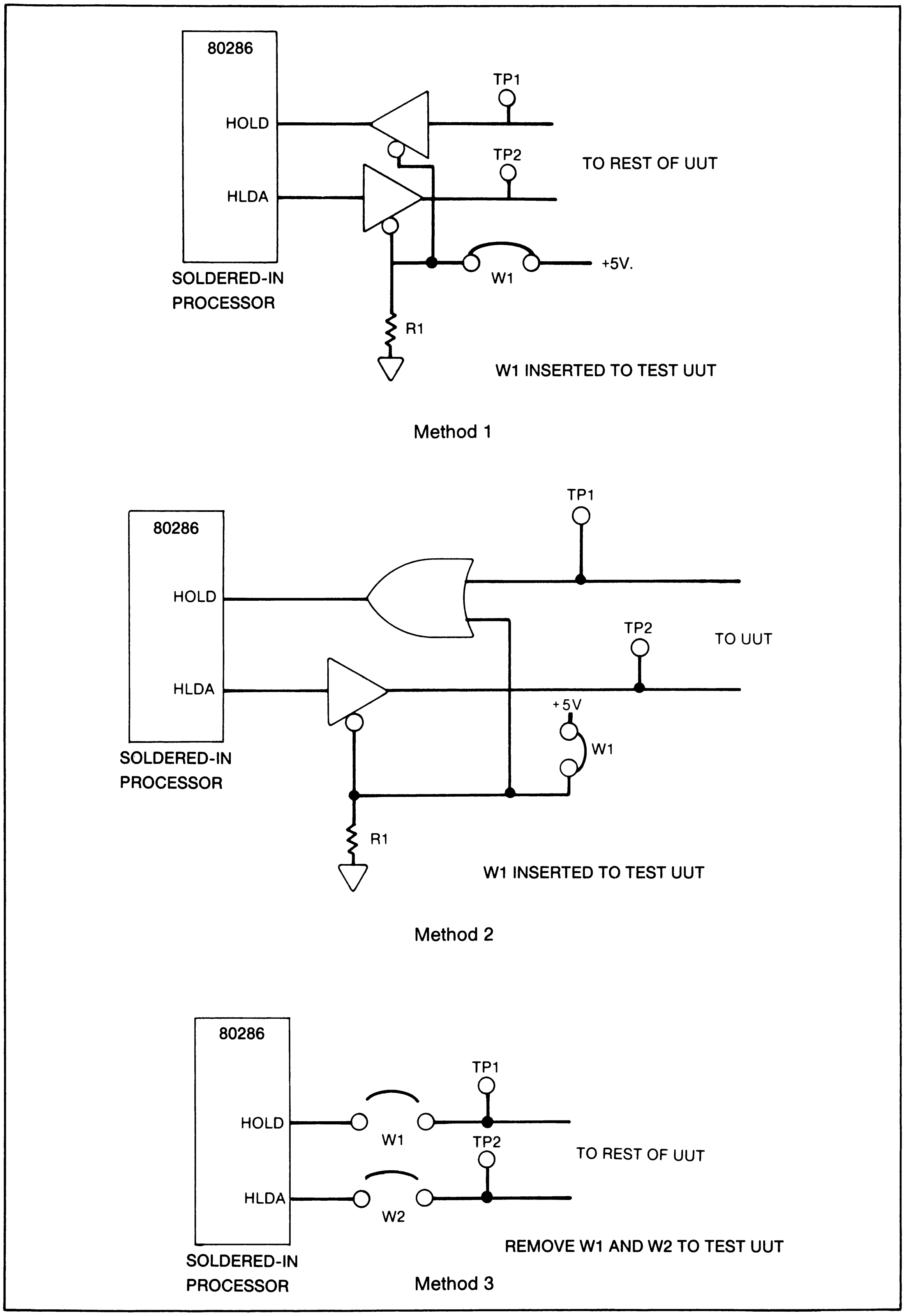


Figure C-1. UUT Isolation of HOLD and HLDA

The Clip-on Adapter does not perform the normal Pod CAP test. With the Clip-on Adapter, the CAP test always passes.

INSTALLATION OF THE PLCC ADAPTER C-4.

Section 1-8 in this manual, Connecting the Pod to the UUT, describes how to attach the Pod plug to the UUT's microprocessor socket. To test UUTs with soldered-in microprocessors, a special PLCC clip-on adapter is attached to the Pod plug and clipped over the UUT processor. When using this clip-on adapter, step 5 of that procedure should be amended as follows:

5. Connect the Pod plug to the UUT using the clip-on adapter:

   a. Insert the pins of the Pod plug into the socket on the adapter (as shown in Figure C-2). Make sure that pin 1 of the Pod plug is aligned with pin 1 of the adapter.

   b. Open the adapter clip and attach it to the soldered-in microprocessor. Make sure that pin 1 of the adapter is aligned with pin 1 of the microprocessor.

   c. Attach the clip-on leads to the HOLD and HLDA lines. If HOLD is not used on the UUT, skip this step.

Note that the adapter must be removed before conducting another Pod self test, or an adapter must be devised for the self-test socket.

TROUBLESHOOTING A CLIP-ON ADAPTER C-5.

If the Pod passes the self test, but does not appear to work properly when connected to a UUT using the clip-on adapter, begin troubleshooting the Pod by carefully checking the adapter:

1. Check the adapter for continuity from the clip-on connector contacts back to the Pod plug socket contacts.

2. Check for shorts between adjacent signal lines throughout the adapter.

3. Use an oscilloscope to check that the clock signal to the Pod is present.

If these procedures do not yield a solution, contact Fluke Customer Service for advice.

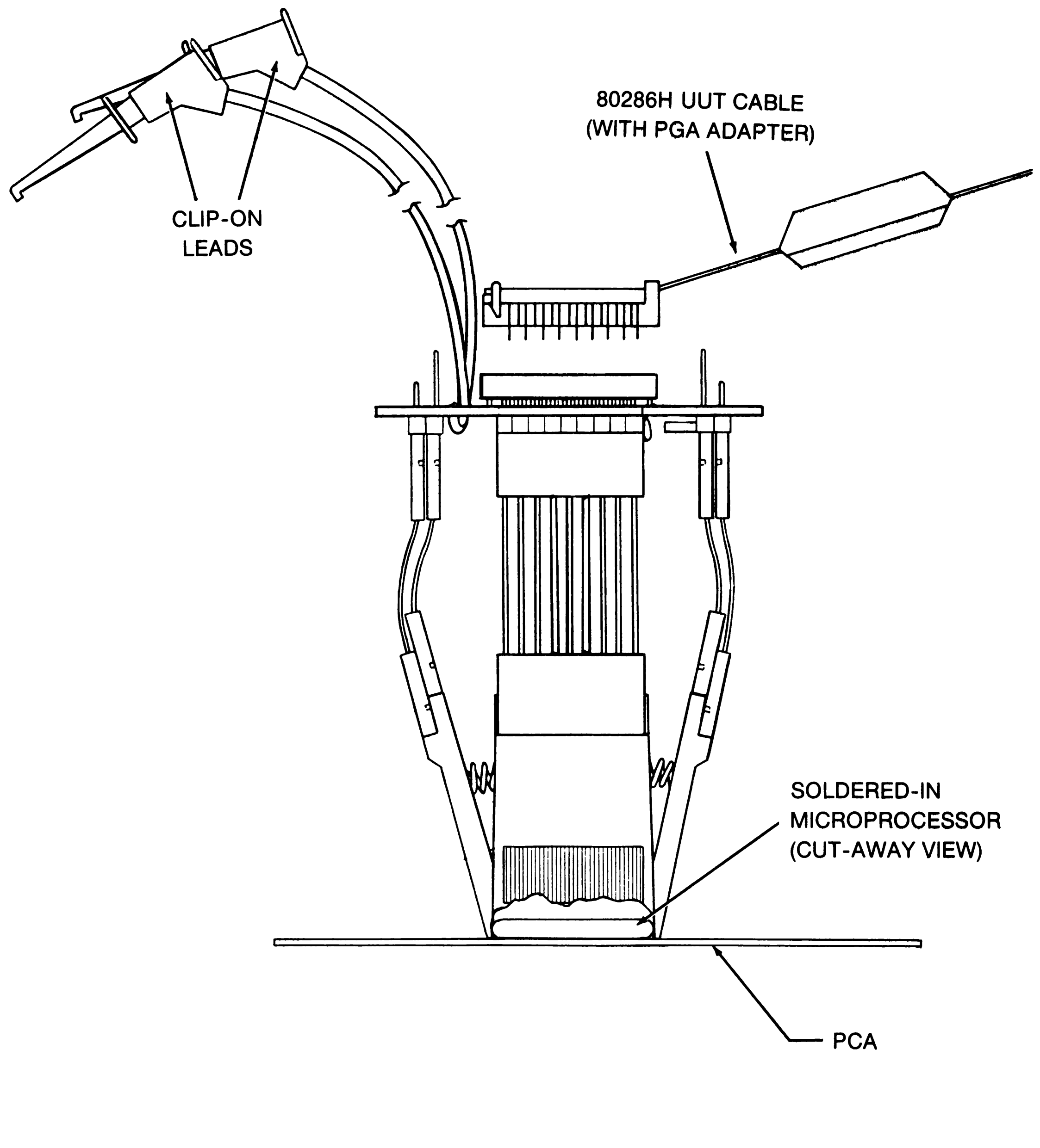


Figure C-2. Connecting the Pod Plug to the PLCC Adapter

Appendix D
Sockets and Adapters

RECOMMENDED LCC SOCKETS D-1.

Table D-1 contains a list of recommended LCC (Leadless Chip Carrier) sockets for the 80286H Pod. These sockets have been tested by Fluke and have been shown to fit the plug end of the Pod UUT cable. Due to the hold-down mechanisms used by some manufacturers, the Pod connector may not fit in sockets made by other manufacturers. This table lists the manufacturer, part number, and any caution that should be used when installing the Pod UUT cable in the socket.

These LCC sockets are recommended because they reduce the likelihood of UUT cable damage through contact with sharp edges on the socket.

PLCC SOCKET ADAPTERS D-2.

You can adapt the Pod UUT cable plug to a UUT that has a socketed PLCC microprocessor with an Emulation Technology BC4-68-PGA1-80286 adapter.

Table D-1. Recommended LCC Sockets

MANUFACTURER	PART NUMBER	COMMENTS
3M/Textool	268-5400-50-1102	Socket used in this Pod. (All other LCC sockets manufactured by 3M fit with no problems reported.)
Kel-Am		Fits with some maneuvering through the cover.
Methode	212-068-00X	Fits using Methode cover p/n 212-10 only. Tangs should be insulated.

Appendix E
Using the 80286H Pod from TL/1 Programs

READING THE DATABASE VERSION NUMBER E-1.

The 80286H Pod database is contained in a disk file on the 9100-Series Mainframe. The disk file contains information that determines the Pod's interface to the rest of the system. To read the version number of the database from the front panel, press SETUP, then the right arrow key (→), SOFT KEYS, the POD_NAME softkey, and the ENTER key.

READING THE POD SOFTWARE VERSION NUMBER E-2.

To determine the Pod software version, you perform a read at a special address. From the front panel, press the READ key, the SPECIAL softkey, and the right arrow key (→). Then enter address F000 0012 and press ENTER.

TL/1 PROGRAMMING APPLICATIONS E-3.

Pod Address Space Options E-4.

The following Pod address space options are available:

SPACE	SIZE
MEMORY	WORD
MEMORY	BYTE
I/O	WORD
I/O	BYTE
INSTRUCT	(Not used-Instruction access is word only.)

The following TL/1 program segment demonstrates how to change the Pod space:

```
program example1
        s = getspace space "I/O", size "WORD"
        setspace space s
end example1
```

Pod-Specific Setup Information E-5.

Pod-specific setup information is listed in Table E-1, along with data values, defaults, and ranges.

The following program demonstrates syntax use in TL/1 commands for the "podsetup" statement.

NOTE

TL/1 hexadecimal data requires a "$" prefix character.

```
program setup
!------------------------------------------------------------------------
!
!   This program is a example of how to use the podsetup function
!   with Pod-specific setup parameters.
!
!   IMPORTANT NOTES:
!       All character strings are case insensitive.
!
!       Standard (not Pod-specific) setups use syntax with single quoted
!       arguments and double quoted values, eg.:
!               podsetup 'report intr' "off"
!
!       Pod specific setups either use a single single-quoted argument
!       or a single-quoted argument followed by a non-quoted value, eg.:
!               podsetup 'intr_ack on'
!               podsetup 'seg_reg ds' $F800
!
!------------------------------------------------------------------------

                                         ! Note, even though ¬READY and HOLD
                                         ! are Pod specific, "enable" is not,
                                         ! so these use the "built-in" syntax.
        podsetup 'enable ¬ready' "off"
        podsetup 'enable hold' "off"

                                         ! These examples are all 80286H
                                         ! Pod-specific.
        podsetup 'standby address' $8800
        podsetup 'standby function on'
        podsetup 'intr_ack on'
        podsetup 'over_ram base_adr' $C40
        podsetup 'over_ram rdy_src auto'
        podsetup 'over_ram function on'

                                         ! Note upper case works too.
        podsetup 'ERRMASK CONTROL' $F7
        podsetup 'ERRMASK FORC&INT' $BF
        podsetup 'SEG_REG CS' $8C00
end setup
```

Table E-1. 80286H Pod Setup Parameters

POD SETUP	RANGE/KEY	DEFAULT	NOTES
ENABLE $\overline{\text{READY}}$ key	ON, OFF	ON	
ENABLE HOLD key	ON, OFF	ON	
STANDBY ADDRESS nnnn	0-FFFF	FFFF	Data is A8-A23.
STANDBY FUNCTION key	ON, OFF	ON	
INTR_ACK key	ON, OFF	OFF	
OVER_RAM FUNCTION key	ON, OFF	OFF	
OVER_RAM BASE_ADR nnnn	0-FFE0	FFE0	Default=FFE000.
OVER_RAM RDY_SRC key	UUT, AUTO	UUT	
ERRMASK CONTROL nn	0-FF	FF	
ERRMASK FORC&INT nn	0-FF	FF	
ERRMASK HI_ADDR nn	0-FF	FF	A16-A23 mask.
ERRMASK LOW_ADDR nnnn	0-FFFF	FFFF	A0-A15 mask.
ERRMASK DATA nnnn	0-FFFF	FFFF	
ERRMASK MISC nnnn	0-3D00	3D00	
SEG_REG CS nnnn	0-FFFF	F000	Runuut special
SEG_REG DS nnnn	0-FFFF	0	reg. contents.
SEG_REG SS nnnn	0-FFFF	0	"
SEG_REG ES nnnn	0-FFFF	0	"

List of Pod Sync Modes E-6.

A list of the Pod Sync Modes and the mnemonic for each is listed below:

NAME	MNEMONIC
Address Sync	ADDR
Data Sync	DATA
Interrupt Acknowledge Sync	INTA
Bus Cycle Sync	BUSCYCL

Pod Sync Calibration Data E-7.

Calibration is the process by which the internal delay lines in the I/O Module and Probe are adjusted to correctly align (in time) the clock and signals to be sampled. To calibrate an I/O Module or Probe to a Pod for a particuiar Pod sync mode, you are prompted to probe a signal on the UUT. The specified reference edge on that signal is found by adjusting the delay lines in the I/O Module or probe relative to the internal Pod Sync signal. The appropriate delay, labeled "tcal" (as shown in Figure E-1) may vary from one sync mode to another. If calibration is not performed, a default setting is used for the tcal value. When calibration is performed, the measured value for tcal replaces the default value.

Once the reference edge is found, an offset is applied to that edge to determine where in time the I/O Module or Probe latches data.

Figure E-1 contains an example of calibration for address sync that shows how the offset data listed in Table E-2 applies to real waveforms. The reference edge for address sync is the rising edge of $\overline{S1}$. The offset data shows that a valid address is best captured when sampled 45 ns before the rising edge of $\overline{S1}$. (A positive offset would indicate that the address should be latched after the reference edge.)

As a result of the calibration process, the offset value is set to that specified for the sync mode in use. If other offsets are required, the TL/1 "setoffset" statement can be used. See the "setoffset" statement and the "getoffset" statement in the 9100-Series TL/1 Reference Manual for exact syntax and details.

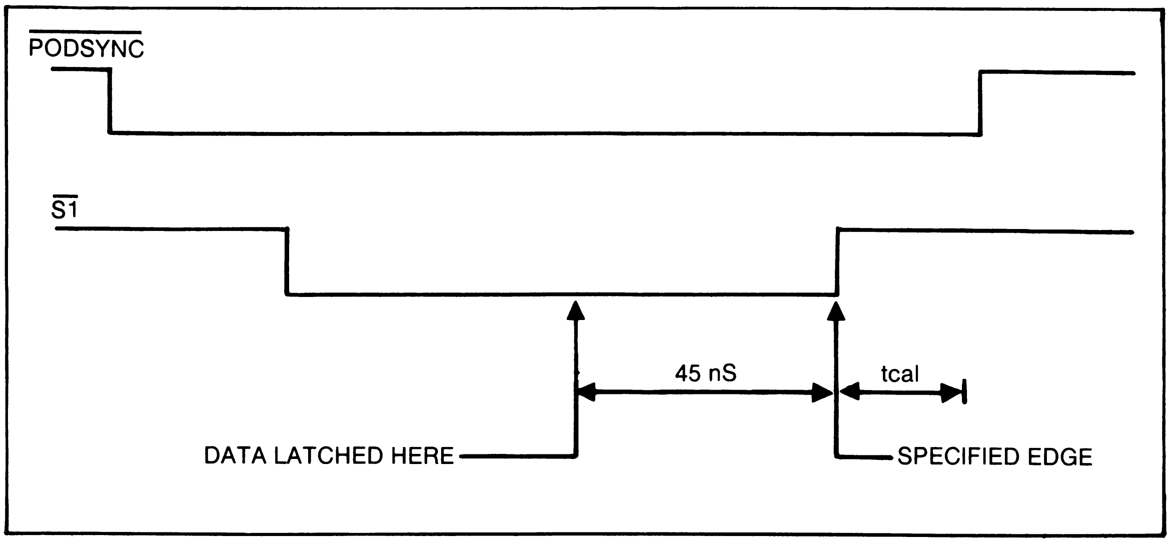


Figure E-1. 80286H Pod Address Sync Timing

Table E-2. 80286H Pod Sync Calibration Data

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	$\overline{S1}$	Rising	-45 ns
DATA	DEN*	Falling	-10 ns
INTA	DEN*	Falling	-10 ns
BUSCYCL	DEN*	Falling	-10 ns

* DEN is an output signal on pin 16 of the 82288 Bus Controller device. If this device is not present on the UUT, select another signal whose rising edge controls when data transceivers on the local data bus should be enabled.

Available TL/1 Support Programs E-8.

The following list describes the available TL/1 support programs for the 80286H Pod.

- FRC_INT

  This program forces an interrupt acknowledge cycle and returns the interrupt vector found on the data bus. The vector data may be meaningless.

- FRC_RINT

  This program forces repetitive interrupt acknowledge cycles and returns the first interrupt vector found on the data bus. The vector data may be meaningless.

- RC_CSCD

  This program returns the 24-bit interrupt cascade address that was found on the address bus during the last interrupt acknowledge cycle.

- RD_NOARM

  This program returns the most recent interrupt vector found on the data bus but does not re-arm the Pod for further interrupt acknowledge cycles.

- RD_REARM

  This program returns the most recent interrupt vector found on the data bus and re-arms the Pod to respond to the next interrupt with an interrupt acknowledge cycle.

- VEC_CAPT

  This program returns a logical TRUE if an interrupt vector has been stored by the Pod. This is found by examining the contents of bit 9 of the Pod status word. The return value is set to "1" if information has been captured, and cleared to "0" if no interrupt vector has been captured.

- HYP_RAM

  This program embodies all the functions of the 80286H Pod HyperRAM test. For further information about operation of the HyperRAM test, see Section 2 of this manual.

Arguments:	ADDR	This is the starting address of the HyperRAM test. ADDR may be any numeric value between 0 and FFDFFC (word space) or FFDFFE (byte space). Only memory spaces are allowed.
	UPTO	This is the ending address of the HyperRAM test. UPTO may be any numeric value between 2 and FFDFFE (word space) or FFDFFF (byte space). Only memory spaces are allowed. UPTO's value must be greater than that of ADDR.
	DELAY	This field produces a delay between the end of a write pass to RAM and a subsequent read from the RAM. The delay field is a CHARACTER STRING in the range of 0 to 65535 decimal. The default is 250.

Faults: test_aborted

reason "Illegal address in invocation".
This fault is included for consistency with the way 9100 Series Mainframes handle the HyperRAM program and should never be seen in normal operation.

reason "Illegal data in invocation".
This fault is included for consistency with the way 9100 Series Mainframes handle the HyperRAM program and should never be seen in normal operation.

reason "Illegal start address".
The value of the ADDR argument does not conform to the restrictions detailed above.

reason "Illegal stop address".
The value of the UPTO argument does not conform to the restrictions detailed above.

reason "New command entered".
A new command was entered during the execution of the HyperRAM test. This fault should never be raised because the program is in complete control of the Mainframe while it is executing.

reason "Space not supported".
The current address space that the Mainframe is in is not supported by the program.

test_failed

See the heading "TL/1 Fault Conditions" in this appendix for a list of the RAM Test Fault Conditions.

Returns: Nothing.

- HYP_ROM

This program embodies all the functions of the 80286H Pod HyperRAM test. For further information about operation of the HyperRAM test, see Section 2 of this manual.

Arguments:	ADDR	This is the starting address of the HyperROM test. ADDR may be any numeric value between 2000 and FF FFFC (word space) or FF FFFE (byte space). Only memory spaces are allowed.
	UPTO	This is the ending address of the HyperROM test. UPTO may be any numeric value between 2002 and FF FFFE (word space) or 2001 and FF FFFF (byte space). Only memory spaces are allowed. UPTO's value must be greater than that of ADDR.
	MASK	MASK indicates which bits in the ROM data should be included in the signature.
	ADDRSTEP	This is the address increment for the HyperROM test. If ADDRSTEP has the value zero, the HyperROM test will default to a 2-byte address increment. ADDRSTEP must be even in a word space. ADDRSTEP is numeric. ADDRSTEP must not be greater than FFF(hex).

Faults: test_aborted

reason "Illegal address in invocation".
This fault is included for consistency with the way 9100 Series Mainframes handle the HyperROM program and should never be seen in normal operation.

reason "Illegal address increment".
The value of the ADDRSTEP argument does not conform to the restrictions detailed above.

reason "Illegal data in invocation".
This fault is included for consistency with the way 9100 Series Mainframes handle the HyperROM program and should never be seen in normal operation.

reason "Illegal start address".
The value of the ADDR argument does not conform to the restrictions detailed above.

reason "Illegal stop address".
The value of the UPTO argument does not conform to the restrictions detailed above.

reason "New command entered".
A new command was entered during the execution of the HyperROM test. This fault should never be raised because the program is in complete control of the Mainframe while it is executing.

reason "Space not supported".
The current address space that the Mainframe is in is not supported by the program.

test_failed

reason "Inactive bits detected".
Some set of bits has been determined to be inactive.
Slots:

bad_data_mask - A mask in which the set bits have been determined to be inactive.

Returns: A signature of the area of memory that was tested. This signature is equal to the signature generated by the ROM Full test.

TL/1 Fault Conditions E-9.

The following list shows the TL/1 fault conditions that can result from 80286H Pod operation. (Handlers for most fault conditions are based on one of the mask types listed under the heading "Bit Definitions of Fault Masks" further on in this Appendix.)

- Fault Conditions Using the Data Mask

  pod_data_incorrect
  pod_data_tied

- Fault Conditions Using the Address Mask

  pod_addr_tied

- Fault Conditions Using the Status Mask

  pod_forcing_active
  pod_interrupt_active
  pod_timeout_enabled_line

- Fault Conditions Using the Control Mask

  podcontrol_tied

- Fault Conditions With No Fault Mask

FAULT CONDITION NAME	ARGUMENT
podselftest_failed	⟨code⟩
pod_timeout_bad_pwr	⟨none⟩
pod_timeout_no_clk	⟨none⟩
pod_timeout_recovered	⟨none⟩
pod_timeout_setup	⟨none⟩
pod_uut_power	⟨none⟩

Bit Definitions of Fault Masks E-10.

Tables E-3 through E-9 describe the mapping from specific 80286H Pod signals to bit positions in the 64-bit fault masks generated by the built-in functions when they invoke TL/1 fault handlers. In each format example, unmapped positions have a "0" value to maintain the full mask length of 64 positions. Positions labeled "X" correspond to mapped signals. Mapped signals have a "1" value to indicate an active, faulty, or otherwise significant condition. A value of "0" represents the absence of a significant condition for that signal.

Table E-3. Address Signal Mapping to Fault Masks

Address signal fault mask format:

```
"0000000000000000000000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXX"
 |                                       |                       |
 └── Bit 63                              └── Bit 23              └── Bit 0
```

Address signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	A0	34
1	A1	33
2	A2	32
3	A3	28
4	A4	27
5	A5	26
6	A6	25
7	A7	24
8	A8	23
9	A9	22
10	A10	21
11	A11	20
12	A12	19
13	A13	18
14	A14	17
15	A15	16
16	A16	15
17	A17	14
18	A18	13
19	A19	12
20	A20	11
21	A21	10
22	A22	8
23	A23	7
24 ⋮ 63	unused	

Table E-4. Data Signal Mapping to Fault Masks

Data signal fault mask format:

```
"000000000000000000000000000000000000000000000000XXXXXXXXXXXXXXXX"
 |                                               |              |
 └── Bit 63                                      └── Bit 15     └── Bit 0
```

Data signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	D0	36
1	D1	38
2	D2	40
3	D3	42
4	D4	44
5	D5	46
6	D6	48
7	D7	50
8	D8	37
9	D9	39
10	D10	41
11	D11	43
12	D12	45
13	D13	47
14	D14	49
15	D15	51
16 : 63	unused	

Table E-5. Control Signal Mapping to Fault Masks

Control signal fault mask format:

```
"0000000000000000000000000000000000000000000000000000000XXXXXXXX"
 |                                                       |      |
 └── Bit 63                                              └Bit 7 └── Bit 0
```

Control signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	$\overline{\text{LOCK}}$	68
1	HLDA	65
2	$\overline{\text{PEACK}}$	6
3	$\overline{\text{BHE}}$	1
4	$\overline{\text{S1}}$	4
5	$\overline{\text{S0}}$	5
6	M/$\overline{\text{IO}}$	67
7	COD/$\overline{\text{INTA}}$	66
8 : 63	unused	

Table E-6. Forcing Signal Mapping to Fault Masks

Forcing signal fault mask format:

```
"00000000000000000000000000000000000000000000000000000000XXXX00XX"
 |                                                      |   |
 └─ Bit 63                                              └─ Bit 7 └─ Bit 0
```

Forcing signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	$\overline{\text{READY}}$	63
1	HOLD	64
2	unused	
3	unused	
4	RESET	29
5	PEREQ	61
6	$\overline{\text{BUSY}}$	54
7	$\overline{\text{ERROR}}$	53
8 ⋮ 63	unused	

Table E-7. Interrupt Signal Mapping to Fault Masks

Interrupt signal fault mask format:

```
"000000000000000000000000000000000000000000000000000000000000XX00"
 |                                                          | └─ Bit 0
 └─ Bit 63                                                  └─ Bit 2
```

Interrupt signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	unused	
1	unused	
2	NMI	59
3	INTR	57
4 ⋮ 63	unused	

Table E-8. Miscellaneous Signal Mapping to Fault Masks

Miscellaneous fault mask format:

```
"00000000000000000000000000000000000000000000000000XXXX0XXXXXXXXX"
 |                                                 |         |
 └─ Bit 63                                         └─ Bit 13 └─ Bit 0
```

Miscellaneous mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0 ⋮ 7	unused	
8	CAPFAIL	52
9	unused	
10	PWRFAIL1	30
11	PWRFAIL2	62
12	GNDFAIL1	35
13	GNDFAIL2	9
14 ⋮ 63	unused	

Table E-9. Enableable Line Mapping to Fault Masks

Enableable signal fault mask format:

```
"00000000000000000000000000000000000000000000000000000000000000XX"
 |                                                              |
 Bit 63                                                         Bit 0
```

Enableable signal fault mask bit assignments:

MASK BIT	POD SIGNAL	80286 PIN NUMBER
0	$\overline{\text{READY}}$	63
1	HOLD	64
2 ⋮ 63	unused	

The following programs summarize the information in the previous tables and provide examples with TL/1 format and syntax.

```
!**********************************************************************
!
!       This program is an example of a fault handler for the
!       pod_forcing_active condition.  If the condition occurs due to the
!       ¬READY signal, the handler disables further reporting of the
!       pod_reporting_active fault condition.
!
!**********************************************************************
program example1
    handle pod_forcing_active (mask)
        declare string mask
        !Check if forcing line fault is due to ¬READY.  Test if bit position
        !0 in the forcing signal fault mask is active.
        if (mid str mask, from 64, length 1) = "1" then
            !Turn off reporting of forcing line faults.
            podsetup 'report forcing' "off"
        endif
    end pod_forcing_active
    read addr 0
end example1


program faults
!------------------------------------------------------------------------
!
!       This program shows examples of raising each of the built-in
!       primitive faults requiring specific mask arguments associated
!       with the 80286H Pod.  Note that in each case the "mask" argument
!       is a 64-character "logical bit string".
!
!       When executed, this program stops as each fault is reported.
!       Enter "CONT" to display succeeding faults.
!
!------------------------------------------------------------------------
                      ! Raise built-in address drivability fault on A7
        unused = "0000000000000000000000000000000000000000"
        bad_addr_bit = "000000000000000010000000"
        addr_fault_mask = unused + bad_addr_bit
        fault pod_addr_tied mask addr_fault_mask

                      ! Raise built-in data drivability fault on D11
        unused = "000000000000000000000000000000000000000000000000"
        bad_data_bit = "0000100000000000"
        data_fault_mask = unused + bad_data_bit
        fault pod_data_tied mask data_fault_mask

                      ! Raise built-in control drivability fault on HLDA
        unused = "00000000000000000000000000000000000000000000000000000000"
        bad_ctl_bit = "00000010"
        ctl_fault_mask = unused + bad_ctl_bit
        fault pod_ctl_tied mask ctl_fault_mask

                      ! Raise built-in active forcing line fault on ¬BUSY
        unused = "00000000000000000000000000000000000000000000000000000000"
        active_force_bit = "01000000"
        active_force_mask = unused + active_force_bit
        fault pod_forcing_active mask active_force_mask

                      ! Raise built-in active interrupt fault on NMI
        unused ="000000000000000000000000000000000000000000000000000000000000"
        active_intr_bit = "0100"
        active_intr_mask = unused + active_intr_bit
        fault pod_interrupt_active mask active_intr_mask

                      ! Raise built-in miscellaneous fault on GNDFAIL1
        unused = "00000000000000000000000000000000000000000000000000"
        misc_fault_bit = "01000000000000"
        misc_fault_mask = unused + misc_fault_bit
        fault pod_misc_fault mask misc_fault_mask

                      ! Raise built-in timeout fault on ¬READY
        unused="00000000000000000000000000000000000000000000000000000000000000"
        enabled_timeout_bit = "01"
        enabled_timeout_mask = unused + enabled_timeout_bit
        fault pod_timeout_enabled_line mask enabled_timeout_mask
end faults
```

RUNUUT PROGRAM EXAMPLES E-11.

The following examples demonstrate the different forms the "runuut" statement can take when using the 80286H Pod within TL/1 programs.

Runuut at the reset address (default):

```
runuutspecial ()                ! Start at FFFFF0.
```

Runuut in lowest 1M byte address space:

```
runuut addr $8F000
```

Runuut in top 64k bytes of 16M byte address space:

```
runuut addr $FF4800             ! Must have FF prefix.
```

Runuut with segment registers initialized:

```
podsetup 'seg_reg cs' $4000
podsetup 'seg_reg ss' $7800
podsetup 'seg_reg es' $0
podsetup 'seg_reg ds' $2000
                                !Must have F000 prefix,
                                !      i.e., F000XXXX.
runuutspecial addr $F000A02C    !IP reg -> A02C; runs at 4A02C.
```

Runuut with breakpoint specified:

```
runuut addr $C94B8, break $C9572
runuutspecial addr $F00043D6, break $243F2
```

MISCELLANEOUS TL/1 PROGRAMMING TIPS E-12.

The following list describes some miscellaneous TL/1 programming tips for the 80286H Pod.

- Fast Stimulus Programs

  To produce rapid activity on a number of data lines, consider the Quick Ramp Test described in Section 3. This function is accessed in TL/1 through the "readspecial" and "writespecial" statements.

INDEX