9132A-68020

MEMORY INTERFACE POD

Instruction Manual

P/N 747873
January 1989

FLUKE

WARRANTY

COVERAGE

Fluke warrants the 9132A-68020 Memory Interface Pod to be free from defects in material and workmanship under normal use and service for a period of one (1) year from the date of shipment. The warranty does not cover parts that connect directly to the Unit Under Test (flying lead sets, microprocessor sockets, clips, headers, and Sync Adapter assemblies). 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 are 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 ANY 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 your questions about applications and use. Address all correspondence to: JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WASHINGTON 98206, ATTN: Sales Department.

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

IMPORTANT NOTE

Use of the 9132A Interface Pod requires that the 9100-Series Mainframe have software installed that is version 4.0 or later.

Table of Contents

TABLE OF CONTENTS, *continued*

TABLE OF CONTENTS, *continued*

List of Tables

List of Figures

Section 1
Introduction

PURPOSE OF THE INTERFACE POD 1-1.

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

The 9100-Series Digital Test System/Station (referred to hereafter as the Mainframe) is used to service printed circuit boards, instruments, and systems that use microprocessors and ROMs. The 9132A Memory Interface Pod (referred to as the Pod) replaces the boot (P)ROM 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 ROMs.

In normal Mainframe/Pod operation, the Pod adapts the general-purpose architecture of the Mainframe to the specific pin layout of a variety of ROM and microprocessor types. By supplying instructions to the microprocessor through the ROM interface, the Pod gains complete access to devices on the UUT that are connected to the microprocessor's bus. 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 UUT microprocessor is connected through buffers to the UUT ROMs located in the ROM sockets on the ROM modules. The UUT executes the code in the UUT ROMs.

NOTE

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

UNPACKING 1-2.

Unpack the 9132A and inspect each component for possible shipping damage. If the equipment is damaged or there are parts missing, contact your shipping agent or your Fluke sales representative immediately.

Save the original shipping box and any protective shipping devices, including foam packing. If repairs, equipment relocation, or extended storage are necessary, use the original foam-packed shipping containers to prevent unnecessary damage. If the original packaging is not available, order new containers from your Fluke sales representative.

PHYSICAL DESCRIPTION OF THE POD 1-3.

The Pod connects to the Mainframe through a round shielded cable, and connects to the UUT through a set of ROM modules that are inserted into the UUT's boot ROM sockets. The UUT's ROMs are removed from the UUT and are replaced by the Pod's ROM modules. (The UUT's boot ROMs are placed in sockets located on the ROM modules.) In addition, a Sync Module is connected to various lines on the UUT to control timing and reset for the UUT processor. The components of a standard 9132A Interface Pod are shown in Figure 1-1. Figure 1-2 shows the communication between the Pod, the Mainframe, and the UUT.

The Pod consists of a base unit and several attached parts. One Sync Module and up to four ROM Modules may be attached to the Pod. The Pod contains the 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.

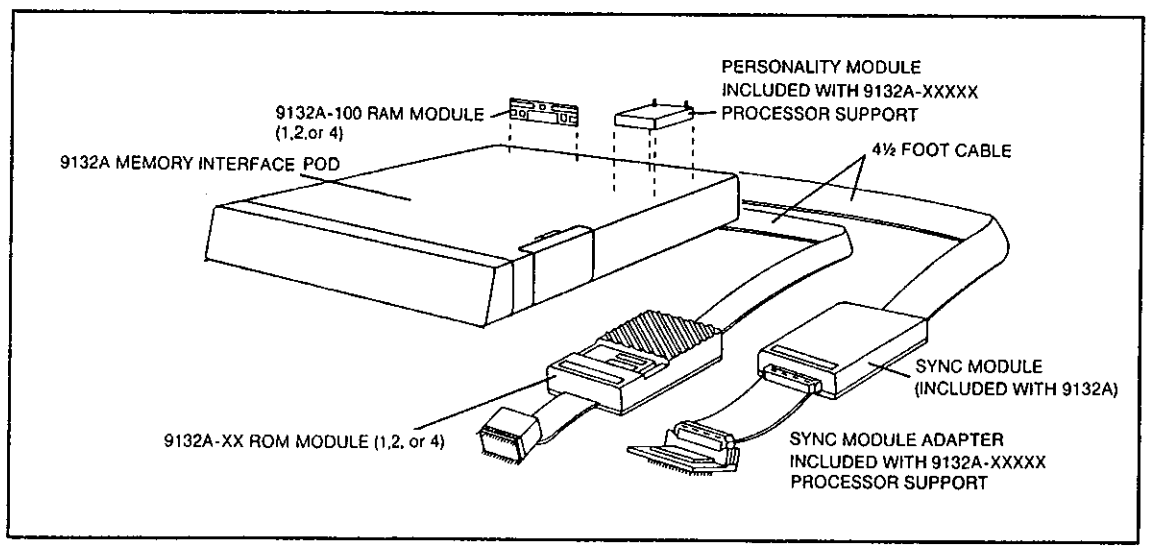


Figure 1-1. Components of a Standard 9132A Pod

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

Overvoltage protection circuits or fuses on each line to the UUT guard against damage to the Pod that could result from the following:

- Incorrectly inserting a ROM Module cable plug into the UUT's boot ROM socket.
- UUT faults that place potentially damaging voltages on the lines to the UUT's boot ROM sockets.

NOTE

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

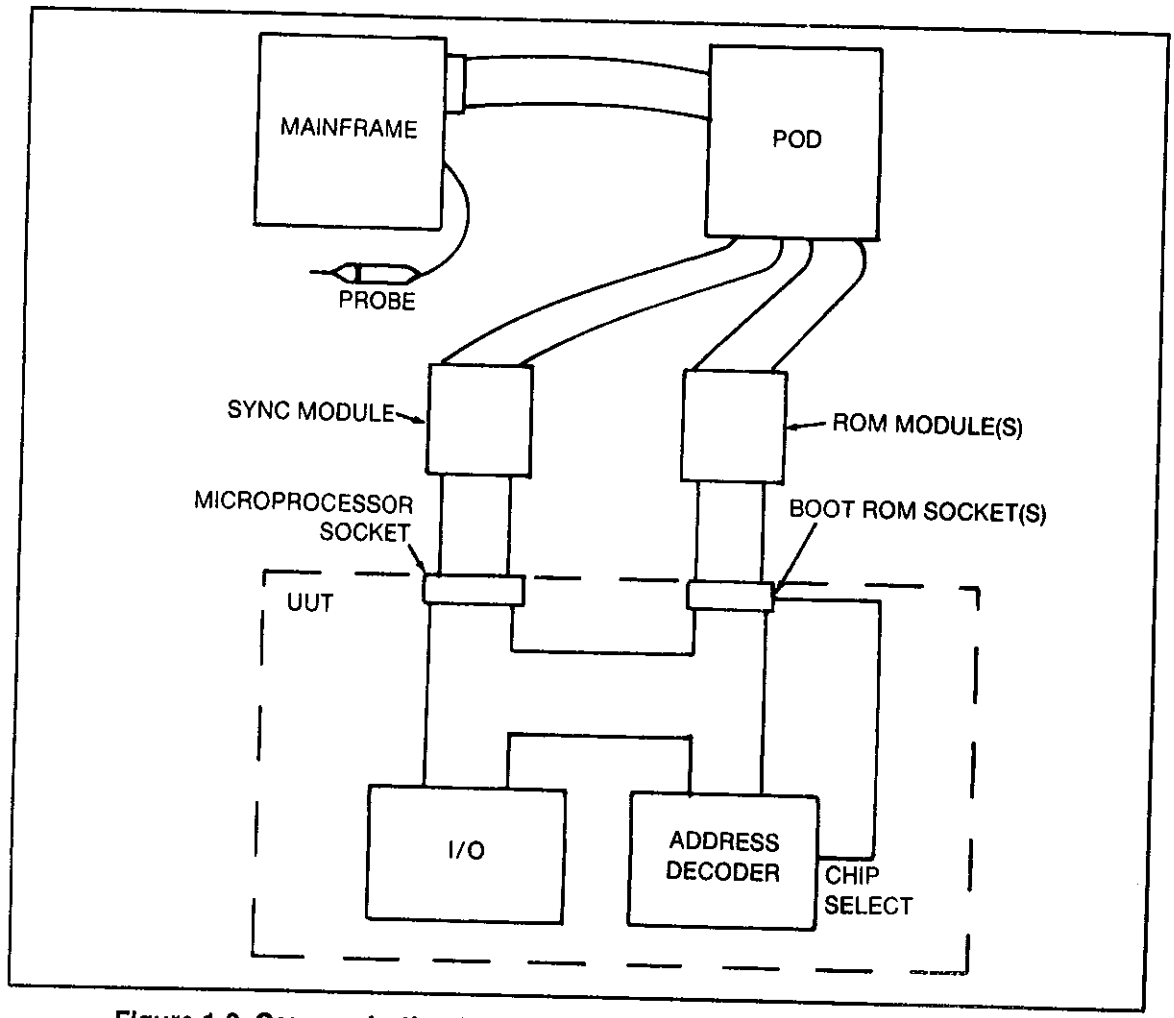


Figure 1-2. Communication Between the Mainframe, the Pod, and the UUT

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

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

POD SPECIFICATIONS

1-4.

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

Table 1-1. 9132A Memory Interface Pod Specifications

EMULATION SPECIFICATIONS

Boot ROM Space Size (See Appendix A) ..... 4K bytes min.

DC ELECTRICAL SPECIFICATIONS

Protection

Maximum External Voltage on a Single UUT Access Pin ..... -7 to +12 volts
ESD Protection on UUT Access Pin ..... 15 kV through 100 pF and 500 ohms without damage to the Pod
Improper Connection to UUT ..... No damage to the Pod or UUT
UUT Power Detection ..... UUT power sensed as present for Vcc > 3.5V (typ.)
UUT Protection against Latchup ..... Pod outputs disabled when UUT Vcc < 3.5V (typ.)
Float Voltage Limit ..... 30V maximum above earth ground
Mechanical interlocks ..... Pod power is interrupted by opening rear panel

Pod Input Specifications

Input Low Voltage ..... 0.8 V max.
Input High Voltage ..... 2.0 V min.
Input Leakage Current ..... ±50 µA, +0.45 ≤ V_{IN} ≤ Vcc

ROM Module Output Specifications

ROM Module Outputs:

Output Low Voltage ..... 0.2V (max.) at I_{OL} = 50 µA
0.8V (typ.) at I_{OL} = 10 mA
Output High Voltage ..... 3.5V (typ.) at I_{OH} = -3 mA
Output Tri-state Leakage ..... ±50 µA, +0.45 ≤ V_{OUT} ≤ Vcc

Sync Module Output Specifications

Overdrive Outputs (for UUT Reset):

Output Low Voltage ..... 0.6V (max.) at I_{OL} = 80 mA pulsed
Output High Voltage ..... 3.5V (min.) at I_{OH} = -80 mA pulsed
Protection ..... Status monitoring provides overdrive lockout

AC ELECTRICAL SPECIFICATIONS (See Figure 1-3)

ROM Module

Access Time:

Address to Data ..... 100 ns max. (t_{ACC})
Slower of CE or OE to Data ..... 50 ns max. (t_{CE})
Data Hold Time (Address to Data) ..... 20 ns min. (t_{OH})
Data Float Time (CE or OE to Data Float) ..... 40 ns max. (t_{DF})

Table 1-1. 9132A Memory Interface Pod Specifications (cont)

AC ELECTRICAL SPECIFICATIONS (cont)

Sync Module

Clock Input (for Sync):

Frequency	0 to 50 MHz
Pulse Width	10 ns min. (high or low)
Other Pulse Widths	20 ns min. (high or low)

ENVIRONMENTAL

Operating Temperature	+5 to 50° C, 8 to 70% RH (noncondensing)
Storage Temperature	-40 to 70° C, 8 to 80% RH (noncondensing)
Temperature Gradient	±10° C per hour

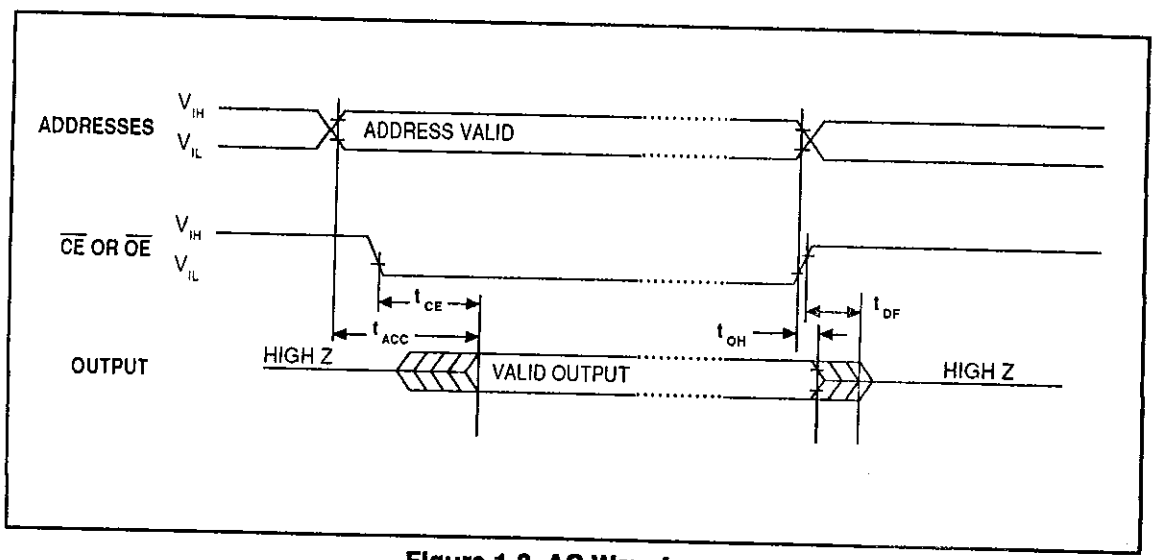


Figure 1-3. AC Waveforms

USING THIS MANUAL

1-5.

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

Section 1 — Explains the purpose of the Memory Interface Pod and contains a general description of the Pod.

Section 2 — Contains installation instructions for connecting the Pod to the Mainframe and the UUT. This section also contains a description of the built-in self test to ensure that the Pod is functioning correctly, and a setup function to tailor Pod operating characteristics to a UUT type.

Section 3 — Describes how to set various functions that are specific to the processor on your UUT. Section 3 also describes the address structure of the Pod and lists the processor signals on your UUT.

Appendices — Contains miscellaneous material that may prove valuable when using the Pod.

LIST OF REPLACEMENT COMPONENTS 1-6.

Because the plugs and clips associated with the 9132A can be attached and detached from the UUT many times, it is possible that pins on the plugs or the clip leads may break. Replacement components may be ordered from John Fluke Mfg. Co., Inc. or an authorized representative by using the Fluke Stock Number. Table 1-2 contains ordering information for these parts.

Table 1-2. 9132A Ordering Information

DESCRIPTION	STOCK NUMBER
9132A-4406 Flying Lead Set	777078
Hook Clip	757500
Pincer Clip	845409
24-pin Header	845391
28-pin Header	845388
32-pin Header	845396
114-position PGA Socket	851378
9132A-7612 68020 Standard Sync Adapter Assembly	777482
9132A-7610-02 68020 Personality Module	849344
Fuse, .25 x 1.25, 0.25A, 250V, Fast (USA)	109314
Fuse Holder Part, Cap, 5 x 20 mm (European)	461020

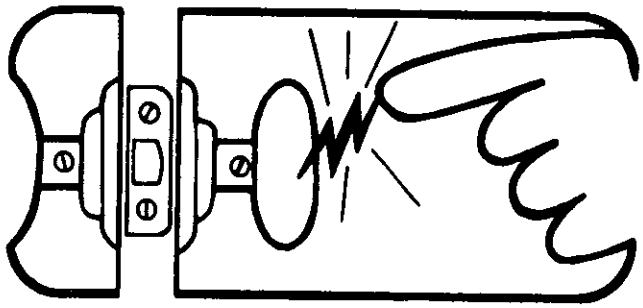
68020 SOCKET ADAPTER ACCESSORIES 1-7.

The Sync Adapter Assembly is designed to have the minimum size needed to connect to the UUT microprocessor socket. If you find it difficult to probe points around the microprocessor, consider using a socket adapter to raise the Sync Adapter Assembly (such as the CS114-72TG socket from Advanced Interconnections Corp.).

static awareness

A Message From

John Fluke Mfg. Co., Inc.

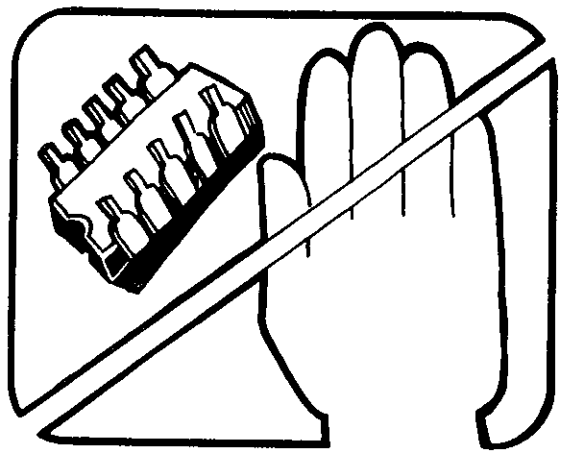


Some semiconductors and custom IC's can be damaged by electrostatic discharge during handling. This notice explains how you can minimize the chances of destroying such devices by:

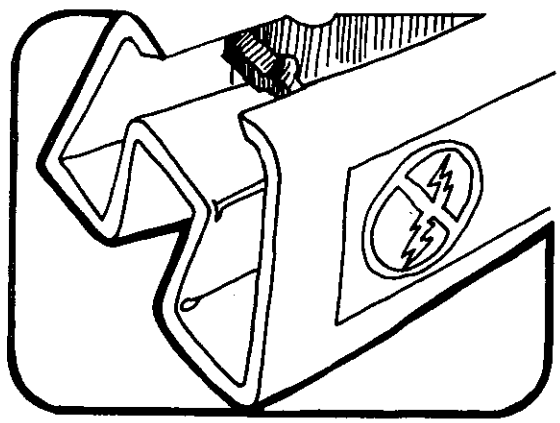
1. Knowing that there is a problem.
2. Learning the guidelines for handling them.
3. Using the procedures, and packaging and bench techniques that are recommended.

The Static Sensitive (S.S.) devices are identified in the Fluke technical manual parts list with the symbol "⊗"

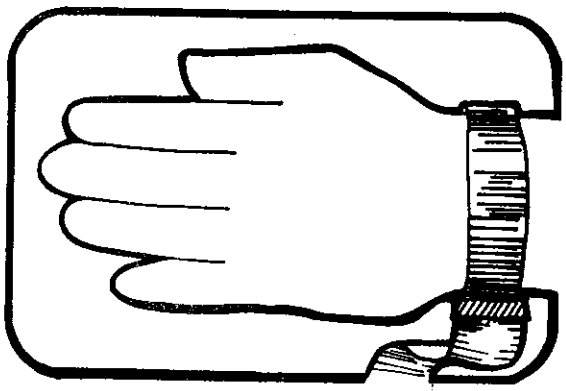
The following practices should be followed to minimize damage to S.S. devices.



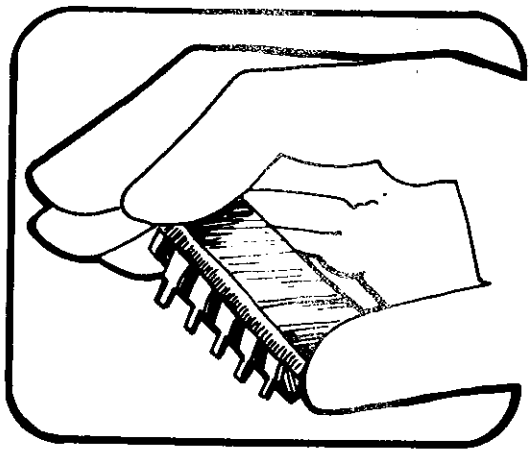
1. MINIMIZE HANDLING



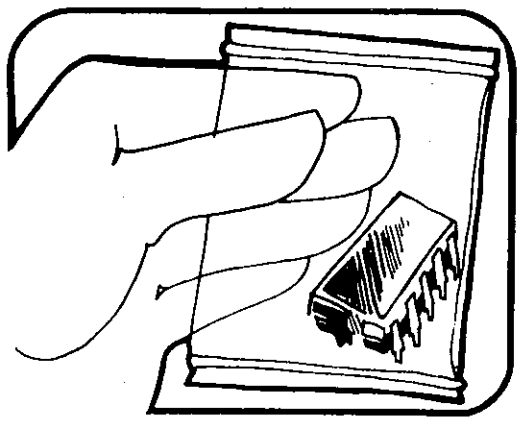
2. KEEP PARTS IN ORIGINAL CONTAINERS UNTIL READY FOR USE.



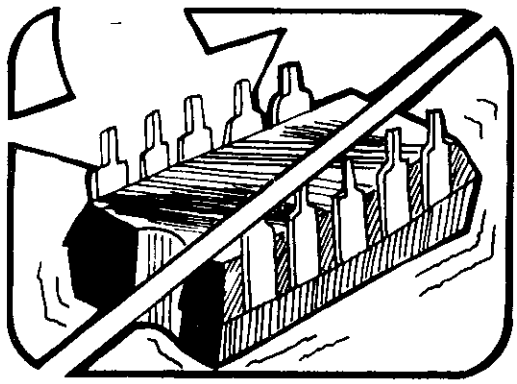
3. DISCHARGE PERSONAL STATIC BEFORE HANDLING DEVICES. USE A HIGH RESISTANCE GROUNDING WRIST STRAP.



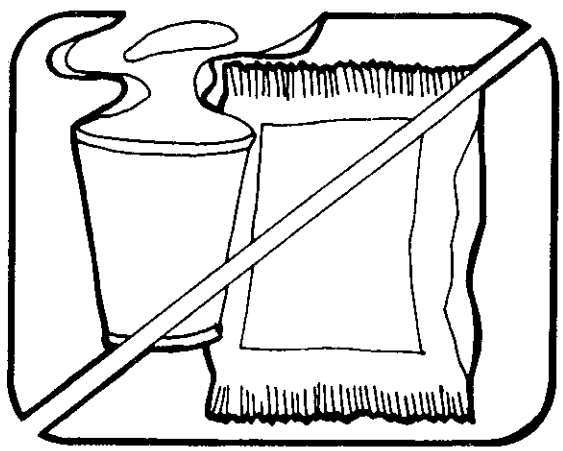
4. HANDLE S.S. DEVICES BY THE BODY



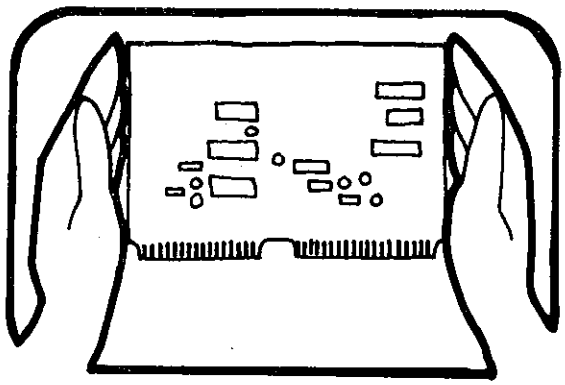
5. USE STATIC SHIELDING CONTAINERS FOR HANDLING AND TRANSPORT



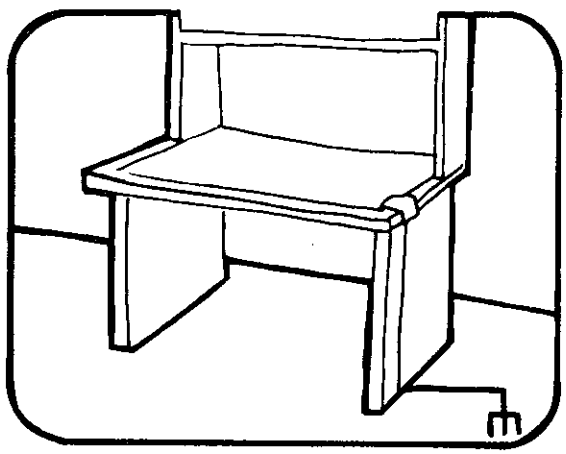
6. DO NOT SLIDE S.S. DEVICES OVER ANY SURFACE



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 2
9132A Setup for 9100-Series Mainframes

GETTING STARTED 2-1.

This section contains setup and installation instructions for the Pod. These instructions show how to install processor support to test UUTs that use the 68020 processor and how to connect the external and internal modules that support your specific 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 ROM and Sync Modules to the UUT, initialize the Pod and the UUT, and begin testing.

INSTALLING THE 68020 DATABASE 2-2.

The 68020 database for the 9100-Series Mainframe is contained on one 3.5-inch floppy disk supplied with the 9132A-68020.

To install the database on a Mainframe with a hard drive, insert the disk into the Mainframe floppy drive, press MAIN MENU on the keypad, press SOFT KEYS, then select COPY DISK FROM DR1 TO HDR and press ENTER. (The database only needs to be installed once.) Once the database is installed on the Mainframe, place the original floppy disk in a safe place in case it is needed in the future.

To use the database on a Mainframe with two floppy drives, first copy the database disk to a backup disk using the Mainframe COPY function. Remove the original from the floppy drive and insert the copy into the system USERDISK. Each time the Mainframe is reset the database is read from the floppy disk. Place the original floppy disk in a safe place in case it is needed in the future.

INSTALLING PROCESSOR AND ROM SUPPORT 2-3.

Before installing or changing the internal or external modules in the Pod, use the following steps to gain access to the module connectors inside the back panel of the Pod:

1. Check that the Mainframe power is OFF.

2. Open the back panel of the Pod by turning the thumbscrews on each side counterclockwise, and then pulling the panel out from the case (see Figure 2-1).

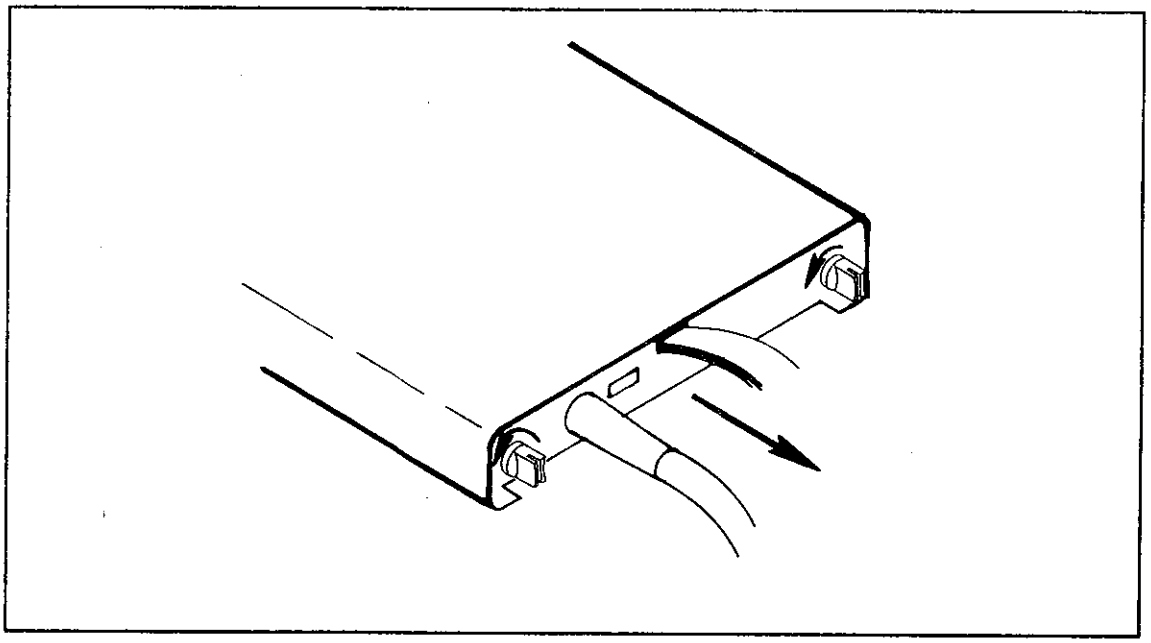


Figure 2-1. Opening the Back Panel of the Pod

Installing the Personality Module 2-4.

To configure the Pod for the 68020 processor, a 68020 Personality Module must be installed in the Pod. The following steps describe the procedure for installing the Personality Module:

1. Carefully plug the personality module in the connector on the left-hand side of the main printed circuit assembly (pca) as shown in Figure 2-2.

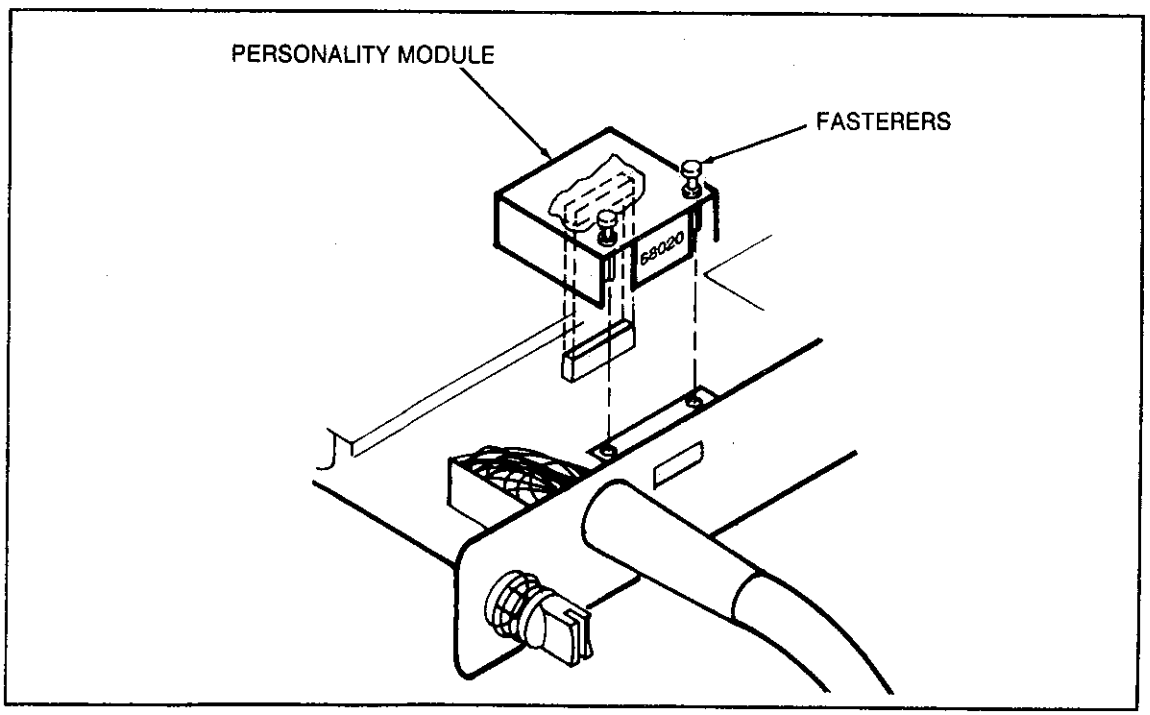


Figure 2-2. Installing the Personality Module

2. Push down on the fasteners on the back of the personality module to snap it into place. The processor type should show through the "Configured for" slot in the back panel.

Installing the Sync Module 2-5.

The following steps describe the procedure for installing the Sync Module into the Pod:

1. Plug the Sync Module cable into the connector labeled "SYNC" near the back of the main pca (as shown in Figure 2-3), making sure not to twist or kink the Sync Module cable.

2. After plugging in the Sync Module (and one or more ROM Modules), secure the cables by tightening the cable tie located on the back panel of the Pod.

Installing the ROM Module(s) 2-6.

The following steps describe the procedure for installing the ROM Module(s) into the Pod:

1. Plug the ROM Module(s) into the numbered connectors on the main pca (as shown in Figure 2-3), making sure not to twist or kink the ROM Module cables. If you are using one ROM Module, that module must be plugged into the sockets marked "ROM 1." If you are using two ROM Modules, the modules must be plugged into the sockets "ROM 1" for ROM Module number 1 and "ROM 2" for ROM Module number 2.

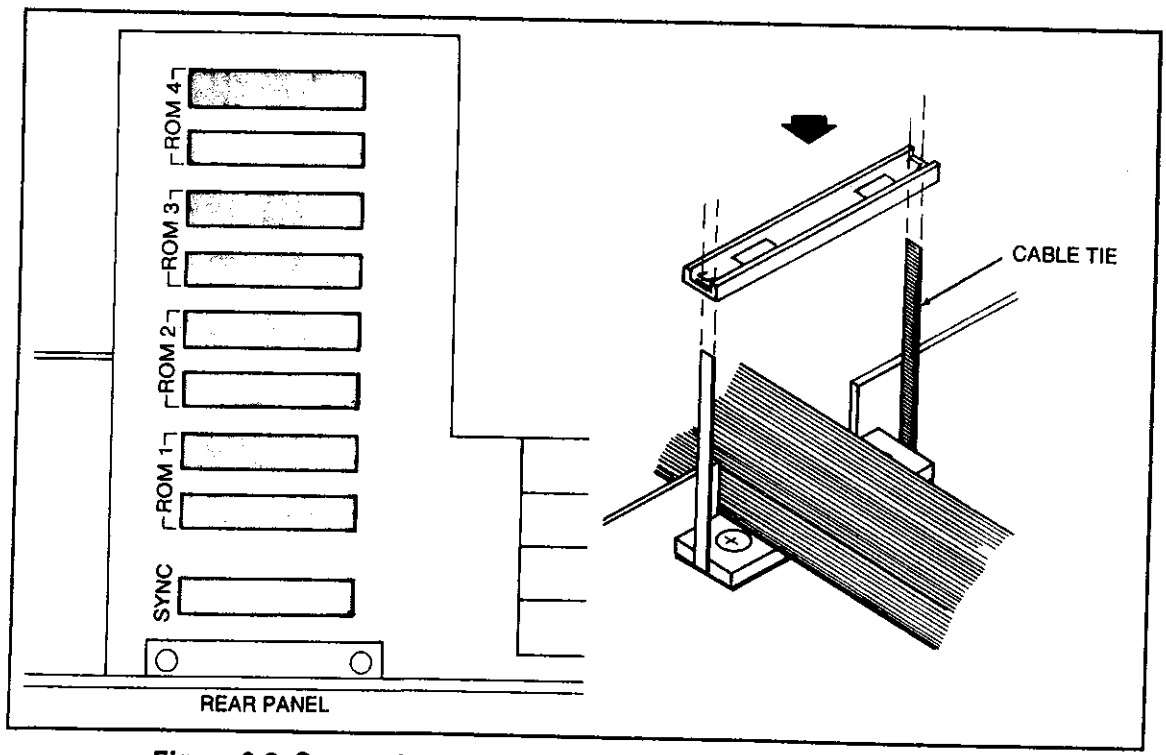


Figure 2-3. Connection of the External Modules to the Interface Pod

NOTE

To prevent confusion after the back of the Pod is closed, each of the ROM Modules should be numbered as you insert them into the ROM Module connectors. Stick one of the numbered labels (provided with the Pod) on the back of the UUT connector plug.

2. After plugging in the ROM Module(s) (and the Sync Module), secure the cables by tightening the cable tie located on the back panel of the Pod.

Installing the RAM Module(s) 2-7.

For each ROM Module that has been installed in the Pod, an equivalent number of RAM Modules must also be installed. (For convenience, the Pod operates correctly if more RAM Modules than ROM Modules are installed.)

Plug the RAM Module(s) into the connectors on the right side of the main pca (as shown in Figure 2-4). If only one ROM Module has been installed, install only one RAM Module in the "RAM 1" connector. If two ROM Modules have been installed, install RAM Modules into "RAM 1" and "RAM 2" connectors. If four ROM Modules have been installed, install RAM Modules in all the RAM connectors.

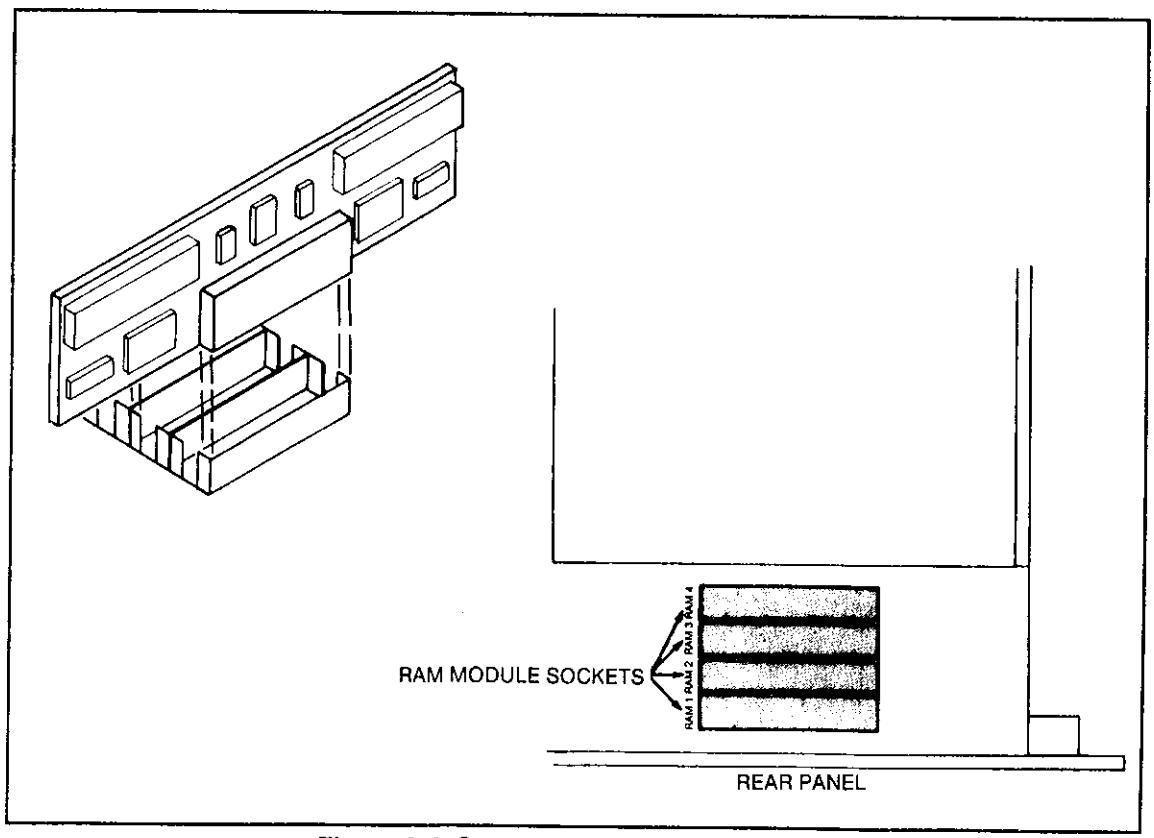


Figure 2-4. Connecting the RAM Modules

Closing the Pod Case

2-8.

After installing or changing the internal or external modules in the Pod, push the back panel back into the Pod case. Make sure the ROM Module and Sync Module cables are inserted properly into the slot on the top of the panel. Tighten the thumbscrews by pressing them in and turning clockwise.

CONNECTING THE POD TO THE MAINFRAME

2-9.

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 2-5. Secure the connector using the sliding collar.

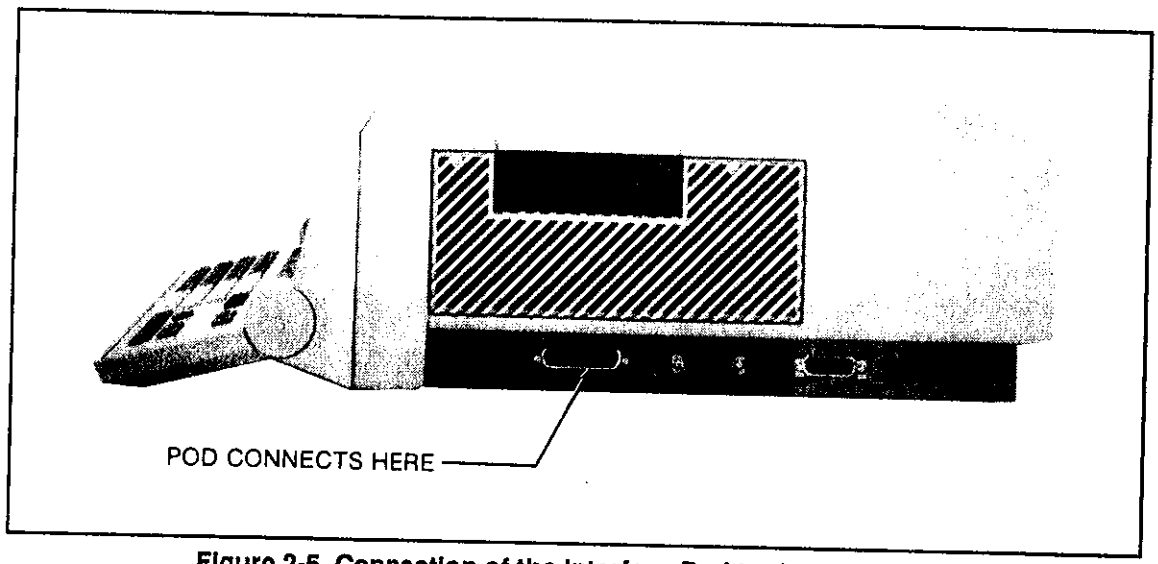


Figure 2-5. Connection of the Interface Pod to the Mainframe

PERFORMING THE POD SELF TEST

2-10.

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

1. Make sure that the Pod is connected properly to the Mainframe.
2. If the ROM Modules are connected to a UUT, release the ROM Module cables from the UUT's ROM sockets. Remove the UUT ROMs from the ROM Module sockets.

NOTE

You must remove the UUT boot ROMs from the ROM Module sockets for self test to work properly. Self test may indicate an error if the UUT ROMs are not removed.

3. If the Sync Module is connected to a UUT, disconnect the Sync Module from the UUT.

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

5. Insert the Sync Module Adapter Board cable into the socket on the self test pca (as shown in Figure 2-6). Connect the UUT Reset line flying lead to the reset line test connector and the UUT Reset ground line flying lead to the ground line test connector on the self test pca. (The Sync Module Reset ground line flying lead must be disconnected from the UUT during self test, otherwise an error will occur.)

NOTE

If you are not using the standard Sync Module Adapter assembly, unplug your adapter assembly and cable from the Sync Module and replace them with the standard Sync Module Adapter assembly cable.

6. Insert the ROM Module cable plug into the Zero-Insertion Force (ZIF) self test socket (as shown in Figure 2-6):

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

   b. Insert the pins of the ROM Module cable plug into the ZIF socket. If you are using the 24-pin, 28-pin, or 32-pin ROM Module, insert the cable plug into the 32-pin socket. If you are using the 40-pin ROM Module, insert the cable plug into the 40-pin socket. Ensure that pin 1 of the ROM Module cable plug is inserted into pin 1 of the ZIF socket for the corresponding size.

NOTE

Pin 1 for a 24-pin, 28-pin, and 32-pin ROM Module cable plug is indicated on the 32-pin ZIF socket. If the ROM Module cable plug is inserted incorrectly, self test cannot determine if a ROM Module is connected and displays an error message.

   c. Secure the ROM Module cable plug in place by pushing down the latch lever.

7. Begin the Pod self test. Press the Main Menu key on the Mainframe to obtain the display "MAIN: SELFTEST POD". Press the ENTER key.

   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 Appendix F for an explanation of the failure codes.

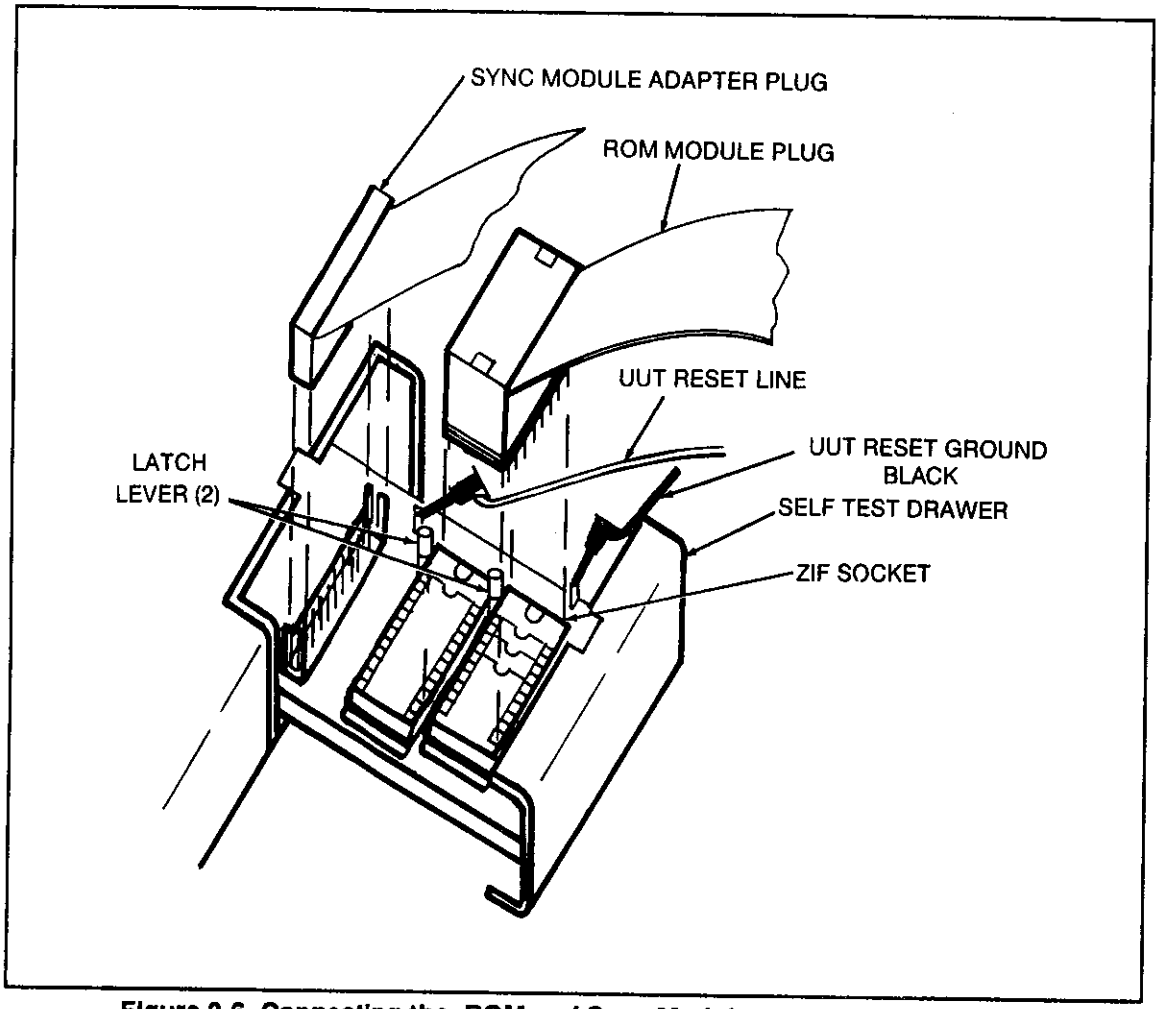


Figure 2-6. Connecting the ROM and Sync Modules to the Self Test PCA

8. Remove the ROM Module from the Self Test Socket.

9. Repeat steps 6 through 8 for all the ROM Modules connected to the Pod.

CONNECTING THE POD TO THE UUT 2-11.

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 ROM sockets. If the ROMs are still in the sockets, remove them. Place the ROMs into the ZIF sockets on the ROM Modules (ROM 1 into ROM Module 1, ROM 2 into ROM Module 2, etc.) as shown in Figure 2-7.

CAUTION

UUT power must be off when the UUT boot ROMs are removed from the UUT. Otherwise damage to the ROMs may occur.

CAUTION

The UUT boot ROMs must be plugged into the ROM Module sockets correctly or damage to the ROMs may occur.

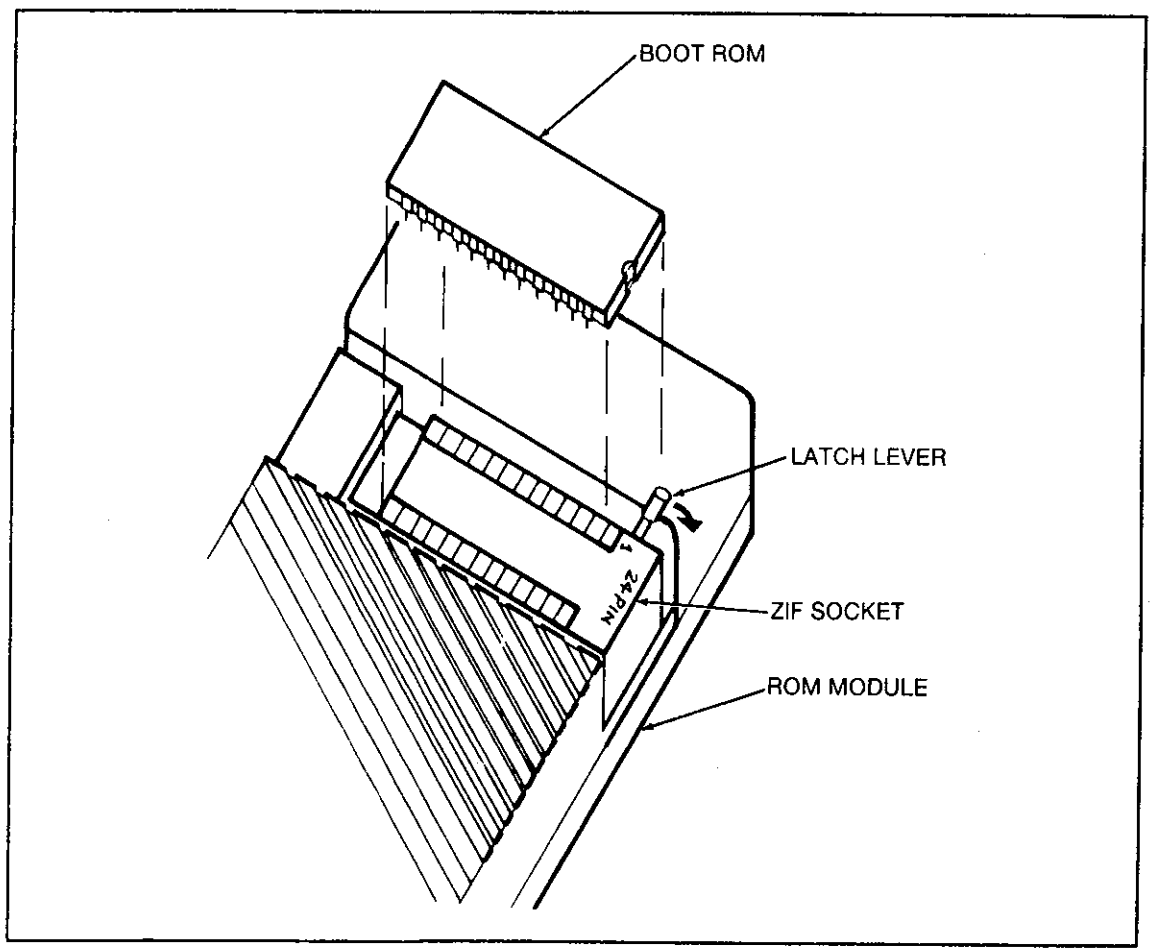


Figure 2-7. Inserting UUT ROMs into the ROM Module

NOTE

UUT data bits 24 through 31 correspond to ROM Module 1, data bits 16 through 23 to ROM Module 2, data bits 8 through 16 to ROM Module 3, and data bits 0 through 7 to ROM Module 4. If the UUT has an 8-bit boot ROM data bus, use only ROM Module 1. For a 16-bit boot ROM data bus, use ROM Modules 1 and 2. For a 32-bit boot ROM data bus, use ROM Modules 1, 2, 3, and 4.

4. Insert the ROM Modules into the UUT's ROM sockets and secure them (using the same means used to secure the ROMs). Make sure that pin 1 of the ROM Module plug is aligned with pin 1 of the ROM socket.

CAUTION

UUT power must be turned off before plugging the ROM Modules into the UUT boot ROM sockets or the UUT boot ROMs in the ROM Module sockets may be damaged.

NOTE

The ROM Module plug ends in an adapter that is easily damaged if not carefully inserted and removed from the UUT ROM sockets (a spare adapter is included with the ROM Module). Before testing, ensure an adequate supply of replacements are available. The part is included in the List of Replacement Components in Section 1.

If the UUT uses soldered-in ROMs, see Appendix C of this manual, Testing UUTs With Soldered-in Components. A list of ROMs that can be replaced by the ROM Modules is contained in Appendix A of this manual.

5. Connect the Sync Module adapter board to the Sync Module (as shown in Figure 2-8). Remove the 68020 processor from the UUT and plug it into the socket on the adapter board. Connect the adapter board plug into the processor socket on the UUT. Connect the UUT RESET line (white wire) on the Sync Module to system reset on the UUT. Connect the UUT RESET ground line (black wire) to UUT ground.

NOTE

Connect the UUT RESET clip from the Sync Module to the UUT system reset line, not the $\overline{RESET}$ pin of the 68020 microprocessor. For more information about connecting the Sync Module RESET clip, see Appendix G.

The Sync Module adapter board adapts the general design of the Sync Module to the processor-specific design of the UUT. In some cases, it may be necessary to design a different type of adapter board or test connector for a UUT. Table 2-1 contains a list of microprocessor signals and pin numbers for a typical adapter board connected to the Sync Module adapter board cable. For a proper connection between the adapter board cable and the custom adapter board, use a 3M connector (part number 3494-2002) or equivalent.

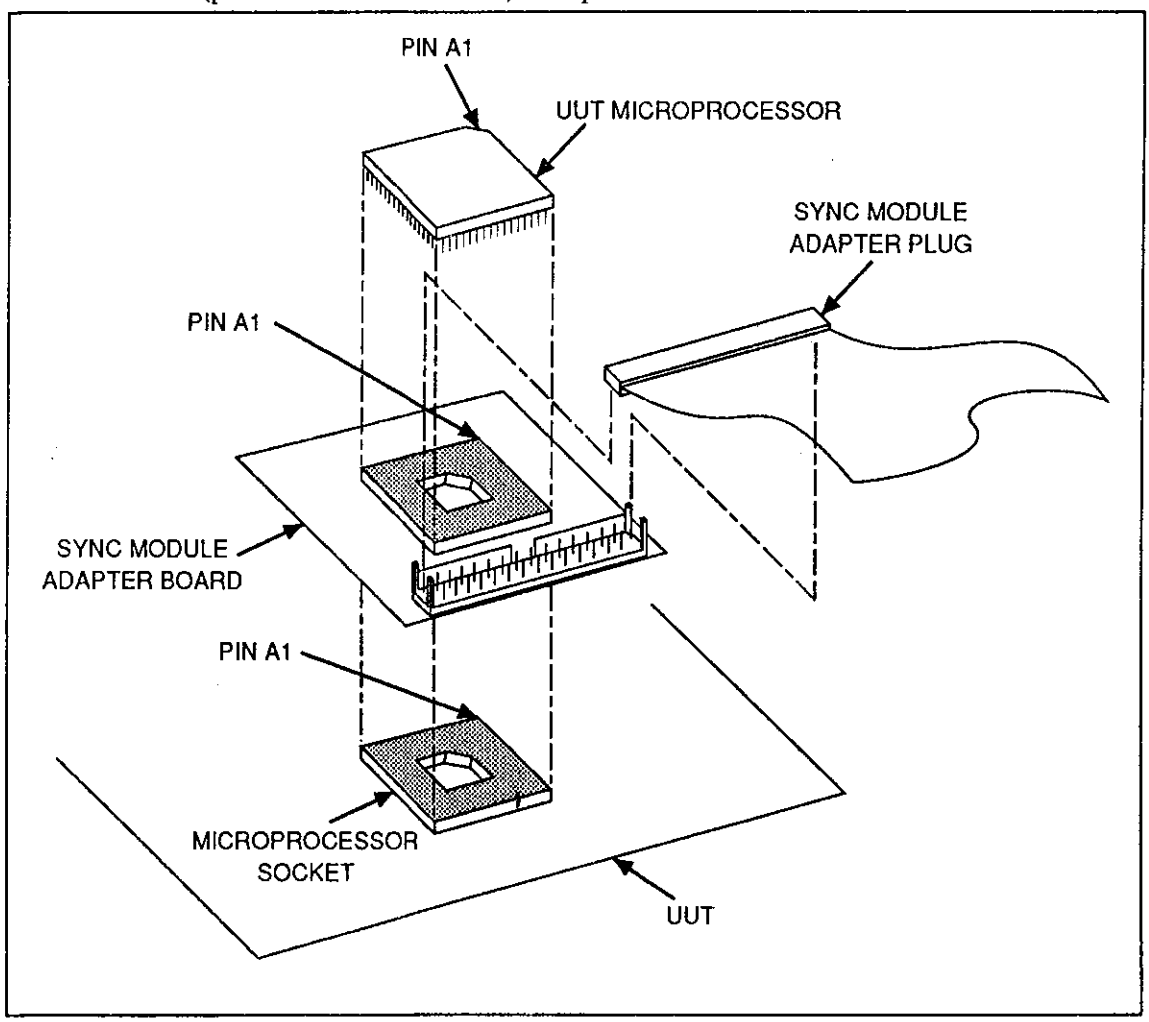


Figure 2-8. Connecting the Sync Module to the UUT

If the UUT uses a soldered-in microprocessor, see Appendix C of this manual, Testing UUTs With Soldered-in Components.

NOTE

The Sync Module plug ends in an adapter that is easily damaged if not carefully inserted and removed from the UUT processor socket. Before testing, ensure an adequate supply of replacements are available. The part is included in the List of Replacement Components in Section 1.

Table 2-1. Sync Adapter Signals

PIN NUMBER	SIGNAL	PIN NUMBER	SIGNAL
1	GND	2	D24
3	GND	4	D25
5	GND	6	D26
7	GND	8	D27
9	GND	10	D28
11	GND	12	D29
13	GND	14	D30
15	GND	16	D31
17	GND	18	CLK
19	GND	20	$\overline{\text{RESET}}$
21	GND	22	$\overline{\text{AS}}$
23	GND	24	$\overline{\text{BR}}$
25	GND	26	$\overline{\text{BG}}$
27	GND	28	$\overline{\text{BGACK}}$
29	GND	30	$\overline{\text{HALT}}$
31	GND	32	$\overline{\text{BERR}}$
33	GND	34	(KEY)

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

CAUTION

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

7. Apply power to the UUT.

POD SETUPS

2-12.

Since the Pod tests UUTs with a variety of different ROM types and number of boot ROMs, the unique features of each UUT must be described to the Pod (i.e., the characteristics of the UUT Reset, the type of ROM, the ROM data width, etc.). These attributes can be configured individually, or the Interactive Setup and Calibration routine can be used to configure the attributes automatically.

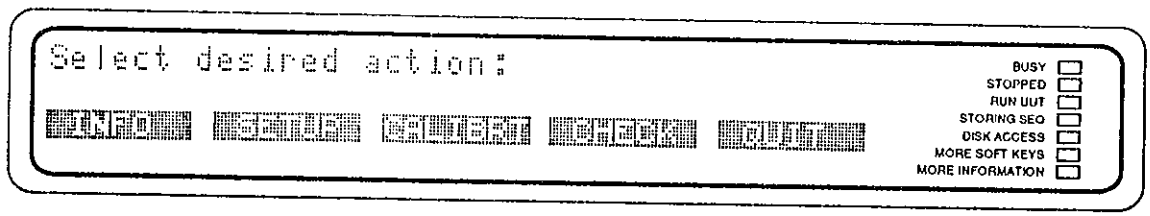
This section describes automatic configuration. Manual configuration is described in Appendix D for front panel operation and in Appendix E for TL/1 programming applications.

Use a known good UUT to set the attributes with the Interactive Setup and Calibration routine when making the first connection to a new type of UUT. Once the setups are verified with the known good UUT, the attributes should be saved so the setup values can later be restored to test and troubleshoot other UUTs of the same type.

Interactive Setup and Calibration 2-13.

To select the Interactive Setup and Calibration routine, press the POD key on the Mainframe keypad, select the SETUP softkey, and press ENTER.

Once the program has loaded, the Mainframe displays the following menu:



The INFO softkey contains information about the basic operation of the routine. First-time users are encouraged to read this information before continuing with the operation of the routine.

The Setup, Calibrate, and Check routines must be run in the correct order to ensure the Pod attributes are set correctly. (Once the Setup and Calibration routines are run, use the Check routine to verify that the Pod can communicate correctly with the UUT.)

Press the SETUP softkey to begin describing the physical makeup of your UUT to the Pod. (SETUP must be run before CALIBRT to ensure that the values under these routines are accurately set.) Setup runs through a series of questions that determine general information about the UUT, queries how the UUT is connected to the Pod, and verifies that the correct connections are made. These questions are presented in the following order:

1. Verify the UUT communications address (XFER_ADR). UUT to Pod communications depend on one address on the UUT that is written to by the UUT. The transfer address can be set to any legal address as long as reads or writes to the address do not cause the UUT to halt or cause a bus exception. Each time data is communicated with the Pod, the UUT microprocessor reads and saves data from the specified address, transfers data to the Sync Module with a write cycle, then restores the original data with a second write cycle.

   Pod accesses at the XFER_ADR are made with supervisor data long bus cycles.

   If the transfer address is correct, press OK (F1). To change the transfer address, press CHANGE (F2), enter the new address, and press ENTER. The new address is then confirmed.

2. Verify the UUT boot ROM type (ROM_TYPE).

   If the UUT boot ROM type is correct, press OK (F1). To change the ROM type, press CHANGE (F2) and select the new type. If your UUT ROM type is not shown on the display, select OTHER and use the information contained in Appendix A of this manual to set the ROM description number.

3. Verify the number of ROM Modules (ROM_MODS) connected to the UUT. UUTs tested by the Pod can have boot ROM with bus sizes that vary from design to design. A byte-wide boot ROM requires one ROM Module, a word-wide boot ROM requires two ROM Modules, and a longword-wide boot ROM requires four ROM Modules.

   If the number of ROM Modules is correct, press OK (F1). To change the number of ROM Modules, press CHANGE (F2) and select the new number. The new number is then confirmed.

4. Verify that the ROM Module(s) are connected to the UUT boot ROM sockets. If the UUT has soldered in boot ROMs, see Appendix C.

   After the ROM Modules are connected, press OK (F1).

   If the correct order of ROM Modules cannot be determined, connect the ROM Modules in any order (the order is verified during Steps 11 and 12.)

5. Verify that the Sync Module is connected to the UUT (in most cases, the standard Sync Module Adapter board replaces the microprocessor in the UUT , which is placed in the socket on the Sync Module Adapter board). If the UUT has a soldered in microprocessor, see Appendix C.

   After the Sync Module is connected, press OK (F1).

6. Verify that the RESET lead from the Sync Module is connected to the UUT (see Appendix G for information about where to connect the RESET lead to the UUT).

   After the Sync Module RESET lead is connected, press OK (F1).

7. Verify the RESET pulse polarity (RST_POL). This is the polarity that is asserted on the RESET lead from the Sync Module to the UUT. (Since the RESET lead is generally not connected directly to the UUT microprocessor, the connection polarity may differ from the $\overline{\text{RESET}}$ line of the microprocessor.)

   If the RESET pulse polarity is correct, press OK (F1). To change the polarity, press CHANGE (F2) and select the correct polarity. The new RESET pulse polarity is then confirmed.

8. Verify the RESET pulse length (RST_LEN). The Reset line on different UUTs may require different signal durations to reset the board successfully. This function allows you to select the length of the RESET signal (in microseconds) sent from the Pod to the UUT.

   If the RESET pulse length is suitable, press OK (F1). To change the pulse length, press CHANGE (F2), enter the length of the pulse, and press ENTER. The new RESET pulse length is then confirmed.

9. Verify that power to the UUT is on by pressing OK (F1). During this step, the Pod performs a power sensing function at all the ROM Module connections. Press CONT (F1) to proceed to the next step.

10. Verify the RESET connection.

This step requires the use of the Mainframe probe. To verify the RESET connection, press OK (F1). You are asked to probe the RESET pin of the UUT microprocessor and then press the probe button. When the button is pressed, the Pod resets the UUT, which is detected by the probe. If the RESET pulse is determined to be incorrect, an error message is displayed. If the pulse is determined to be correct, the connection is confirmed.

To skip this check, press the SKIP (F2) softkey (skipping this step may be appropriate if the SETUP function has been previously executed and RESET verified at that time).

11. Automatically verifies the data path to ROM Module 1.

    If the ROM Modules are connected to the wrong UUT ROM sockets, the error is detected and reported. Once the ROM Modules are reconnected, you are allowed to run the verification again. If for some reason ROM Module 1 cannot be identified with this step, the program continues to the next step.

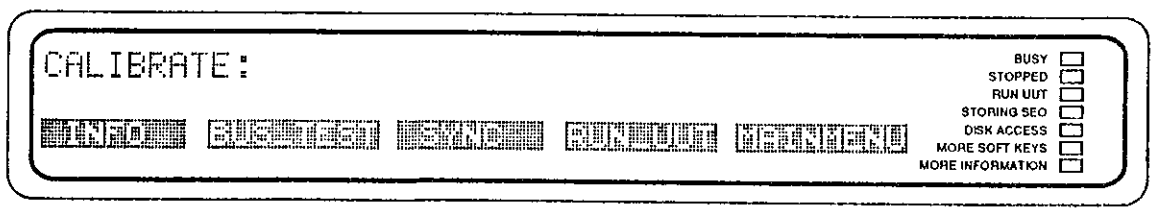
12. Verify the remaining data paths to any other ROM Modules.

    This step requires the use of the Mainframe probe. To verify the data paths to the ROM Module(s), press OK (F1). You are asked to probe a data line at the UUT microprocessor and then press the probe button. If Step 11 was unable to identify ROM Module 1, you are asked to probe D0. If ROM Module 1 was successfully identified in Step 11, you are asked to probe either D8, D16, or D24 (depending on the number of ROM Modules). If the data line you probe is determined to be incorrect (i.e., the ROM Module is in the wrong socket), an error is detected and reported. If the data line is determined to be correct, the connection is confirmed.

    To skip this check, press the SKIP (F2) softkey.

13. The Setup routine is complete and a list of parameters that have been automatically set are displayed. Press ENTER/YES to return to the main menu.

Once the Setup routine has been completed, the Pod can now be calibrated to the UUT. Press CALIBRT (F3) to begin the Calibration routine. The Mainframe displays the following menu:



The INFO softkey contains information about the operation of the options listed in this menu. First-time users should read this information before continuing with the operation of these options.

NOTE

BUS_TEST (F2) must be run before RUN_UUT (F4) to ensure accurate calibration.

BUS_TEST (F2) is the first routine to use. This routine performs a series of steps that calibrate setup parameters that are required for Bus Test to interact properly with the UUT. These steps are presented in the following order:

1. Performs a series of tests on the UUT and automatically sets the bus cycle clock signal source (INTRFACE BCYCLCLK) and the ratio of boot ROM addresses accessed to boot ROM enables (INTRFACE BURST_SZ).

2. Probe A3 at the UUT microprocessor with the Mainframe probe. Press the probe button. This allows the routine to set the number of bus cycles expected between UUT reset and the appearance of the stimulus address on the UUT address bus (CALIBTN ADR_STIM) and specify the bus width in bytes at the microprocessor divided by the number of ROM Modules (CALIBTN CY_SPLIT). Press ENTER/YES to continue to the next step.

3. Verifies that the Bus Test Calibrations are complete and that the routine is now ready to perform read/write sync pulse calibrations. Press ENTER/YES to return to the calibration menu.

SYNC (F3) is the next routine to perform. This routine calibrates the read/write sync pulses generated by the Pod during UUT accesses. Begin by probing A3 at the UUT microprocessor with the Mainframe probe. Press the probe button. The Pod performs a series of reads and writes to the UUT. Continue to hold the probe on A3 until the test concludes. Once the routine is finished, press CONT (F1) to return to the calibration menu.

NOTE

If the setup procedure fails during the SYNC routine, it may be due to UUT memory management circuit hardware that requires a different setup procedure. For more information, see the heading, Using the XFER_CAL Setup, in Appendix D. Once the Transfer calibration has been changed, repeat the SYNC routine.

RUN_UUT (F4) is the final calibration routine to perform. This routine automatically sets the number of microprocessor bus cycles expected between UUT reset and the fetch instruction at the RUN UUT starting address (CALIBRTN RUN_UUT). Press CONT (F1) to return to the calibration menu.

To exit the calibration menu, press MAINMENU (F5).

Once the Setup and Calibration routines have been run, use CHECK (F4) to verify that the Pod can successfully interact with the UUT. Upon entering the

Check routine, probe A3 of the UUT microprocessor with the Mainframe probe and press the probe button. As each test is performed, a pass or fail message is displayed. When the Check routine has run through the tests, a global pass or fail message is displayed. To return to the main menu, press CONT (F1). CHECK is also useful for confirming restored setups (those that were previously saved with the Mainframe SAVE SYSTEM SETTINGS).

To exit the Interactive Setup and Calibration routine, press QUIT (F5).

Saving and Restoring Pod Setups 2-14.

Once the Pod setup and calibration information has been configured, use the SAVE SYSTEM function in the Setup Menu of the Mainframe to store the information in either a UUT file or on the Userdisk. To reload the information, use the RESTORE SYSTEM function in the Setup Menu of the Mainframe. For more information on the SAVE and RESTORE functions of the Mainframe, see the 9100-Series Technical User's Manual.

Setup for Relocated Boot ROM Address Space 2-15.

Most UUTs have the boot ROM code located at the 68020 microprocessor's reset address. Some, however, have boot ROM relocated to a different region of memory. In such cases, the UUT boot address must be changed from the default value of 0. See the heading, Using the 68020 ROM_BASE Setup, in Appendix D for more information.

TROUBLESHOOTING HINTS 2-16.

When you begin testing, certain conditions on the UUT may cause the Pod to behave unpredictably. If this happens, check the following list to see if one of the conditions exists on your UUT (also see Appendix B, Problems Due to a Marginal UUT).

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

Section 3
Pod Operations with 9100-Series Mainframes

INTRODUCTION 3-1.

Once the Pod is connected to the Mainframe and UUT, testing of a 68020-based UUT can begin. This section describes the functions of the Pod and demonstrates how these functions operate while using the 9100-Series Mainframes. Also included in this section is a description of the 68020 processor signals.

Before you begin testing, 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.

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 (←).

USING THE POD 3-2.

To properly troubleshoot a non-functional UUT with the 9132A Pod, you must first ensure that the portion of the UUT kernel connected to the UUT boot ROMs and the Pod Sync Module is operational. Without an operational kernel, other tests (such as read, write, and RAM test) may return unreliable information. Therefore, the first step in testing any nonfunctional UUT is to run Bus Test. If the initial test indicates that the kernel is nonfunctional, Bus Test provides you with an extended set of features for troubleshooting the kernel. Once the kernel is functional, you can use the other features of the Pod to locate problems within other parts of the UUT.

NOTE

Because the Pod uses the built-in instruction cache of the 68020, the speed of tests are affected by the level present on the $\overline{CDIS}$ line. Testing still proceeds if the cache has been disabled by the assertion of $\overline{CDIS}$, but may take up to twice as long to finish.

TESTING THE UUT BUS 3-3.

There are two major procedures to Bus Test in the 9132A Pod: test and diagnose. The first phase tests the portion of the UUT bus connected to the boot ROMs to determine if there is a problem. If a failure does occur, the Bus Test branches to a set of enhanced diagnostic features to help determine the cause of the problem.

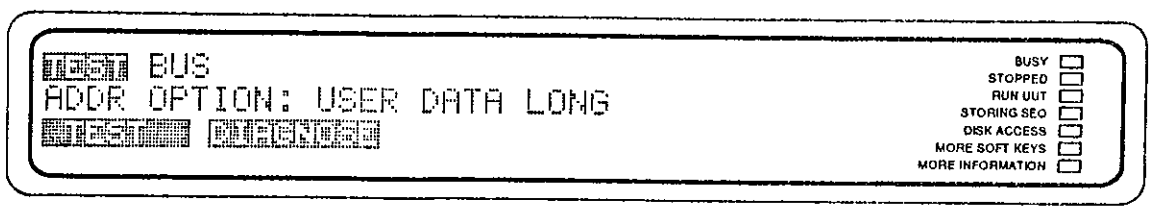
NOTE

Before running Bus Test you must set the attributes of the UUT by either restoring the settings from a UUT file or the User Disk as described in Section 2, by using the Interactive Setup and Calibration routine as described in Section 2, or by setting them manually using the descriptions in Appendix D.

Testing with Bus Test 3-4.

Use the following steps to begin testing the UUT bus:

1. Press the BUS key on the Mainframe keypad. The Mainframe displays the following message:



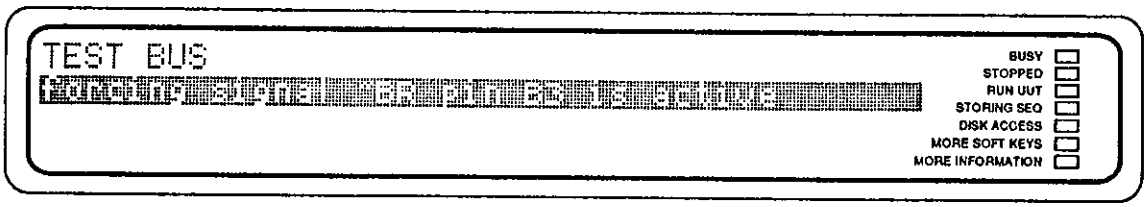
NOTE

Though the address option is shown on the second line of the Mainframe display, the Pod ignores the options set in the address option field during Bus Test.

2. Press ENTER on the Mainframe to begin the bus test.

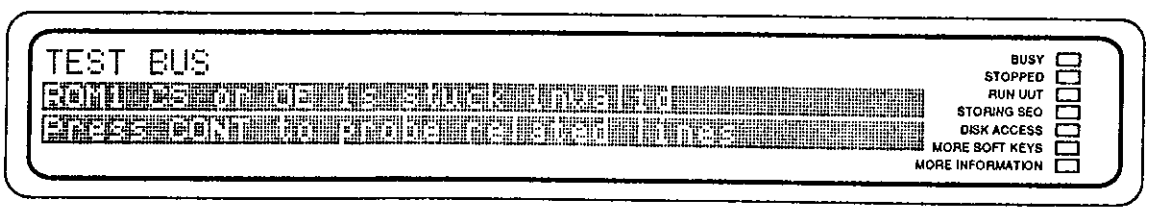
If the bus test passes, the Mainframe displays PASSED. If a failure occurs during the bus test, the Mainframe begins to diagnose the problem. (A manual diagnostic routine can be run by selecting DIAGNOSE BUS on the Mainframe Bus Test.)

There are two major steps to the diagnostic features of the Bus Test. The first step is to determine if the cause of the error can be detected without probing. Since the Bus Test can only detect that part of the UUT kernel that is connected to the UUT boot ROMs and Sync Module, the test checks the lines connected to this portion of the UUT first. If the diagnostic procedure detects an error in this portion of the kernel, the Mainframe displays a message showing the location of the fault. For example, if the Bus Test diagnostics determine that $\overline{BR}$ is stuck low, the following message is displayed:

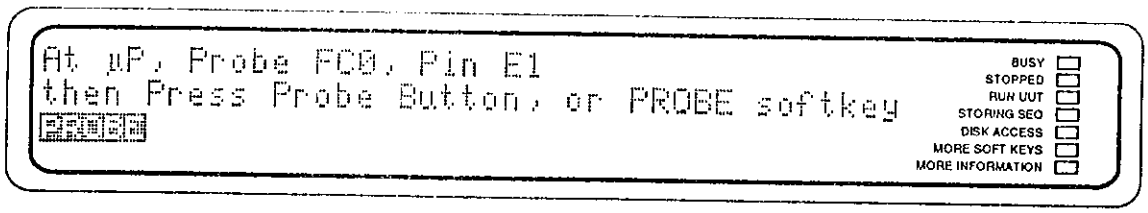


If the Bus Test diagnostic routines do not detect a fault on the lines connected to the Sync Module, or if the symptoms are inconclusive, a further diagnostic step begins. If Bus Test first displays a fault message indicating the general nature of the UUT failure, press the CONT key to continue the diagnostic. Bus Test prompts you to measure activity on additional lines with the Mainframe probe. In some cases, the probe uses its enhanced capabilities to remeasure lines that are monitored by the Sync Module, providing new information to the Bus Test routine.

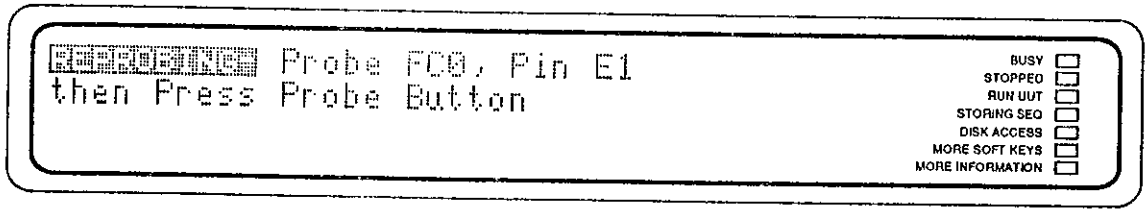
For example, in a UUT with the FC0 line stuck high, the following display may appear after the initial portion of the test:



After you press CONT, Bus Test prompts for the FC0 line to be probed:



Place the probe on the UUT line indicated by the Mainframe and press the probe button. If no fault is detected on that line, the Mainframe displays a message saying the line is OK. After a short time, the Mainframe displays another line to try. If, however, a fault is detected, the Mainframe beeps several times and prompts you to try the line again (in case you may not have made good contact with the line), as shown in the following display:



If the fault is detected again, the Mainframe beeps several times and a diagnostic fault condition message is displayed.

The Bus Test diagnostic routines continue to prompt you to test the data lines, status lines, and control lines of the microprocessor until a fault is found. If, for any reason, you choose to probe a different line than shown by the prompt, press the PROBE (F1) softkey. Since Bus Test checks UUT signals in groups (i.e., data, status, control) the signal you select after pressing the PROBE key must fall within that group. Two softkeys, PREV and NEXT, allow you to scroll through the group of lines. To select the line you want to test, press the ENTER key.

As Bus Test progresses, faults are reported as they are encountered. After each fault message is displayed, press CONT on the Mainframe keypad to continue testing. Many times, related or additional faults are reported. If no other faults occur, the message "TEST BUS = FAILED" is displayed. Generally, the first fault reported is the primary fault, though other reported faults could have caused the primary fault. Continue testing until Bus Test reports all the faults it can find, then determine which is the primary fault and which are secondary effects.

To exit from the diagnostic routines once a fault message is displayed, press STOP.

NOTE

The Pod monitors power at the ROM Module connections to the UUT boot ROM sockets (power and ground pins on the UUT microprocessor are not monitored). If the UUT kernel is not functional because a power or ground line connected to any ROM Module is open, Bus Test detects and reports this condition.

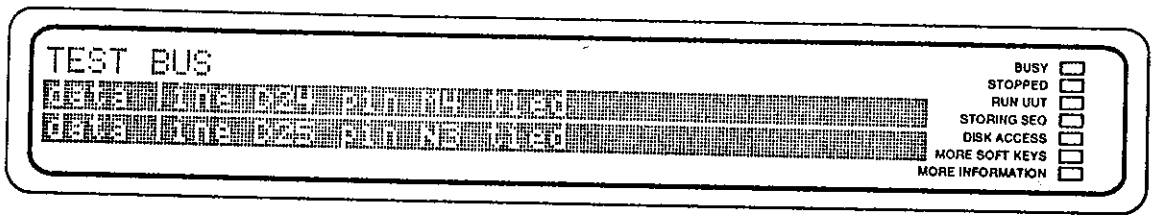
Examples of Bus Test Operations and Fault Messages 3-5.

The following examples are included to illustrate how Bus Test detects various faults. These fault diagnostic examples not only demonstrate simple fault messages, but also show how various signals of the 68020 microprocessor interact with other signals on the processor. In the latter case, the CONT key on the Mainframe keypad can be used to continue testing the UUT once an initial fault is found.

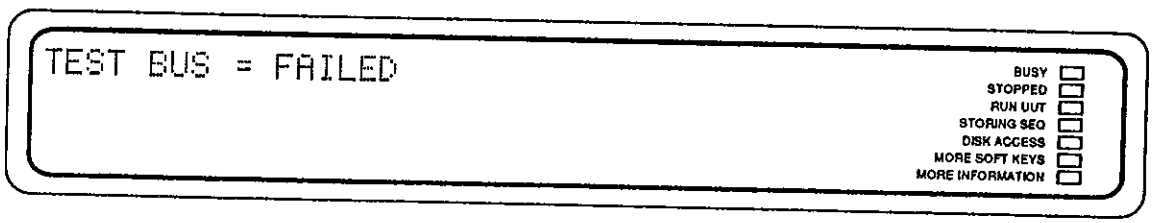
Example 1: D24 tied to D25.

When TEST BUS is run, you are first prompted to probe various status and control lines. These lines could cause Bus Test failure symptoms similar to the data line faults and must be verified before the data lines are checked.

Upon completion of the probing, the data lines are automatically tested. Since these lines are among those monitored by the Sync Module, no probing is required to test them. When the fault is diagnosed, the following fault message is displayed:



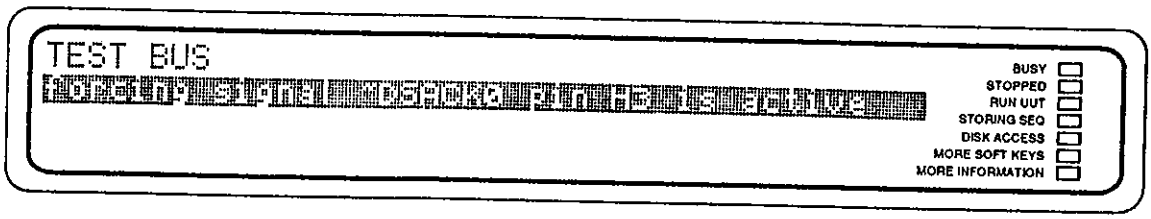
If CONT is pressed, Bus Test determines there are no further lines to test, so the following message is displayed:



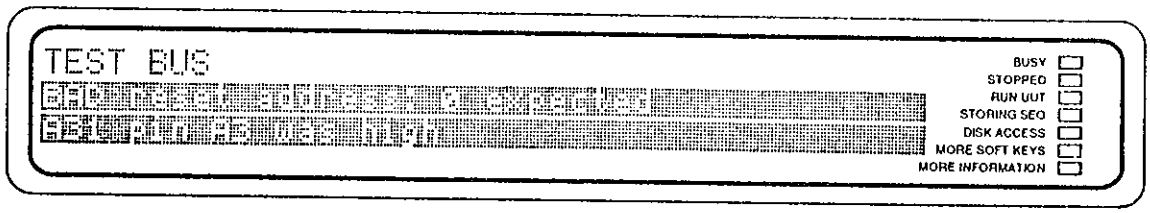
This example demonstrates a Bus Test in which no additional faults are found after the initial fault message, indicating that the two shorted lines are the only problem.

Example 2: A31 shorted high (in a UUT that includes A31 in the Boot ROM decoding).

When TEST BUS is run, the following fault message is displayed:



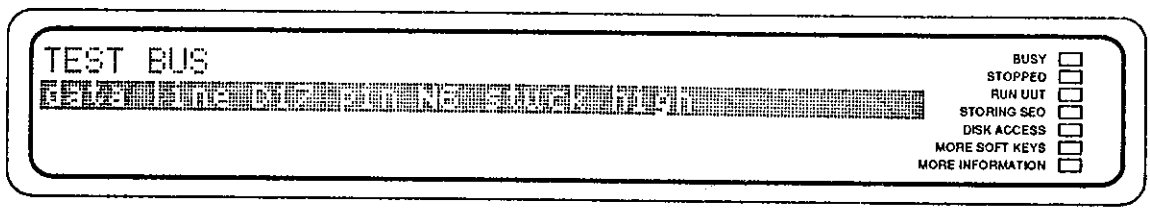
If CONT is pressed, the following fault message is displayed:



This example demonstrates some of the interaction between lines typically found in 68020 UUT faults. Since A31 is used by the UUT to generate DSACK0, the original message accusing DSACK0 of being stuck was correct. Pressing CONT caused TEST BUS to examine additional lines, which detected the A31 fault.

Example 3: D17 stuck high and D20 stuck low.

When TEST BUS is run (and after the status and control lines are probed), it determines there is a problem on the data bus and prompts you to probe the additional data lines. When D17 is probed, a fault is found and the following message is displayed:

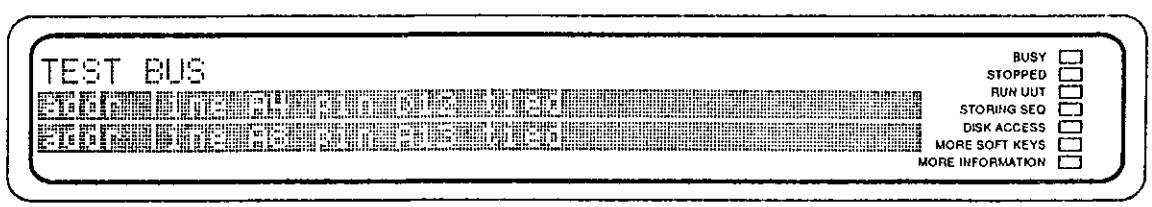


If CONT is pressed, you are prompted to probe successively higher data lines, starting with D18. When D20 is reached, a fault message for that line is displayed.

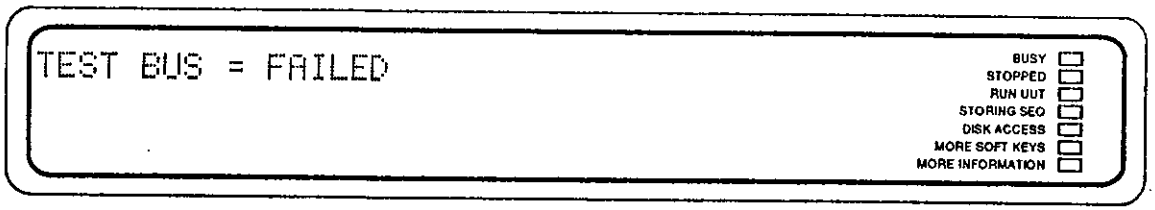
This example demonstrates that continuation is possible after a fault message is displayed. TEST BUS always generates a fault message as soon as one is found, but, by pressing CONT, the test continues to diagnose the rest of the lines.

Example 4: A8 tied to A4.

When TEST BUS is run(and after the status and control lines are probed), the following fault message is displayed:

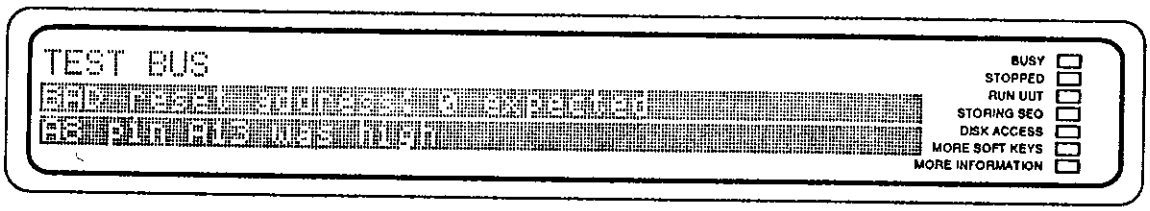


If CONT is pressed, the following message is displayed:

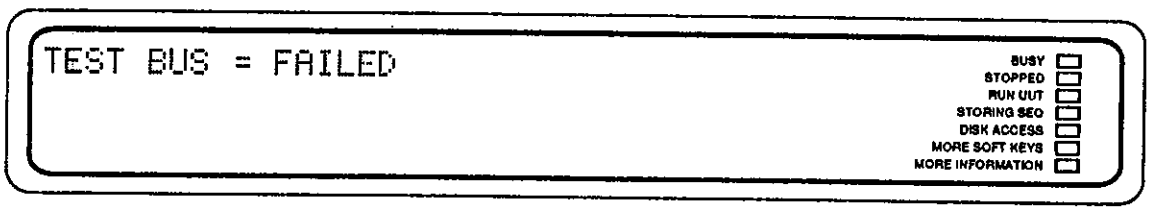


Example 4a: A8 stuck high, A3 stuck low.

When TEST BUS is run, the following fault message is displayed:



If CONT is pressed, the following message is displayed:



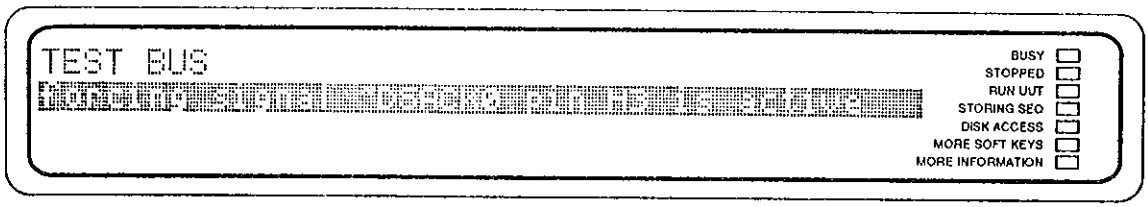
Note the differences between the fault messages in Example 4 and Example 4a. These examples show the hierarchical method of kernel testing used with the 68020 Pod. Once Bus Test is started, the UUT processor is reset, which generates the reset address. Once the reset address is generated on the UUT bus, more tests are run by Bus Test.

In example 4, A8 was tied to A4. Both of these lines are expected to be low during the reset address. Because the reset address was correct, the Pod was able to fetch enough opcodes to correctly perform the STIM_ADR subtest and diagnose the address bus.

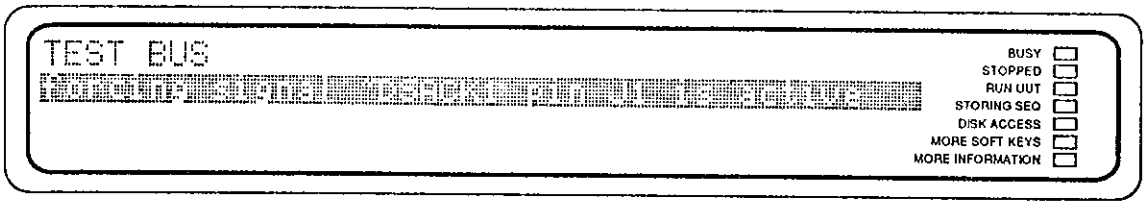
In example 4a, Bus Test again expected the two lines to both be low during the microprocessor reset address. In this case, the Pod sensed that a problem occurred with the reset address and A8, and a fault message was generated. However, since the correct reset address was not seen at ROM Module 1, no further diagnosis was possible (i.e., a fault message for A3 was not generated).

Example 5: $\overline{\text{DSACK0}}$ tied low (active) on a UUT with a 2 wait-state, 16-bit Boot ROM.

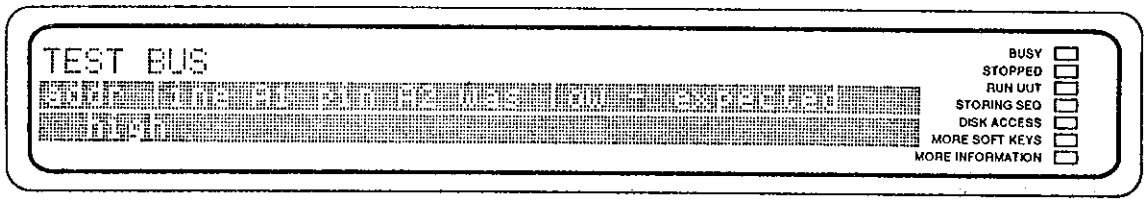
When TEST BUS is run, the following message is displayed:



If CONT is pressed, the following message is displayed:



If CONT is pressed again, the following message is displayed:



$\overline{\text{DSACK0}}$ stuck low is one of the more difficult faults to diagnose, since a continuously low $\overline{\text{DSACK0}}$ signal may not be a fault in some UUTs. In the UUT in this example, $\overline{\text{DSACK0}}$ should be high and $\overline{\text{DSACK1}}$ should be low during Boot ROM accesses, allowing the UUT to perform 16-bit fetches from the Boot ROM. TEST BUS recognizes that only two ROM Modules are in use and determines the expected levels. When $\overline{\text{DSACK0}}$ is found to be low, the first fault message is displayed.

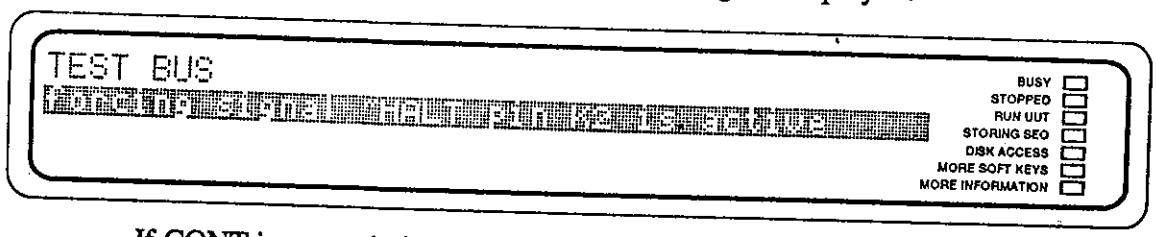
When CONT is pressed, $\overline{\text{DSACK1}}$ is tested. TEST BUS expects to find $\overline{\text{DSACK1}}$ low, but because $\overline{\text{DSACK0}}$ is tied low, the microprocessor's dynamic bus sizing mechanism determines that a 0 wait-state, 8-bit bus acknowledgment has occurred, and proceeds to terminate the bus cycle before $\overline{\text{DSACK1}}$ has gone active. TEST BUS then finds that $\overline{\text{DSACK1}}$ is high and reports the second fault message.

The second time CONT is pressed, TEST BUS checks the low-order address lines monitored by the ROM Modules. Since TEST BUS recognizes that a 16-bit bus is present, it expects addresses to increment two bytes at a time. But, since the microprocessor is reacting to $\overline{\text{DSACK0}}$ tied low and assumes that it has received an acknowledgment for only 8 bits of data, the addresses are incremented by only one address at a time. This causes TEST BUS to find an incorrect address, resulting in the third fault message.

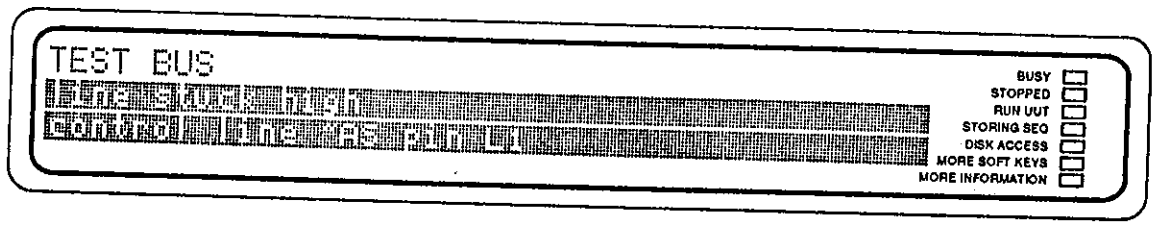
This example shows the hierarchical nature of TEST BUS and how faults may interact, resulting in extra fault messages.

Example 6: $\overline{\text{HALT}}$ tied low (active) on a UUT with 16-bit boot ROM.

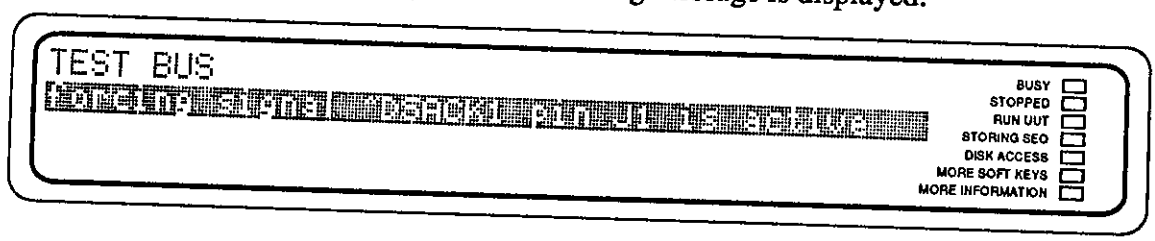
When TEST BUS is run, the following message is displayed:



If CONT is pressed, the following message is displayed:



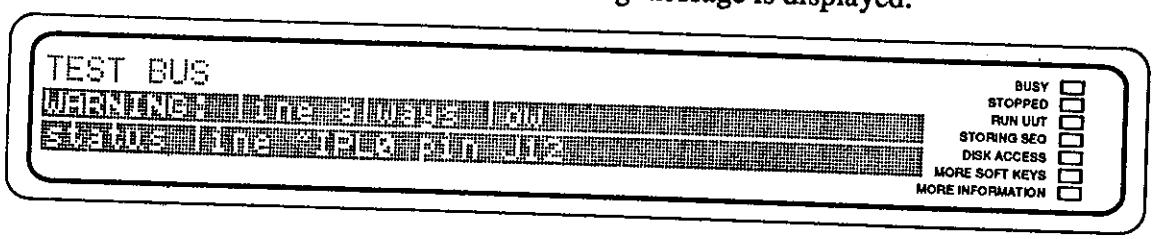
If CONT is pressed again, the following message is displayed:



This example demonstrates another case of multiple faults being detected by TEST BUS as a result of signal interaction on a UUT. The first fault displayed was correct, indicating a problem with the $\overline{\text{HALT}}$ signal. Because the microprocessor halted execution, activity on the $\overline{\text{AS}}$ line is missing. This is reported in the second fault message. Finally, because the processor is halted, activity on the $\overline{\text{DSACK1}}$ line is missing and is reported in the third fault message.

Example 7: $\overline{\text{IPL0}}$ tied low.

When TEST BUS is run, the following message is displayed:



This fault message indicates that the interrupt request input $\overline{\text{IPL0}}$ is active. Because this information by itself does not indicate a UUT fault, a "WARNING" is shown in the fault message. However, if you continue probing and find low levels on each of the three interrupt inputs ($\overline{\text{IPL0}}$, $\overline{\text{IPL1}}$ and $\overline{\text{IPL2}}$), then a non-maskable interrupt may be continuously asserted. In this case, TEST BUS may fail, requiring that the source of the non-maskable interrupt be checked for proper function.

NOTE

If a non-maskable interrupt is continuously or repeatedly asserted and this condition is normal for the UUT, measures should be taken to suppress the non-maskable interrupt request so that TEST BUS can run without interruption.

Using the Diagnose Bus Test 3-6.

To perform a manual diagnostic test of the UUT status and control lines, press the BUS key on the Mainframe and select DIAGNOSE (F2). The Diagnose Bus test differs from the diagnostic routines of Bus Test in that it requires you to manually probe all the lines to be tested. You are prompted to probe the clock, status lines, and control lines of the UUT microprocessor. If a fault is detected, a fault message is displayed. If no fault is detected, you are prompted to probe the next line.

If, for any reason, you choose to probe a different line than shown by the prompt, press the PROBE (F1) softkey. Since the Diagnose Bus test checks UUT signals in groups (i.e., status, control), the signal you select after pressing the PROBE key must fall within that group. Two softkeys, PREV and NEXT, allow you to scroll through the group of lines. To select the line you want to test, press the ENTER key.

To continue the Diagnose Bus test once a fault message is displayed, press CONT on the Mainframe keypad.

To exit from the Diagnose Bus test, press STOP.

Using the Stimulus Routines 3-7.

STIM_DAT and STIM_ADR are two basic stimulus routines used in the Pod Bus Test. Using the Mainframe LOOP key, these stimulus routines allow you to loop on Bus Test faults.

STIM_DAT is the most basic stimulus available to use in the presence of UUT kernel faults. This program causes the UUT microprocessor to be reset and attempts to fetch the data you have specified over the data bus. Along with this fetch, a sync pulse is generated (either address or data sync, set by the Mainframe SYNC function). The STIM_DAT stimulus routine is useful in the following cases:

- Bad reset address.

  This TEST BUS fault is generated because the ROM Module did not see the correct reset address. Perform a looping STIM_DAT with the probe synced to "pod address". Probe the address bus and associated buffers to determine the cause of the problem.

- No chip select detected at ROM Module 1.

As in the previous case, a looping STIM_DAT can be used with the probe to examine the ROM address decoder lines to determine the problem.

- Data bus faults.

  To troubleshoot data bus faults, perform a looping STIM_DAT with the probe synced to "pod data". The fault can be traced from the UUT microprocessor back through the data buffers.

STIM_ADR requires more of the kernel to be working than does STIM_DAT. For STIM_ADR to work correctly, the data bus and address lines A0 through A4 must be functional. Once these conditions are met, STIM_ADR can place any arbitrary address on the address bus and generate a sync pulse simultaneously. To use STIM_ADR to troubleshoot address line faults, a looping STIM_ADR is performed with the probe synchronized to "pod addr".

To run either STIM_DAT or STIM ADR, press the POD key on the Mainframe keypad, select either STIM_DAT (F4) or STIM_ADR (F3), and press ENTER.

If you select STIM_DAT, enter the 32-bit data pattern you want placed at the reset address of the UUT microprocessor and press ENTER. Use the Mainframe SYNC key to select the type of synchronization pulse needed.

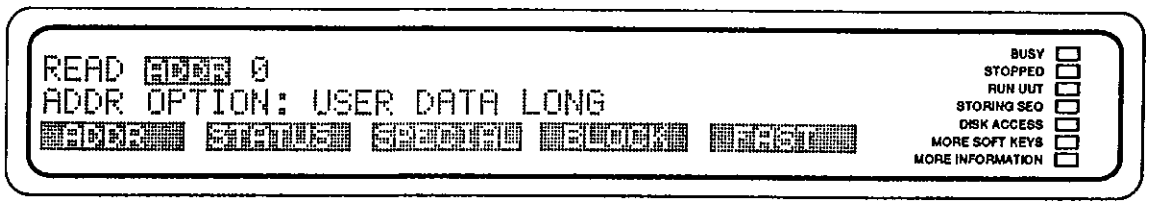
If you select STIM_ADR, enter the 32-bit address where you want the UUT microprocessor to perform a fetch. This address must be within the UUT boot ROM address space.

READ AND WRITE OPERATIONS 3-8.

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 and to fill blocks of memory with data.

Read Operations 3-9.

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



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

READ AT UUT ADDRESS 3-10.

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 a multiple of 4 between address 0000 0000 and FFFF FFFC (for a longword 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 STATUS 3-11.

The Read Status function is not implemented on the 9132A Memory Interface Pod. Though a value is returned when the Mainframe softkey is pressed, this number does not reflect the actual status of the UUT processor.

READ POD SPECIAL ADDRESS 3-12.

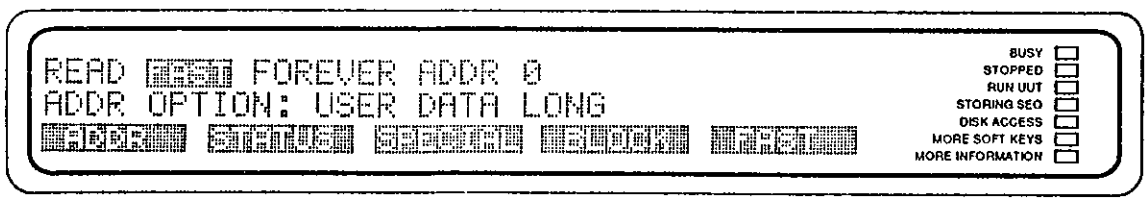
Because the 68020 microprocessor uses 32-bit addresses, the Special Address command cannot be used to read the contents of the Pod's special addresses. To read these addresses, use the VIRTUAL softkey command described further on in this manual.

READ BLOCK 3-13.

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 3-14.

The Pod allows you to perform repeated reads at an address (for example, to check the operation 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:



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.

For a faster repetition rate, use the Quick Looping Read pod function.

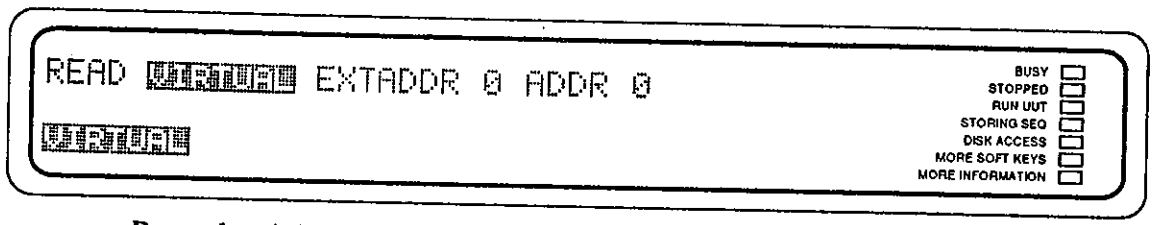
READ VIRTUAL ADDRESS

3-15.

NOTE

Pod special (virtual) addresses are only meaningful when troubleshooting the Pod. Reads from special addresses are not needed in normal 9100-Series operation.

To read the contents of the 68020 Pod's virtual addresses, press the READ key, press SOFT KEYS, and select the VIRTUAL (F1) softkey. The Mainframe displays the following message:



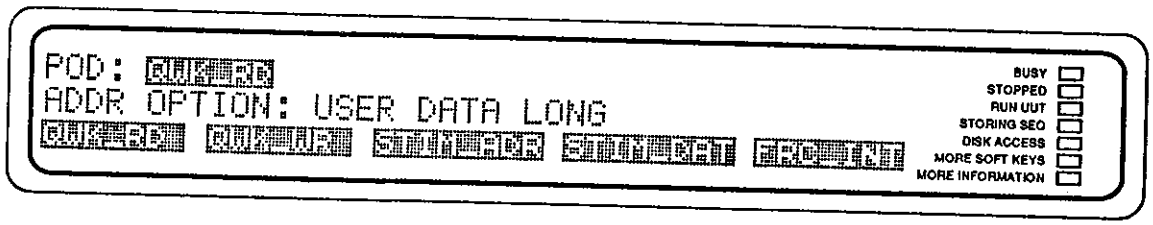
Press the right arrow key (→) to move the cursor and enter the value of the extended address. Press the right arrow key again and enter the value of the virtual address. Press ENTER on the keypad. The data from the special address is displayed as an 8-digit hex value (with leading zeros suppressed). A complete list of available virtual addresses is located in Appendix E.

QUICK LOOPING READ

3-16.

A Quick Looping Read function is available on the 68020 Pod. This function has a faster repetition rate than the ordinary Read Fast function. Because of the increased repetition rate, this function is particularly suited for enhanced viewing of signal traces on an oscilloscope.

To perform the Quick Looping Read function, press the Pod key on the Mainframe keypad. The following menu is displayed:



Select QWK_RD (F1) and press ENTER. Select the UUT address for the read operation and press ENTER. The Mainframe displays the data found at that address.

Unlike other read functions, the routines that control the Quick Looping Read function reside in the Pod, not in the Mainframe. After the Mainframe begins execution of Quick Looping Read, the Pod reports data or fault messages to the Mainframe only once, then continues to send reads to the specified address.

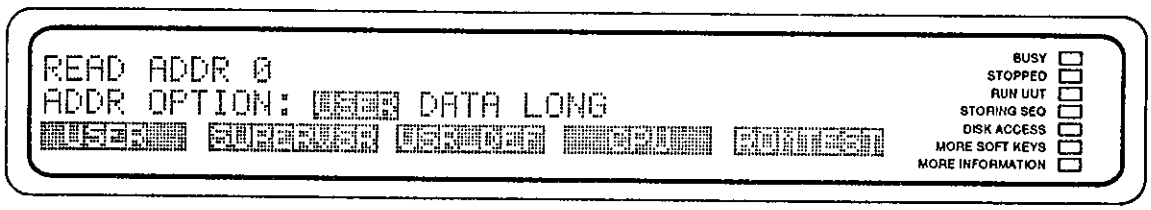
Continuous error reporting with the Quick Looping Read function can be achieved by pressing the Mainframe LOOP key. The Mainframe then commands read operations with full error reporting. For every ordinary read operation, the Pod interjects a few Quick Looping Read operations (with no error reporting) to enhance oscilloscope viewing.

NOTE

Looping on the QWK_RD function may result in the loss of UUT hardware initialization. See Appendix G for more information..

READ ADDRESS OPTIONS 3-17.

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:



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 twenty-one different settings. The address options correspond to address spaces defined by the manufacturer, plus UUT_ROM, ST_ROM, and Overaly. Within each address space, separate options are provided for each possible data size (byte, word, and long word). The address options are:

- User data long, word, or byte
- User program long, word, or byte
- Supervisor data long, word, or byte
- Supervisor program long, word, or byte
- User-defined long, word, or byte
- CPU long, word, or byte
- UUT_ROM
- ST_ROM
- Overlay

Description of 68020 Address Options 3-18.

User data long, word, or byte accesses UUT memory that contains user program data. To select user data, press the USER (F1) softkey, move the cursor by pressing the right arrow key (→), press the DATA (F1) softkey, move the cursor to the right, and select either LONG (F1), WORD (F2), or BYTE (F3).

User program long, word, or byte accesses UUT memory that contains user program instructions. To select user program, press the USER (F1) softkey, move the cursor by pressing the right arrow key (→), press the PROGRAM (F2) softkey, move the cursor to the right, and select either LONG (F1), WORD (F2), or BYTE (F3).

Supervisor data long, word, or byte accesses UUT memory that contains supervisor program data. To select supervisor data, press the SUPERVSR (F2) softkey, move the cursor by pressing the right arrow key (→), press the DATA (F1) softkey, move the cursor to the right, and select either LONG (F1), WORD (F2), or BYTE (F3).

Supervisor program long, word, or byte accesses UUT memory that contains supervisor program instructions. To select supervisor program, press the SUPERVSR (F2) softkey, move the cursor by pressing the right arrow key (→), press the PROGRAM (F2) softkey, move the cursor to the right, and select either LONG (F1), WORD (F2), or BYTE (F3).

User-defined long, word, or byte accesses UUT memory that has been assigned to specific functions by the user. To access the user-defined area, press USR_DEF (F3), move the cursor by pressing the right arrow key (→), and select either LONG (F1), WORD (F2), or BYTE (F3).

CPU long, word, or byte accesses UUT addresses that are reserved for processor functions. To access the CPU area, press CPU (F4), move the cursor by pressing the right arrow key (→), and select either LONG (F1), WORD (F2), or BYTE (F3).

The UUT_ROM option reads a byte of data directly from the UUT boot ROM inserted in the ROM Module socket while the ROM Module is plugged into the UUT. If the address selected is outside the range of the UUT boot ROM, a memory byte operation at the address indicated is performed. To select the UUT_ROM option, press the UUT_ROM (F3) softkey.

The ST_ROM option reads a byte of data from the UUT boot ROM inserted in the ROM Module socket while the ROM Module is plugged into the Pod self test socket. During this operation, the address specified in the top line of the Mainframe display must be within the range of 0 to (size of ROM in the ROM Module minus 1). To select the ST_ROM option, press the ST_ROM (F4) softkey.

The overlay option reads a byte of data from the Pod Overlay memory. During this test, the address specified in the top line of the Mainframe display must be within the bottom (8K x number of ROMs) bytes of UUT

memory or the address range of the UUT boot ROMs, whichever is smaller. To select the overlay option, press the OVERLAY (F5) softkey.

See Table 3-2 for a list of 68020 function codes corresponding to each of the address option spaces.

Address Option and 68020 Dynamic Bus Sizing 3-19.

Because of the dynamic bus sizing feature of the 68020, any size data can be transferred on any size data bus. If more data must be transferred than fits on the UUT bus at one time, then additional bus cycles are executed until all data is transferred.

When longword data is transferred on a 32-bit wide UUT data bus, data lines D0 through D31 are used, with the least significant bit (LSB) located on D0. If the longword is transferred on a 16-bit data bus (in two separate cycles), the transfer occurs on data lines D16 through D31 with the LSB on D16. If the longword is transferred on an 8-bit data bus (in four cycles), the transfer occurs on data lines D24 through D31 with the LSB on D24.

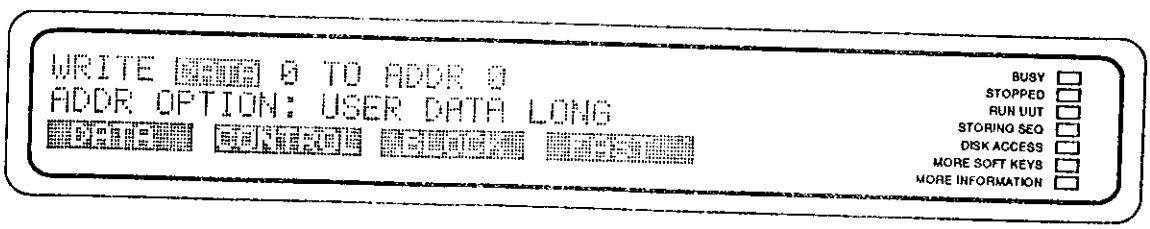
When word data is transferred on a 32-bit data bus, the data is either transferred on D16 through D31 (if A1 is 0) or D0 through D15 (if A1 is 1). On a 16-bit bus, the data is transferred on D16 through D31. On an 8-bit bus, the data is transferred (in two cycles) on D24 through D31. In each case, the LSB is on the lowest numbered data line used.

When byte data is transferred on a 32-bit bus, the data either appears on D24 through D31 if (A1, A0) is (0, 0), D16 through D23 if (A1, A0) is (0, 1), D8 through D15 if (A1, A0) is (1, 0), or D0 through D7 if (A1, A0) is (1, 1). For a 16-bit bus, the data is transferred on D24 through D31 if A0 is 0 or D16 through D23 if A0 is 1. For an 8-bit bus, the data is transferred on D24 through D31.

The dynamic sizing feature of the 68020 microprocessor allows transfers of any size data at any address location (for example, word-wide data can be located at odd addresses). During testing (with the exception of RUN UUT), the 9132A Pod restricts address alignment of the different data types. Longword data can only be accessed at addresses divisible by 4 and word data can only be accessed at addresses divisible by 2. (Byte data can be accessed at any address.)

Write Operations 3-20.

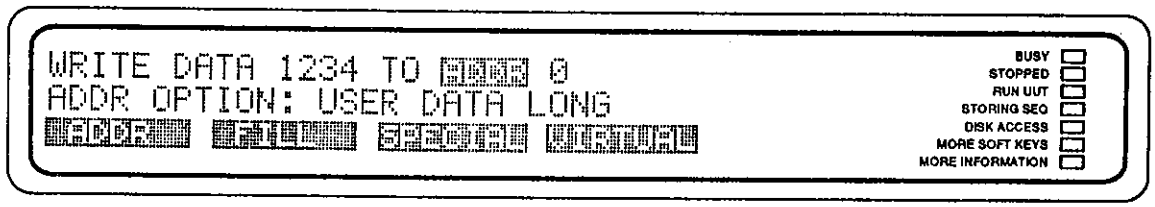
To enter the Write menu of the Mainframe, press WRITE on the keypad. The Mainframe displays the following message:



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), and check the accuracy of the address and data lines.

WRITE UUT ADDRESS 3-21.

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:



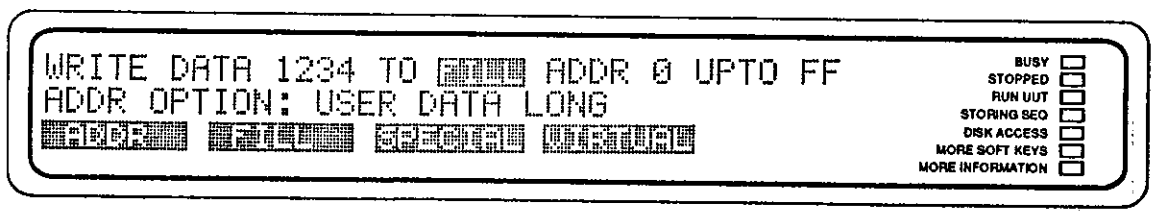
NOTE

"1234" represents 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 0000 0000 and FFFF FFFC (for a longword access). Press ENTER to write to the address you selected.

FILL MEMORY AREA 3-22.

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:



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 3-23.

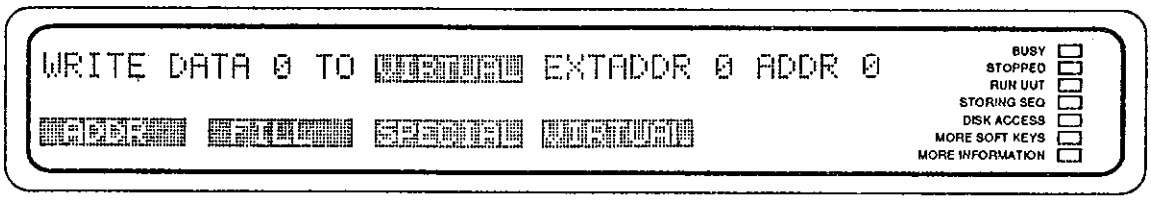
Because the 68020 microprocessor uses 32-bit addresses, the Special Address command cannot be used to write data to the Pod's special addresses. To write to these addresses, use the VIRTUAL softkey command.

WRITE VIRTUAL ADDRESS 3-24.

NOTE

Pod special (virtual) addresses are only meaningful when troubleshooting the Pod. Writes to special addresses are not needed in normal 9100-Series operation.

To write to a Pod virtual 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 VIRTUAL (F4). The Mainframe displays the following message:



Press the right arrow key (→) to move the cursor and enter the value of the extended address. Press the right arrow key again and enter the value of the virtual address. Press ENTER on the keypad. A complete list of available virtual addresses is located in Appendix E.

WRITE CONTROL 3-25.

The Write Control function is not implemented on the 9132A Memory Interface Pod. Though the Mainframe and Pod accept the entered value, no control lines on the UUT microprocessor are affected.

WRITE BLOCK 3-26.

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 3-27.

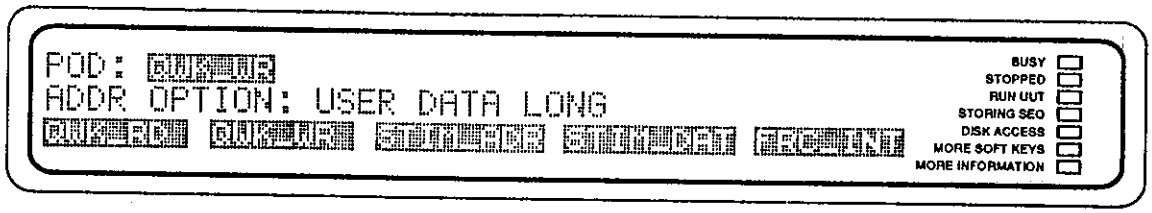
The Pod allows you to perform repeated writes to an address (to check the accuracy of the address or data bus lines with an oscilloscope, for example). Press the WRITE key and select the FAST (F4) softkey. The Mainframe displays the following message:

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.

For a faster repetition rate, use the Quick Looping Write function.

QUICK LOOPING WRITE 3-28.

To perform the Quick Looping Write function, press the Pod key on the Mainframe keypad. The following menu is displayed:



Select QWK_WR (F2) and press ENTER. Select the UUT address for the write operation, move the cursor to the right, select the data to be written, and press ENTER. The Mainframe then writes the data you entered at the specified address.

Unlike other write functions, the routines that control the Quick Looping Write function reside in the Pod, not in the Mainframe. After the Mainframe begins execution of Quick Looping Write, the Pod reports fault messages to the Mainframe only once, then continues to write data to the specified address.

Continuous error reporting with the Quick Looping Write function can be achieved by pressing the Mainframe LOOP key. The Mainframe then commands write operations with full error reporting. For every ordinary write operation, the Pod interjects a few Quick Looping Write operations (with no error reporting) to enhance oscilloscope viewing.

NOTE

Looping on the QWK_WR function may result in the loss of UUT hardware initialization. See Appendix G for more information..

WRITE ADDRESS OPTIONS 3-29.

The address option line of the Write function operates in the same manner as the address option line of the Read function (except writes are performed).

NOTE

The UUT_ROM and ST_ROM options do not function during write operations.

For more information, see the heading "Read Address Options" earlier in this section.

TESTING THE UUT RAM 3-30.

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. HyperRAM and RAM Fast are shorter and faster tests than RAM Full.

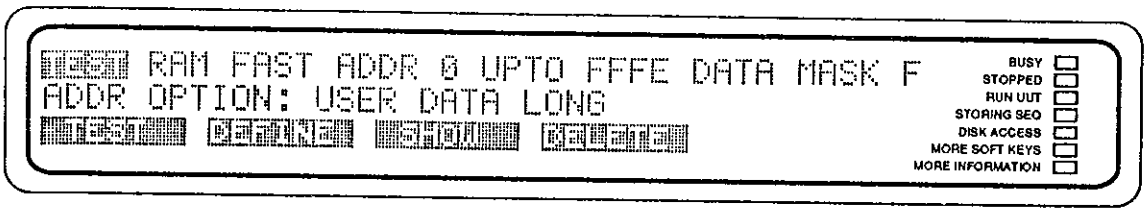
RAM fault coverage is essentially the same for the RAM Fast and HyperRAM tests. The difference is that the HyperRAM test is completed in much less time.

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

RAM Fast Test 3-31.

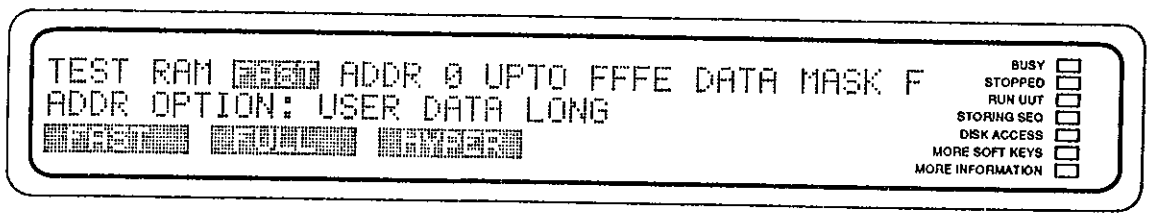
RAM Fast test is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable. The following steps describe how to select the RAM Fast test.

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



2. Press the down arrow key (↓) and select the address option. Press the up arrow key (↑) to return to the first line of the display.

3. 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 RAM Fast test, press F1 on the Mainframe keypad. The Mainframe displays the following menu:



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

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

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

7. 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. To test all the data in a longword address, enter FFFF FFFF into the field.

8. 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. If you are testing longword addresses, change the setting to 4.

9. 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).

10. 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.

11. 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 Mainframe display.

RAM Full Test 3-32.

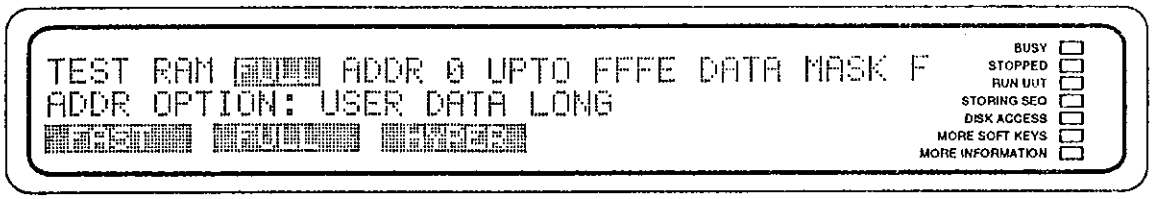
The RAM Full test is the most comprehensive RAM test algorithm available in the 68020 Pod. Besides the fault types detected by other algorithms available in the Pod, the RAM Full test makes several additional passes through memory to help find coupling faults. The following steps describe how to select the RAM Full test.

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

2. Press the down arrow key (↓) and select the address option. Press the up arrow key (↑) to return to the first line of the display.

3. 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

RAM Full test, press F2 on the Mainframe keypad. The Mainframe displays the following RAM Full test menu:



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

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

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

7. 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. To test all the data in a longword address, enter FFFF FFFF into the field.

8. 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. If you are testing longword addresses, change the setting to 4.

9. 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).

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 Mainframe display.

HyperRAM Test 3-33.

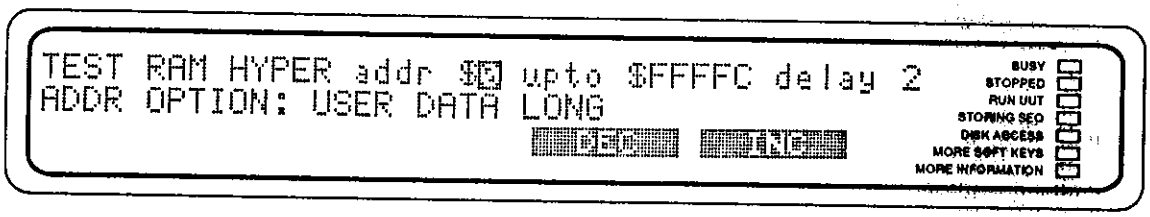
HyperRAM is the fastest RAM test algorithm available in the 68020 Pod. It is designed to quickly identify common RAM failures such as address decoding errors or bits that are not read/writable.

The following steps describe how to select the HyperRAM test.

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

2. Press the down arrow key (↓) and select the address option. Press the up arrow key (↑) to return to the first line of the display.

3. 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:



4. Enter the starting address. The valid address range is the top of boot ROM space + 4 to FFFF FFF8 (longword space), the top of boot ROM space + 2 to FFFF FFFC (word space), or the top of boot ROM space + 1 to FFFF FFFE (byte space).

5. Press the right arrow key again to move the cursor to the ending address field. Key in the ending address. The valid ending address range is the top of boot ROM space + 8 to FFFF FFFC (longword space), the top of boot ROM space + 4 to FFFF FFFE (word space), or the top of boot ROM space + 2 to FFFF FFFF (byte space). The ending address must be greater than the starting address.

6. 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 65535 (the default is 250).

7. 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.

8. 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 4 when the address option is set to long, at 2 when the address option is set to word, and at 1 when the address option is set to byte.

The Pod can perform the HyperRAM test with any address option except CPU space. Normally RAM devices are present only in DATA spaces, but the HyperRAM test allows you to test PROGRAM space as well. (HyperRAM is optimized for SUPERVSR DATA space, executing up to twice as fast as in other address spaces.)

For the most rapid HyperRAM test, select the LONG address option. This selection can be used regardless of the physical bus size. LONG access causes one access for data on a 32-bit data bus, two sequential accesses on a 16-bit data bus, and four sequential accesses on an 8-bit data bus.

Occasionally HyperRAM generates a fault message that does not identify a specific failure, but instead reports that expected data and actual data read at a specified address are the same value. This is possible because the HyperRAM test algorithm performs rapid memory-direct compare operations. When a comparison fails, the failing data is not yet stored in a register, but must be read a second time before it can be reported. In a marginal UUT situation, it is possible the original expected data is read the second time.

When the HyperRAM fault message reports that actual data is equal to the expected data, the RAM at the address indicated did fail. In this case, look for marginal UUT conditions such as:

- Noise.
- Power supply level out of tolerance.
- Inadequate timing margins.

TESTING THE UUT ROM 3-34.

ROM Test computes and tests the ROM signatures gathered either from a UUT boot ROM located in the ROM Module or other UUT ROMs located on the UUT, and compares the signature to a previously stored 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 the address range of a ROM produces a signature that characterizes that range. The idea of using signatures is to obtain them from data in a known-good ROM, and then compare the good values with values obtained from the suspect ROM.

Signature Gathering from UUT Boot ROMs 3-35.

ROM MODULE PLUGGED INTO THE SELF TEST SOCKET 3-36.

To obtain the UUT boot ROM signature from a ROM Module that is plugged into the self test socket, perform the following procedure:

1. Unplug ROM Module 1 from the UUT.
2. Plug a known-good ROM into the ROM socket of ROM Module 1.
3. Open the self test socket door on the right-hand side of the Pod.
4. Insert the ROM Module cable into the ZIF self test socket. Secure the ROM Module cable plug in place by pushing down the latch lever.
5. Select the ROM Test by pressing ROM on the Mainframe keypad. The Mainframe displays the ROM signature gathering menu:

6. Press the down arrow key (↓) and select the ST_ROM (F4) address option. Press the up arrow key (↑) to return to the first line of the display.
7. 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.
8. 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.
9. 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 all data bits, enter FF into the data mask field.

NOTE

While the address option is set to ST_ROM, the data mask only accepts a byte-wide mask.

10. Press the right arrow key again to move the cursor to the address step field. Key in the step number (normally 1 to test all ROM locations).

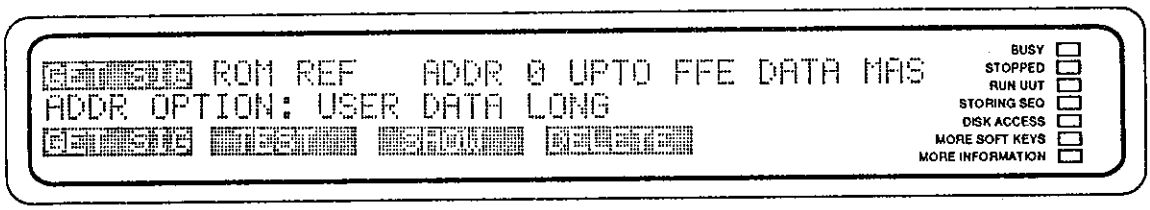
11. Press ENTER to begin the signature gathering routine.

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.

ROM MODULE PLUGGED INTO THE UUT 3-37.

To obtain the UUT boot ROM signature from a ROM Module that is plugged into the UUT, perform the following procedure:

1. Turn the UUT power OFF.
2. Plug a known-good ROM into the ROM socket of ROM Module 1.
3. Turn the UUT power ON.
4. Select the ROM Test by pressing ROM on the Mainframe keypad. The Mainframe displays the ROM signature gathering menu:



5. Press the down arrow key (↓) and select the UUT_ROM (F3) address option. Press the up arrow key (↑) to return to the first line of the display.
6. 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.
7. 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.
8. 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 all data bits, enter FF into the data mask field.

NOTE

While the address option is set to UUT_ROM, the data mask only accepts a byte-wide mask.

9. Press the right arrow key again to move the cursor to the address step field. Key in the step number (normally 1 to test all ROM locations).

10. 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.

UUT BOOT ROMS SOLDERED TO THE UUT 3-38.

To obtain the ROM signature of UUT boot ROM soldered onto the UUT, the Pod must be able to disable the boot ROM. For information on testing UUTs with soldered UUT boot ROMs, see Appendix C. If the Pod is capable of overriding the UUT boot ROM, use the method for gathering signatures from other UUT ROMs to obtain the signature for the soldered-in boot ROMs.

Signature Gathering from Other UUT ROMs 3-39.

To obtain the ROM signature of ROM located on the UUT (not including boot ROM), 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.

3. Press the down arrow key (↓) and select the address option. Move the cursor to the right and select the width of the ROM data bus. Press the up arrow key (↑) to return to the first line of the display.

4. 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.

5. 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.

6. 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. To test all the data in a longword address, enter FFFF FFFF 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. If you are testing longword addresses, change the setting to 4.

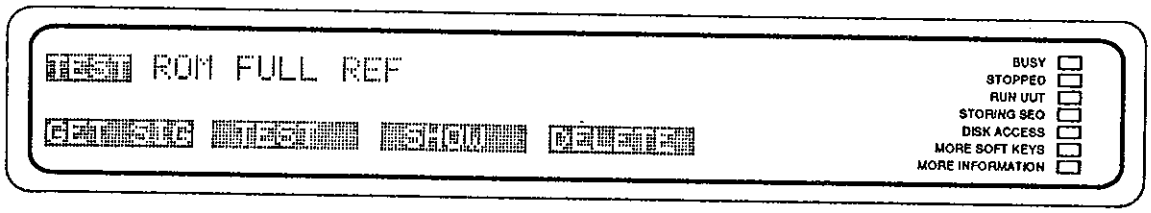
8. Press ENTER to begin the signature gathering routine.

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.

Signature Testing 3-40.

A ROM Test can be performed after the signature is obtained. The signature test must be performed using the same method as described under signature gathering. For example, if the signature was obtained with the UUT ROMs in the Self Test socket, the same method must be used to test the signature of another UUT ROM.

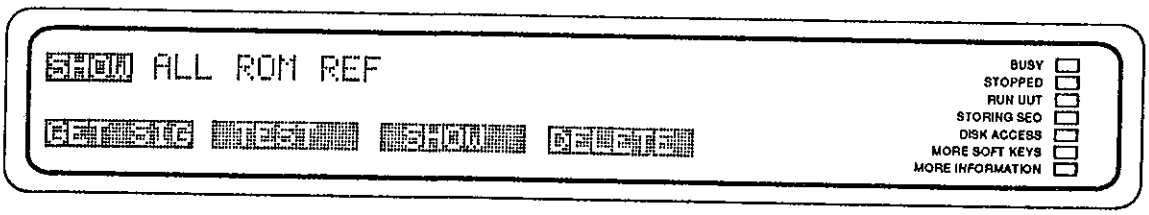
Select the ROM Test by pressing ROM on the Mainframe keypad. Move the cursor to the leftmost field using the left arrow key (←). Press F2 on the Mainframe keypad to select the ROM test. The Mainframe now displays the ROM Test menu:



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 enter 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.

Obtaining a List of Signatures 3-41.

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:

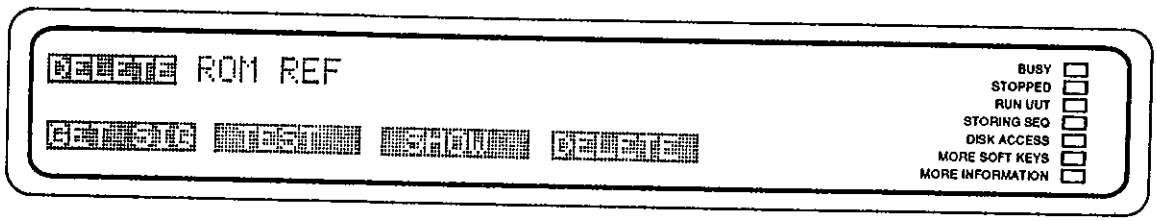


Press ENTER. If any signatures have been previously stored, the list of signature files is shown on the Mainframe display. If the MORE INFORMATION indicator on the display is lit, you can scroll though the file by pressing the down arrow (↓) to move down or the up arrow (↑) to move up.

Deleting a Signature File

3-42.

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:



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.

USING THE RUN UUT MODE

3-43.

RUN UUT allows you to run the UUT software located in the boot ROMs. The UUT boot ROMs must be plugged into the ROM Module sockets as described in Section 2.

The starting address of RUN UUT may be either the default address (0 for the 68020), or any address located from 0 to FFFF FFFE. To begin RUN UUT at the default address, press the RUN UUT key on the Mainframe keypad. Select STARTING (F2). Move the cursor to the right and enter 0. Press ENTER to begin RUN UUT. To begin RUN UUT at an address other than the default address, press the RUN UUT key, select STARTING (F2), move the cursor to the right, and enter the starting address (within the address limitations listed above). If you specify a size of LONG on the address option line of the Mainframe, the starting address must be a multiple of 4. If you specify WORD, the starting address must be a multiple of 2.

The starting address for RUN UUT does not need to be located within the UUT boot ROM space or the Overlay Memory space. If UUT RAM is located within the address space specified for RUN UUT, a test program can be loaded into UUT memory (using WRITE BLOCK) and can be executed using RUN UUT.

RUN UUT runs with any address option specified other than CPU. However, all RUN UUT operations place the processor in the Supervisor mode. UUT hardware determines data size, and RUN UUT software determines operating modes.

To halt RUN UUT, press the RUN UUT key on the Mainframe keypad. Select HALT (F3). Press ENTER to halt RUN UUT.

NOTE

The Pod performs a UUT reset after exit from RUN UUT. In some cases, this reset may cause UUT hardware initialization to be lost. If this occurs, repeat the hardware initialization sequence after exiting RUN UUT to continue testing.

RUN UUT Special 3-44.

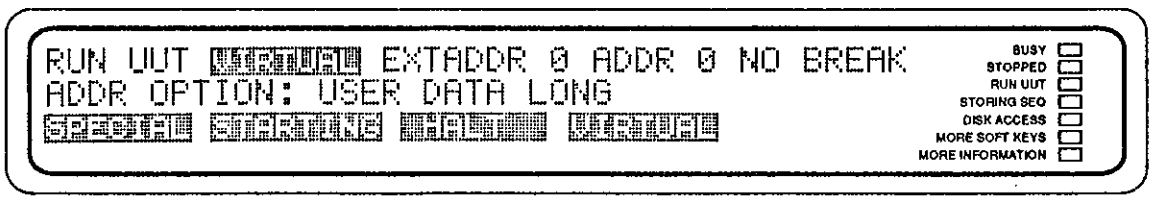
Because the 68020 microprocessor uses 32-bit addressing, the Special Address command cannot be used to RUN UUT at the Pod's special addresses. To RUN UUT at these addresses, use the VIRTUAL softkey command.

RUN UUT Virtual 3-45.

NOTE

Pod RUN UUT special (virtual) addresses are only meaningful when troubleshooting the Pod. RUN UUT at special addresses are not needed in normal 9100-Series operation.

To RUN UUT at a UUT virtual address, press the RUN UUT key and select VIRTUAL (F4). The Mainframe displays the following message:



Press the right arrow key (→) to move the cursor and enter the value of the extended address. Press the right arrow key again and enter the value of the virtual address. Press ENTER on the keypad. A complete list of available RUN UUT virtual addresses is located in Appendix E.

USING BREAKPOINTS 3-46.

The 68020 Pod implements hardware breakpoints as an optional means of terminating the RUN UUT operation in Overlay Memory. A "break" occurs in the RUN UUT operation when the address on the UUT address bus matches the user-specified break address. Once the break occurs, control is transferred to the Mainframe.

Enabling Breakpoints

3-47.

Breaks are enabled by entering the RUN UUT menu, moving the cursor to the Break/No Break field, and selecting BREAK (F2). Next, move the cursor to the address option field using the down arrow key, press SOFTKEYS, and select OVERLAY (F2). To disable breakpoints, enter the RUN UUT menu, move the cursor to the Break/No Break field, and select 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

3-48.

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.

As soon as the physical address of the 68020 matches the contents of the RUN UUT break address field (and breaks are enabled), the break occurs. Table 3-1 shows which microprocessor address bits are checked for breakpoints. All address lines not listed in Table 3-1 are ignored.

Table 3-1. Address Bits Checked for Breakpoints

ROM SIZE	1 ROM	2 ROMS	4 ROMS
2K x 8	A1 - A10	A1 - A11	A2 - A12
4K x 8	A1 - A11	A1 - A12	A2 - A13
8K x 8	A1 - A12	A1 - A13	A2 - A14
16K x 8	A1 - A13	A1 - A14	A2 - A15
32K x 8	A1 - A14	A1 - A15	A2 - A16
64K x 8*	A1 - A15	A1 - A16	A2 - A17
* The last line refers to all ROMs that are 64K x 8 or larger.			

The breakpoint address can only be set to the boot ROM address range. If the address is outside the boot ROM address range, the breakpoint does not occur.

NOTE

Because of the method used by the Pod to implement breakpoints, addresses 0 through 7 should not be set as the breakpoint address. If any of these addresses are set as the breakpoint address, an accidental break may occur.

Once a break occurs, the Pod resets the UUT, which then executes a standby loop at the reset address. The 68020 register contents before the break are lost and cannot be recovered. The contents of the Overlay Memory 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 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.

USING OVERLAY MEMORY 3-49.

The Pod is equipped with memory space that can be used to run programs written on external development systems. These programs can be downloaded to the Mainframe disk, loaded to the Pod Overlay Memory, and then run using the RUN UUT mode.

Whenever the Overlay Memory is selected, a read within the Overlay Memory address range returns the data in the Overlay Memory rather than the UUT boot ROMs (located in the ROM Module sockets).

NOTE

The maximum size of a program loaded into Overlay Memory (for each ROM Module) is 8K bytes or the size of the boot ROM, whichever is less.

The first step in downloading programs is to move the program from the development system to the disk of the Mainframe. The 9100-Series Mainframe contains a terminal emulator that communicates over an RS-232-C port. A description of the use of the terminal emulator and directions on how to download programs to the Mainframe disk drive is located in Section 6 of the 9100-Series Programmer's Manual. To properly write the block to memory, you must locate the program in a Mainframe text file.

Selecting Overlay Memory 3-50.

When power is applied to the Pod, the Overlay Memory is disabled (default) and does not interact with the UUT in any way (for example, the Pod does not perform a RUN UUT out of the Overlay Memory until the Overlay Memory is selected as the address option). To select the Overlay Memory, move the cursor on the Mainframe display to the address option line and select OVERLAY (F5).

To discontinue using Overlay Memory, move the Mainframe cursor to the address option line and select any option except OVERLAY.

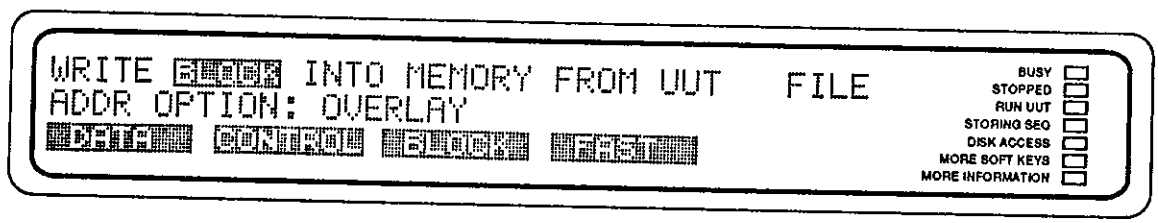
NOTE

The program that was stored in Overlay Memory remains until it is overwritten by another program or power is removed from the Pod.

Moving the Program From Disk to Overlay Memory 3-51.

Once Overlay Memory is selected, a program that is located on a disk can be moved to Overlay Memory in the Pod. The following steps describe how to copy the program to Overlay Memory:

1. Press the WRITE key on the Mainframe keypad. Select the BLOCK (F3) softkey. The Mainframe displays the following menu:



2. Move the cursor to the right by pressing the right arrow key (→). Type in the location of the file on the Mainframe.

3. Move the cursor to the right and type in the name of the file in which the program was saved.

4. Move the cursor to the right and select the format of the program you previously downloaded. Select Motorola™ (F1) format.

5. Move the cursor to the right and enter any offset from the address of the downloaded program to the address used to run the program in UUT. In most cases, the offset should be 0.

6. Press ENTER.

Once the program has been loaded into Overlay Memory, use RUN UUT to start the program. The RUN UUT address can either be the default reset address or any address within the address limit of the UUT boot ROM space or 8K bytes times the width of the boot ROM (in bytes), whichever is less.

PROBE AND SCOPE SYNCHRONIZATION MODES 3-52.

You may use the Mainframe's Synchronization function (selected with the SYNC key) to choose the sync pulse sent by the Pod to the Mainframe for use by the probe and I/O Module. The chosen sync pulse is also available in

an isolated (and slightly delayed) version at the TRIGGER OUTPUT connector on the back of the Mainframe. The three synchronization modes that are available and their Mainframe selection codes are:

ADDR = Address Sync
DATA = Data Sync
FREERUN = Free-Run Sync (Mainframe generated)

The 68020's dynamic bus sizing allows accesses to occur where the data transferred is larger than the data bus it is transferred on by performing multiple bus cycles. If this type of access is performed, only one sync pulse output is generated and is only valid for the first bus cycle.

Address Sync

3-53.

If you select the address sync mode, the sync pulse used by the probe and I/O Module (and the related TRIGGER OUTPUT) are synchronized to the address portion of the UUT access. The sync pulse goes low at the end of the previous address cycle, and goes high at the end of the address portion of the UUT access cycle (corresponding to $\overline{AS}$). The probe and I/O Module latch their data on the second or high-going edge of the sync pulse.

Address Sync corresponds to the $\overline{AS}$ signal timing on the 68020 processor. The leading edge begins prior to the bus access and the trailing edge is derived from the trailing edge of $\overline{AS}$.

Data Sync

3-54.

If you select data sync mode, the sync pulse used by the probe and I/O Module (and the related TRIGGER OUTPUT) are synchronized to the data portion of the UUT access. The sync pulse goes low at the end of the previous data cycle, and goes high at the end of the data portion of the UUT access cycle. The probe or I/O Module latch their data on the second or high-going edge of the sync pulse.

Data Sync also corresponds to the $\overline{AS}$ signal timing on the 68020. (The processor's $\overline{AS}$ timing is identical to the $\overline{DS}$ timing.) The leading edge begins prior to the bus access and the trailing edge is derived from the trailing edge of $\overline{AS}$.

Data sync mode can also be used to capture activity on the bus during an interrupt acknowledge cycle.

Free-Run Sync

3-55.

If you select the free-run sync mode, the probe and I/O Module inputs are asynchronous. Stimulus pulses, if enabled, are generated with a frequency of approximately 1 kHz and a duty cycle of 1%. The 1 kHz clock signal is generated within the Mainframe and is unrelated to Pod timing signals.

At power-on, the probe and I/O Module are in the free-run mode, and their outputs are off.

68020 Pod Sync Timing Description and Suggestions 3-56.

The timing diagram in Figure 3-1 shows possible timing variations for different 68020 accesses. The W/R line has no effect on 9132A sync pulse. Write and read operations are shown only to provide a reference for the data bus timing in each cycle shown.

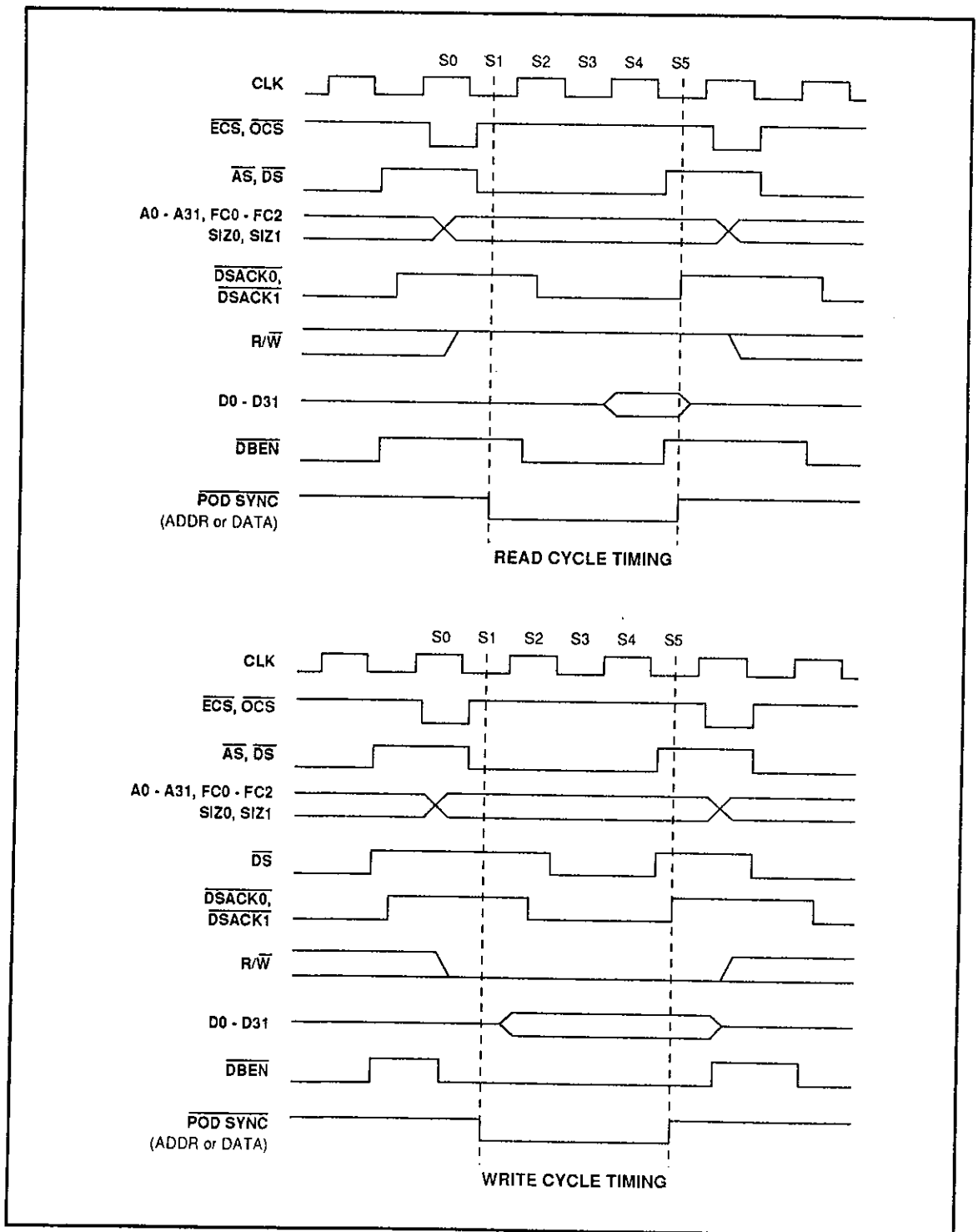


Figure 3-1. 68020 Pod Timing Diagram

INTERRUPTS

3-57.

Interrupts on the 68020 are initiated by a negative logic level interrupt code on lines $\overline{IPL0}$ through $\overline{IPL2}$ on the processor. During UUT testing, the interrupt lines are masked to prevent the processor from being interrupted.

However, level 7 interrupts are non-maskable. If a level 7 interrupt occurs, the operation in progress is aborted and a fault message displayed indicating the interrupt occurred. If the level 7 interrupt is continuous (the $\overline{ILPx}$ lines do not change state), Pod operation can be continued and further tests conducted to determine the UUT fault.

If a level 7 interrupt occurs during Bus Test, the test continues until complete, then reports that the test has failed and begins the diagnostic procedures.

During RUN UUT, interrupts may be enabled by the UUT program contained in the UUT boot ROMs located in the ROM Modules or within the program in Overlay Memory. If interrupts are enabled by the UUT program during RUN UUT, the UUT program must contain an interrupt service routine to receive control once an interrupt occurs.

For information on simulating an interrupt acknowledge, see Simulating Interrupt Acknowledge further on in this section.

USING THE 68020 CPU SPACE

3-58.

The UUT's floating point processor (FPU) and memory management unit (MMU) can be accessed by the Pod using the CPU address option (which sets FC0, FC1, and FC2 on the processor to one). (For information about the address options, see the heading "Read Address Options" earlier in this section.) The CPU address option also allows you to simulate interrupt acknowledge cycles and the microprocessor's breakpoint acknowledge cycle.

Communicating with Coprocessors

3-59.

MMU or FPU coprocessors can be accessed through read and write operations with the address option set to CPU space. The coprocessor ID and register are specified through encoded address line fields. The addresses used are 0002 X0YY, where X (2 through E by twos) is the coprocessor ID number and YY (0 through 1F) is the coprocessor register number. (Coprocessor ID 0 is reserved by the manufacturer for the MMU.)

The MMU access control registers are accessed at a second CPU space address, 0001 00XX, where XX is the MMU register number (0 through 7F).

Refer to the manufacturer's data sheets for more information regarding communication with coprocessors.

Simulating Interrupt Acknowledge

3-60.

An interrupt acknowledge cycle can be simulated by performing a read operation with CPU BYTE address option selected. The address of the read

operation must be specified as FFFF FFFX. X encodes the interrupt level being acknowledged according to the following list:

LEVEL	VALUE OF X
1	3
2	5
3	7
4	9
5	B
6	D
7	F

These values conform with the actual address line encoding during interrupt acknowledge cycles (as defined by the manufacturer).

The response to the read operation is the interrupt vector number if $\overline{\text{AVEC}}$ on the processor is not asserted.

Interrupt acknowledge cycles expect the vector number to be placed on the data bus. On an 8-bit port, the vector number is placed on D24 through D31. On a 16-bit port, the vector number is placed on D16 through D23. On a 32-bit port, the vector number is placed on D0 through D7.

Vector numbers are always correctly read from UUTs that supply the numbers over an 8-bit or 16-bit data port. Because interrupt acknowledge cycles are simulated with a read from CPU BYTE space, not all vector numbers supplied over a 32-bit port can be read. When a 32-bit port is used, correct vector numbers may only be read if the interrupt level specified is 1, 3, 5, or 7. Vector numbers for level 2, 4, or 6 acknowledgments may not be read properly if the UUT data placement is forced to D16 through D23 due to the low level on A1.

Simulating Breakpoint Acknowledge

3-61.

NOTE

Breakpoint under this heading refers to the breakpoint feature built into the 68020 processor. For information on the 9132A Pod RUN UUT breakpoint feature, see the heading "Using Breakpoints" earlier in this section.

A breakpoint acknowledge cycle can be simulated by performing a read operation with the address option set to CPU WORD. The address must be specified as 0000 00XX, where XX (4 through 1C by fours) encodes the breakpoint number (0 through 7) being acknowledged. The UUT responds with either a 16-bit opcode (which is the data returned by the read operation) or by asserting $\overline{\text{BERR}}$. If $\overline{\text{BERR}}$ is asserted, the Pod reports an "interrupt or exception received" fault message.

INFORMATION ABOUT 68020 SIGNALS 3-62.

Table 3-2 lists all of the 68020 microprocessor signals and provides a brief description of how each signal is handled by the Pod. Table 3-3 contains the pin numbers for each of the microprocessor signals. Figure 3-2 shows the 68020 microprocessor pin assignments. Figure 3-3 shows the location of the microprocessor signals on the top side of the standard Sync Module Adapter Board.

Table 3-2. 68020 Signal Descriptions

<table>
<tr><th>SIGNAL NAME</th><th>DESCRIPTION</th></tr>
<tr><td>A0-A31</td><td>The Address Bus consists of 32 unidirectional tri-state output lines, and provides the bus address for all processor operations. Up to 14 address lines are monitored by ROM Module 1.</td></tr>
<tr><td>D0-D31</td><td>These are the 32-bit, bidirectional, tri-state data signals of the 68020 microprocessor. D24 through D31 are monitored by the Sync Module.</td></tr>
<tr><td>FC0-FC2</td><td>The Function Code lines are tri-state outputs that indicate the state (user or supervisor) and address space of the current bus cycle, as shown below:
<table>
<tr><th>FC2</th><th>FC1</th><th>FC0</th><th>CYCLE TYPE</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>(Undefined, Reserved)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>User Data Space</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>User Program Space</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>User Defined</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>(Undefined, Reserved)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Supervisor Data Space</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Supervisor Program Space</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>CPU Space</td></tr>
</table></td></tr>
<tr><td>SIZ0, SIZ1</td><td>Transfer Size is two tri-state outputs that indicate the remaining number of bytes to be transferred during the next bus cycle.</td></tr>
<tr><td>$\overline{ECS}$</td><td>The External Cycle Start line is an output that provides the first indication that the processor may be starting a bus cycle. This signal must be validated by an address strobe since the processor may abort the instruction fetch.</td></tr>
<tr><td>$\overline{OCS}$</td><td>The Operand Cycle Start line is an output with the same timing as the $\overline{ECS}$ line, but is only asserted during the first bus cycle of an operand transfer or instruction prefetch.</td></tr>
<tr><td>$\overline{RMC}$</td><td>The Read/Modify/Write Cycle line is a tri-state output that indicates the current bus operation is an indivisible read-write-modify cycle.</td></tr>
<tr><td>$\overline{AS}$</td><td>The Address Strobe line is a tri-state output which indicates that valid data is present on the Address and Function Code lines, and that size and R/$\overline{W}$ information is available.</td></tr>
</table>

Table 3-2. 68020 Signal Descriptions (cont)

SIGNAL NAME	DESCRIPTION
$\overline{DS}$	The Data Strobe line is a tri-state output which indicates during a read cycle that a slave device is driving the data bus and indicates during a write cycle that the 68020 processor has placed valid data on the data bus.
R/$\overline{W}$	The Read/Write line indicates that direction data is to be transferred on the Data Bus.
$\overline{DBEN}$	The data buffer enable output provides an enable to external data buffers. $\overline{DBEN}$ allows the R/$\overline{W}$ signal to change without possible external buffer contention.
$\overline{DSACK0}$ $\overline{DSACK1}$	The Data Transfer and Size Acknowledge inputs indicate that the bus data transfer will be completed at the end of the current processor clock cycle and the port width of the external data transfer device (8-, 16-, or 32-bit). During Read cycles, the processor latches the input data at the end of the clock cycle in which $\overline{DSACKx}$ is recognized then terminates the bus cycle. When writing, $\overline{DSACKx}$ terminates the current bus cycle.
$\overline{CDIS}$	The Cache Disable input disables the processor's on-board cache on the first cache access after the $\overline{CDIS}$ line is asserted. The on-board cache is reenabled on the first cache access after the $\overline{CDIS}$ line is negated.
$\overline{BR}$	The Bus Request input indicates to the processor that some other device is requesting control of the bus. $\overline{BR}$ is overdriven high during critical periods of Pod operation.
$\overline{BG}$	The Bus Grant line output indicates that the processor will relinquish control of the bus at the end of the current bus cycle.
$\overline{BGACK}$	The Bus Grant Acknowledge input indicates that a device other than the processor has assumed control of the bus.
$\overline{IPL0}$ - $\overline{IPL2}$	The Interrupt Priority Level lines indicate the encoded priority level of an interrupting device. These lines have the following characteristics: • Level 0 indicates no interrupt is pending. • Level 7 is the highest priority interrupt. • Interrupt levels 1 through 6 are maskable, while level 7 is not.
$\overline{IPEND}$	The Interrupt Pending output indicates that the priority level of the signals on the $\overline{IPL0}$ - $\overline{IPL2}$ lines are of a higher priority than the level of the interrupt currently being processed or that a non-maskable interrupt has occurred.
$\overline{AVEC}$	The Autovector input requests internal generation of the vector number during an interrupt acknowledge cycle.

Table 3-2. 68020 Signal Descriptions (cont)

SIGNAL NAME	DESCRIPTION
$\overline{\text{BERR}}$	The Bus Error line indicates a problem with the current bus cycle. This line shows the following characteristics when asserted: • When asserted concurrently with the $\overline{\text{HALT}}$ line, the processor will rerun the current bus cycle if the $\overline{\text{BERR}}$ line is released before or at the same time as the $\overline{\text{HALT}}$ line. • If asserted during Reset Vector Acquisition, the processor will halt. • If asserted for two consecutive bus cycles, the processor will halt. • If asserted alone, it will cause the processor to execute a non-maskable interrupt to the Bus Error Vector (Vector 2).
$\overline{\text{RESET}}$	The Reset line is a bidirectional open collector line that resets the state of the processor.
$\overline{\text{HALT}}$	The $\overline{\text{HALT}}$ line is a bidirectional open collector line that is asserted when the processor stops due to an unrecoverable error sequence. External devices may pull this line low in the following circumstances: • During a bus cycle to stop the processor at the end of the cycle. • To rerun the last bus cycle (used in conjunction with $\overline{\text{BERR}}$).
CLK	The Clock line is an input used to derive the clocks needed internally by the processor.

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

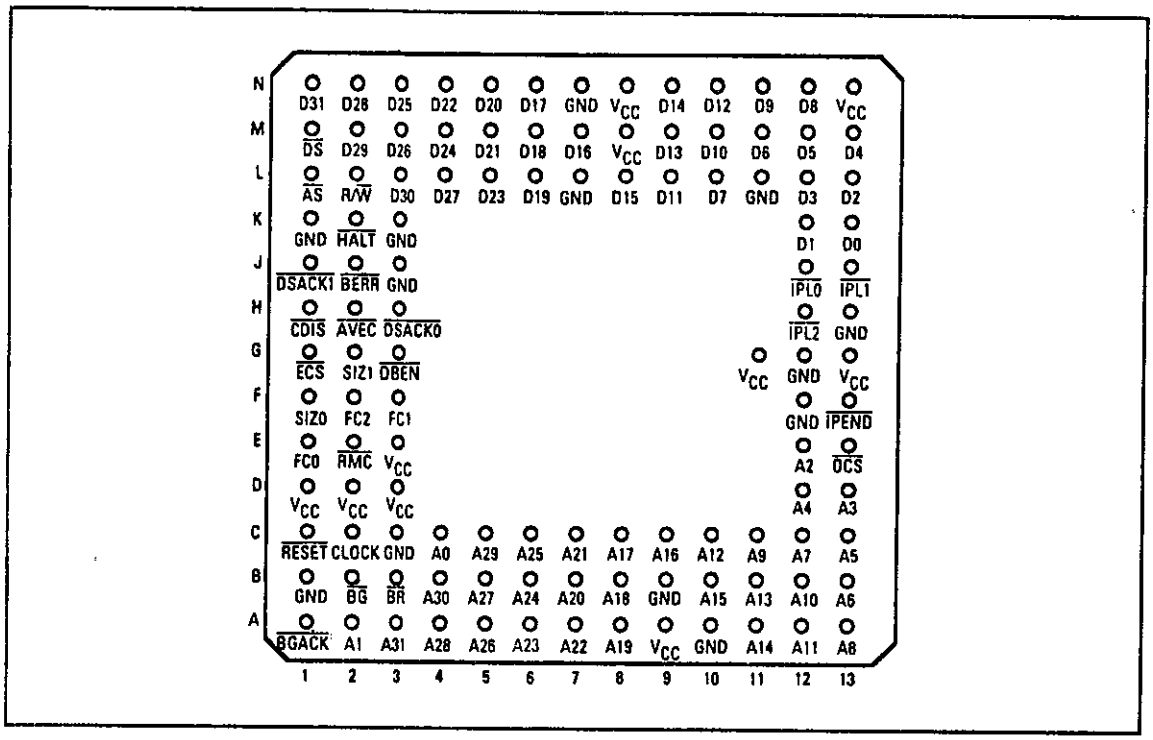


Figure 3-2. 68020 Microprocessor Pin Assignments (Bottom View)

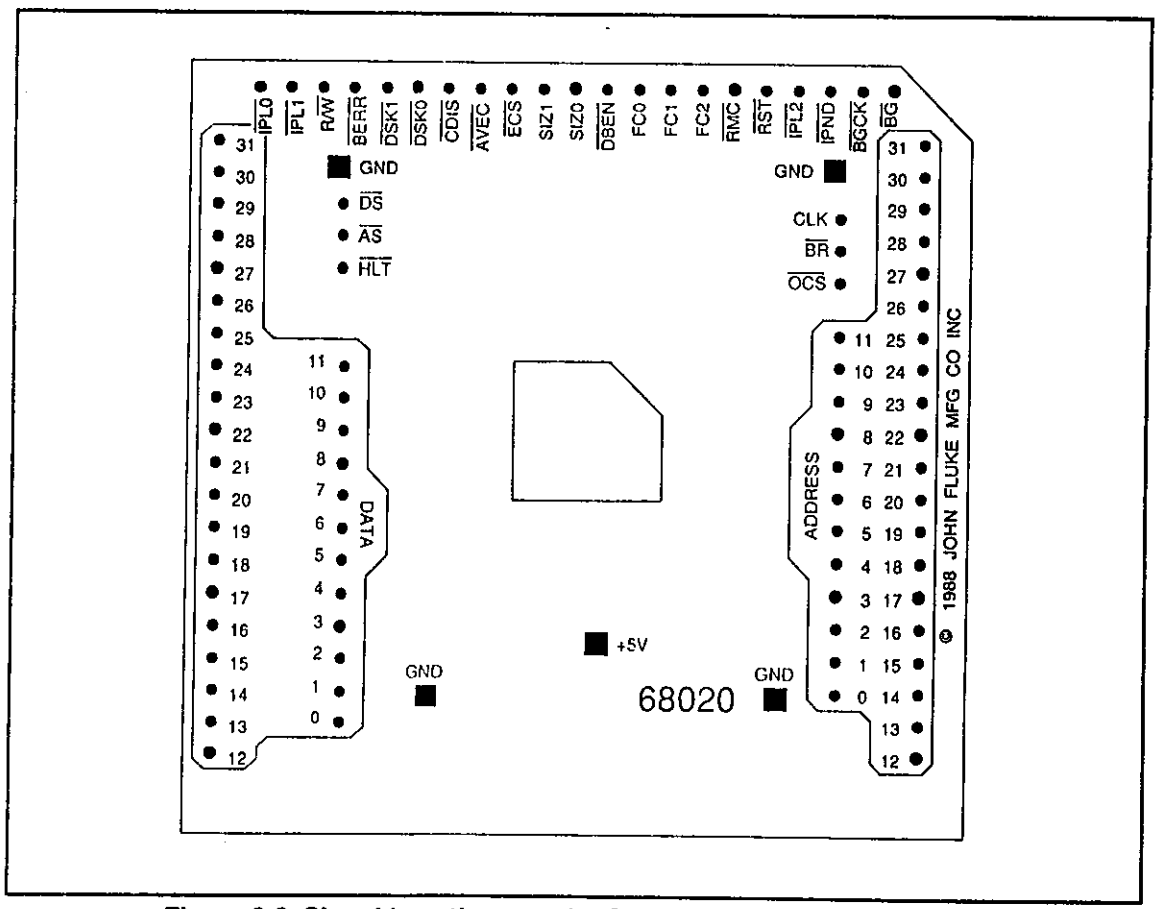


Figure 3-3. Signal Locations on the Standard Sync Adapter Board

Table 3-3. 68020 Microprocessor Pin Locations

PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
A3	A31	N1	D31	F2	FC2	C2	CLOCK
B4	A30	L3	D30	F3	FC1		
C5	A29	M2	D29	E1	FC0	A9	Vcc
A4	A28	N2	D28			D1	Vcc
B5	A27	L4	D27	G2	SIZ1	D2	Vcc
A5	A26	M3	D26	F1	SIZ0	D3	Vcc
C6	A25	N3	D25			E3	Vcc
B6	A24	M4	D24	G1	$\overline{ECS}$	F12	Vcc
A6	A23	L5	D23	E13	$\overline{OCS}$	G11	Vcc
A7	A22	N4	D22	E2	$\overline{RMC}$	G13	Vcc
C7	A21	M5	D21	L1	$\overline{AS}$	M8	Vcc
B7	A20	N5	D20	M1	$\overline{DS}$	N8	Vcc
A8	A19	L6	D19	L2	R/$\overline{W}$	N13	Vcc
B8	A18	M6	D18	G3	$\overline{DBEN}$		
C8	A17	N6	D17	J1	$\overline{DSACK1}$	A10	GND
C9	A16	M7	D16	H3	$\overline{DSACK0}$	B9	GND
B10	A15	L8	D15			C3	GND
A11	A14	N9	D14	H1	$\overline{CDIS}$	F12	GND
B11	A13	M9	D13			G12	GND
C10	A12	N10	D12	H12	$\overline{IPL2}$	H13	GND
A12	A11	L9	D11	J13	$\overline{IPL1}$	J3	GND
B12	A10	M10	D10	J12	$\overline{IPL0}$	K1	GND
C11	A9	N11	D9	F13	$\overline{IPEND}$	K3	GND
A13	A8	N12	D8	H2	$\overline{AVEC}$	L7	GND
C12	A7	L10	D7			L11	GND
B13	A6	M11	D6	B3	$\overline{BR}$	N7	GND
C13	A5	M12	D5	B2	$\overline{BG}$		
D12	A4	M13	D4	A1	$\overline{BGACK}$	D4	N.C.
D13	A3	L12	D3			THROUGH	
E12	A2	L13	D2	C1	$\overline{RESET}$	D11	N.C.
A2	A1	K12	D1	K2	$\overline{HALT}$		
C4	A0	K13	D0	J2	$\overline{BERR}$		

Appendix A
UUT ROM Support

ROM TYPES SUPPORTED BY THE 9132A A-1.

A single ROM Module can be used with a variety of different ROM types (the primary consideration is if the number of pins on the ROM Module correspond with the number of pins on the UUT ROM). The following standard ROM types are directly supported by the 9132A Interface Pod:

2716	27128
2732	27256
2764	27512

To select any of these standard ROM types, press the SETUP MENU key on the Mainframe keypad and select SETUP POD ROM_TYPE. A list of the standard ROM types supported by the Pod is listed on the display (the default ROM type for the 9132A-68020 is the 27256).

CAUTION

The 9132A Pod does not support 2716s that require -5V and +12V supply voltages (e.g., TMS 2716). Damage to the ROM Module may occur if these voltages are applied.

Table A-1 contains a list of ROM types that have the same pin configuration as the standard ROM types. The signals for these ROM types coincide directly with the signals of the standard ROMs listed on the right hand column. To select any of these ROM types, use the SETUP POD ROM_TYPE key sequence and choose the corresponding standard ROM.

Table A-1. ROM Types Similar to Standard ROMs

NUMBER	ORGANIZATION	SELECT ROM_TYPE
2516	2k X 8	2716
27C16	2k X 8	2716
27C32	4k X 8	2732
27C64	8k X 8	2764
27C128	16k X 8	27128
27C256	32k X 8	27256
27C512	64k X 8	27512

Additionally, a variety of other ROM types may be selected by entering a special code under the SETUP POD ROM_TYPE key sequence. Figure A-1 contains the pin diagrams of ROM types that can be selected.

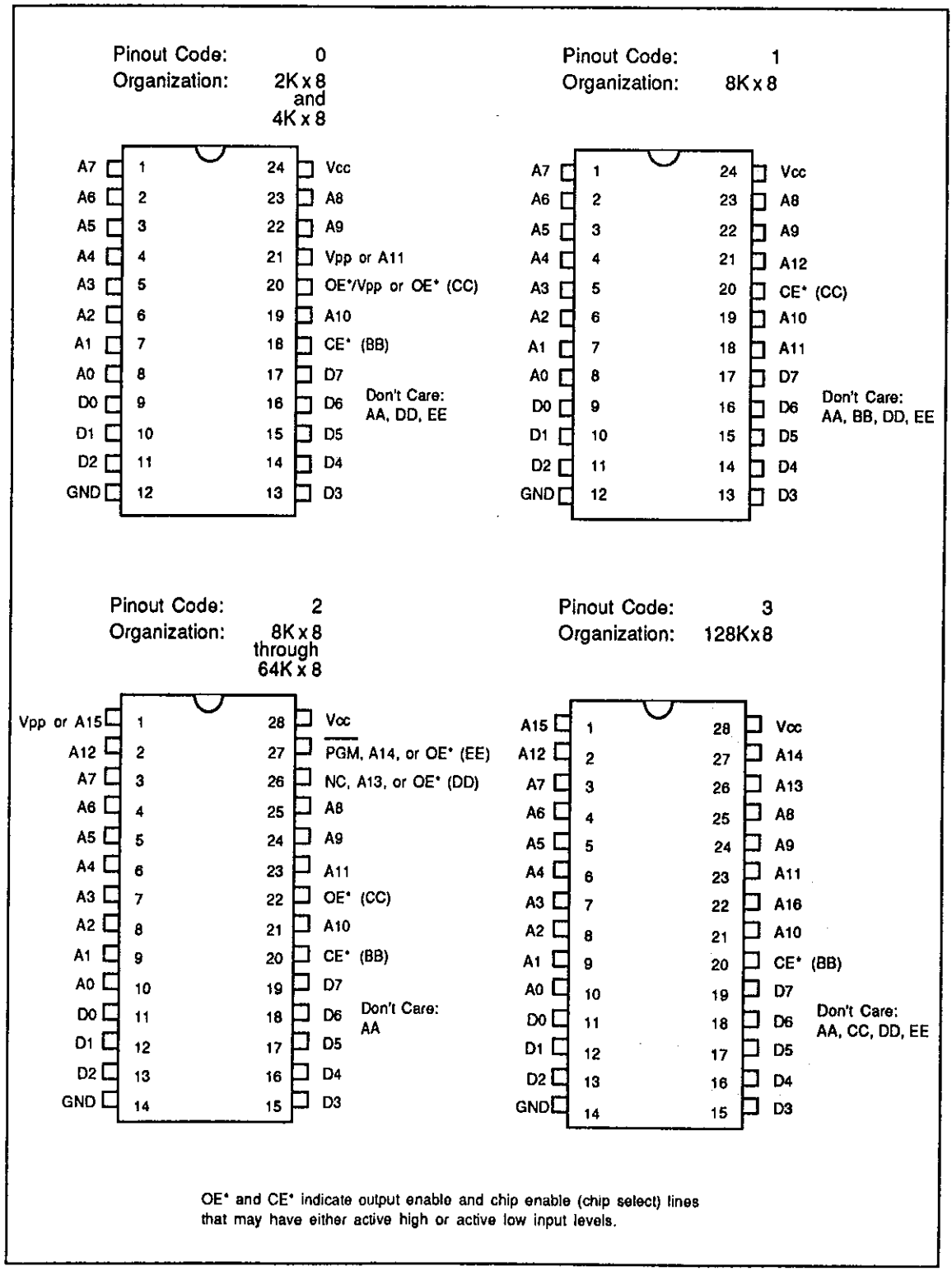


Figure A-1. Pin Diagrams of Supported ROM Types

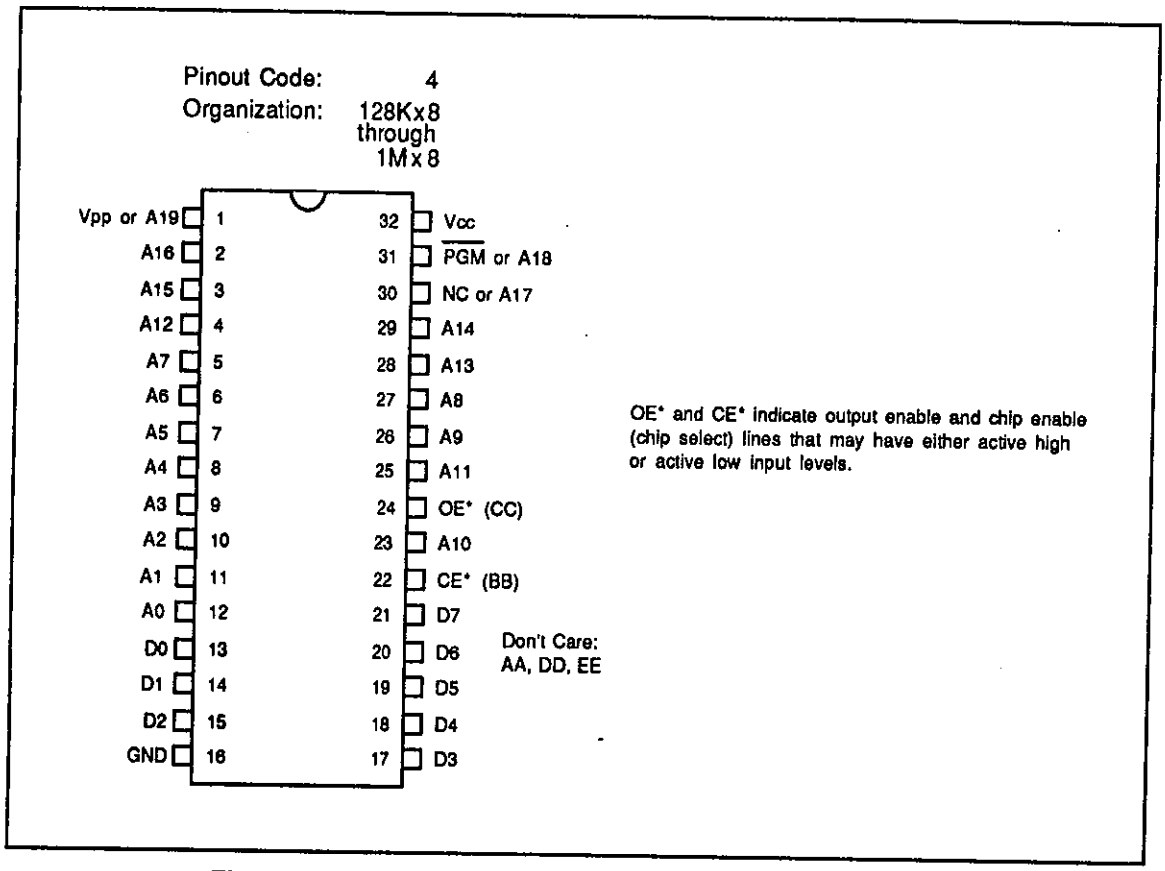


Figure A-1. Pin Diagrams of Supported ROM Types (cont)

A special code for each of these additional ROM types must be entered into the Pod using the SETUP POD ROM_TYPE OTHER function. The number that is entered is made up of a series of bits that describe to the Pod the size of the ROM, the pinout description, and the chip enable (CE) and output enable (OE) variations. These codes are entered as a hexadecimal number with the bit coding:

0000 GGGG PPPP 00EE DDCC BBAA 0000 0000

The "0" bits are reserved for future expansion and must be set to zero.

The G bits are the geometry (organization) code that defines the number of address and data lines on the ROM. These codes are defined (in hex) as:

0	2K x 8
1	4K x 8
2	8K x 8
3	16K x 8
4	32K x 8
5	64K x 8
6	128K x 8
7	256K x 8
8	512K x 8
9	1M x 8
A-F	future expansion

The P bits are pinout codes that define the number of pins and the address and data signal assignments. These codes are defined (in hex) as:

0	24 pin, 2716/2732 style
1	24 pin, 2364/2366 style
2	28 pin, 2764/128/256/512 style
3	28 pin, 23C1000 style
4	32 pin, 27C010 style
5-F	future expansion

The BB, CC, DD, and EE bits define the CE/OE variations according to the following codes (in binary):

00	NC, don't care, or fixed by pinout definition
01	$\overline{CE}/\overline{OE}$ active low
10	CE/OE active high
11	reserved

The location of these codes on the ROM pins is shown in Figure A-1. AA is not presently used and must be set to 0.

Table A-2 contains a set of predefined special numbers for various ROM types. Because some ROM types can be purchased with varying configurations for the chip select and output select, some of the numbers in the table are shown as an "X" to indicate that this hex digit of the setup code number must be determined by the user. Use Figure A-1 to determine the footprint and the signal configuration for the ROM, then use the preceding text on determining the ROM type setup code to establish the hex values required to replace the "X"s in Table A-2.

Table A-2. Predefined ROM Codes

24-PIN ROMS	ORG.	ROM_TYPE SETUP CODE	24-PIN ROMS	ORG.	ROM_TYPE SETUP CODE
2332	4K x 8	0100XX00	4764	8K x 8	0210X000
2333	4K x 8	0100XX00	5332	4K x 8	0100XX00
2364	8K x 8	0210X000	5364	8K x 8	0210X000
2366	8K x 8	0210X000	5366	8K x 8	0210X000
2516	2K x 8	00001400			
28-PIN ROMS	**ORG.**	**ROM_TYPE SETUP CODE**	**28-PIN ROMS**	**ORG.**	**ROM_TYPE SETUP CODE**
2364	8K x 8	022XXX00	23C512	64K x 8	0520XX00
23C64E	8K x 8	022XXX00	231000	128K x 8	06300400
2365	8K x 8	022XXX00	23C1000	128K x 8	06300400
23C65	8K x 8	022XXX00	3864	8K x 8	022XX400
23128	16K x 8	032XXX00	38128	16K x 8	032XX400
23C128	16K x 8	032XXX00	38256	32K x 8	0420X400 *
23256	32K x 8	0420XX00	38512	64K x 8	05201400
23C256	32K x 8	0420XX00	47128	16K x 8	032XXX00
23257	32K x 8	0420XX00	47256	32K x 8	0420XX00
23512	64K x 8	0520XX00	47C256	32K x 8	0420XX00

* Only supported when pin 1 is N.C.

Table A-2. Predefined ROM Codes (cont)

28-PIN ROMS	ORG.	ROM_TYPE SETUP CODE	28-PIN ROMS	ORG.	ROM_TYPE SETUP CODE
47C512	64K x 8	0520XX00	53256	32K x 8	0420XX00
47C1024	128K x 8	06300X00	53257	32K x 8	04202X00
5365	8K x 8	022XXX00	531000	128K x 8	06300X00
53128	16K x 8	032XXX00			
32-PIN ROMS	**ORG.**	**ROM_TYPE SETUP CODE**	**32-PIN ROMS**	**ORG.**	**ROM_TYPE SETUP CODE**
28F256	32K x 8	04401400	48C1024	128K x 8	06401400
27010	128K x 8	06401400	531001	128K x 8	06401400
27C010	128K x 8	06401400	532000	256K x 8	07401400
27C020	256K x 8	07401400	534000	512K x 8	08401X00
27C101	128K x 8	06401400	541000	128K x 8	06401400
27C1001	128K x 8	06401400	571000	128K x 8	06401400
48C512	64K x 8	05401400			

Appendix B
Problems Due to a Marginal UUT

INTRODUCTION

B-1.

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

UUT OPERATING SPEED AND MEMORY ACCESS

B-2.

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

UUT NOISE LEVELS

B-3.

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

BUS LOADING

B-4.

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

CLOCK LOADING

B-5.

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.

Appendix C
Testing UUTs With Soldered-in Components

INTRODUCTION C-1.

The 9132A Interface Pod is normally used with UUTs that have boot ROMs and microprocessors installed in sockets. The ROMs and microprocessor are removed from their UUT sockets and are replaced by the ROM Module plug(s) and Sync Module Adapter Board. UUTs that have soldered-in ROMs or microprocessors need different mechanical connections when testing with the Pod.

This appendix describes several methods for testing with soldered-in components. Also discussed are methods of designing a UUT to simplify testing with the Pod.

TESTING WITH A SOLDERED-IN MICROPROCESSOR C-2.

To test a UUT, the Pod must be able to access various UUT processor signals and overdrive the UUT's system reset. In most cases, the microprocessor is removed from the UUT socket and is replaced by the Sync Module Adapter board, which allows access to 8 data lines and monitors the status of the microprocessor signals.

When the UUT microprocessor is soldered, a different method of accessing the data and stimulus signals must be employed before the Pod can gain control of the UUT. If a test access connector is available on the UUT, connecting the Sync Module Adapter allows the Pod to access data lines and supply other signals needed by the Pod to control the UUT. (Table 2-1 lists the microprocessor signals and pin numbers for a test access connector that directly connects to the Sync Module Adapter cable.)

If a test access connector is not available, other ROM Module and Sync Module connections must be used. The 9132A Pod is provided with a set of test leads that can be plugged into the Sync Module. These test leads allow you to connect the Sync Module to the UUT signals needed by the Pod. Figure C-1 shows how to remove the Sync Module Adapter board cable and connect the test leads. The clips on the test leads must be connected to the proper signals on the UUT for the Pod tests to function correctly. Table C-1 describes the connections between the Sync Module and the UUT.

NOTE

*Connect the UUT RESET clip from the Sync Module to the UUT system reset line, **not** to the $\overline{RESET}$ pin of the 68020 microprocessor. For more information on where to connect the Sync Module RESET line, see Appendix G.*

Table C-1. Connections Between the Pod and the UUT (68020)

RESET		TIMING		DATA BUS	
SYNC MOD. LINE	UUT SIGNAL	SYNC MOD. LINE	UUT SIGNAL	SYNC MOD. LINE	UUT SIGNAL
UUT RESET	System Reset	CLK	CLK	Data Line 0	D24
		Channel 1	$\overline{\text{RESET}}$	Data Line 1	D25
		Channel 2	$\overline{\text{AS}}$	Data Line 2	D26
		Channel 3	$\overline{\text{BR}}$	Data Line 3	D27
		Channel 4	$\overline{\text{BG}}$	Data Line 4	D28
		Channel 5	$\overline{\text{BGACK}}$	Data Line 5	D29
		Channel 6	$\overline{\text{HALT}}$	Data Line 6	D30
		Channel 7	$\overline{\text{BERR}}$	Data Line 7	D31

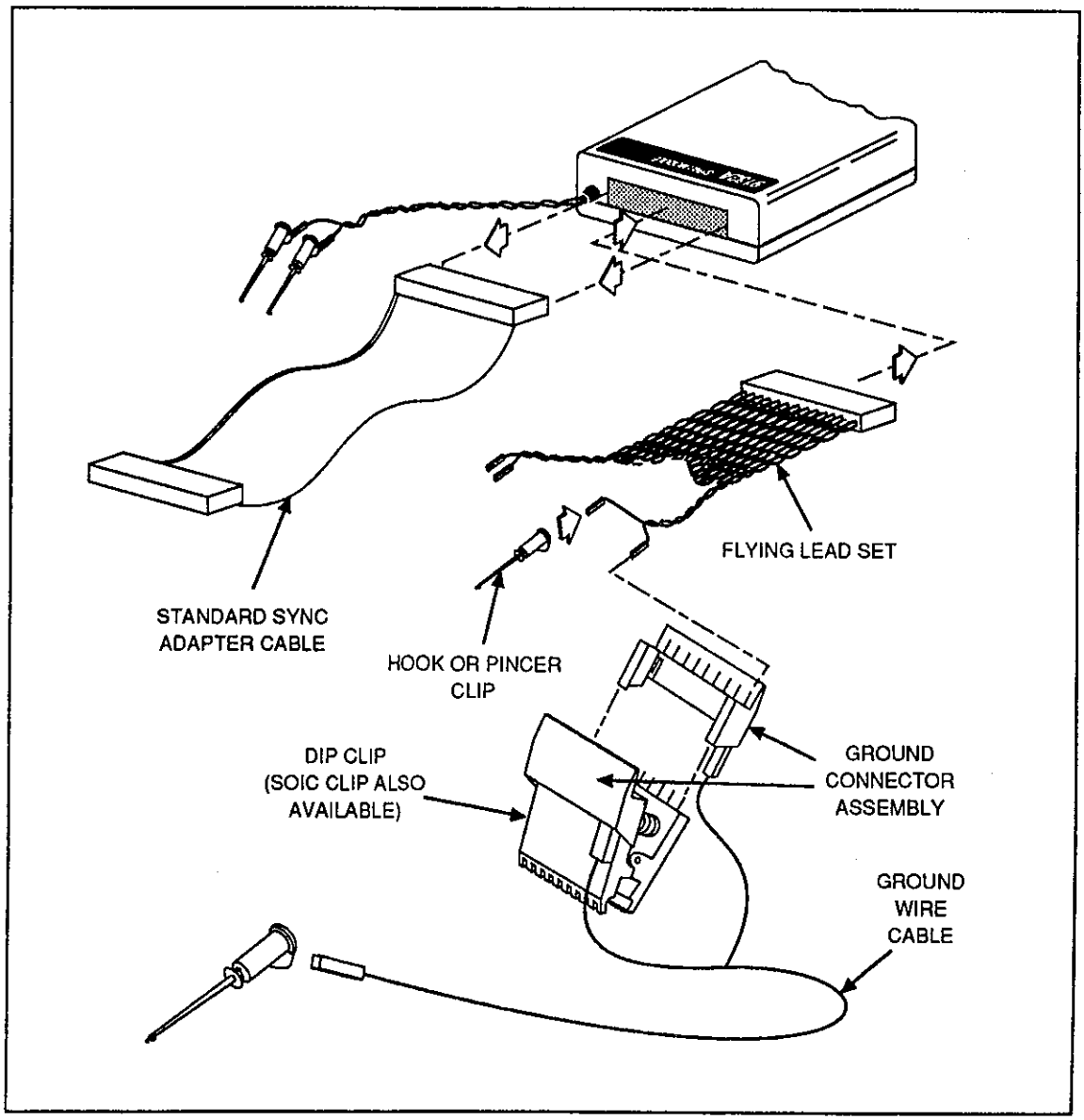


Figure C-1. Installing the Flying Lead Set and Clip Accessories

Each of the Sync Module lines has an individual ground line. The ground line for UUT RESET must be connected to the UUT for effective grounding of the Sync Module lines. For high speed applications, the ground clips for the rest of the Sync Module lines must be connected to the UUT.

Fluke offers a selection of clip and connector accessories to simplify access in testing UUTs with soldered components. These accessories attach to the flying lead set and connect directly to integrated circuits on the UUT (as demonstrated in Figure C-1. In addition, the clip assemblies allow you to tie the ground lines from the flying lead set together to make a single ground connection to the UUT. Table C-2 contains a list of the available parts.

Table C-2. Accessories for the Flying Lead Set

IC DEVICE SIZE	DIP CLIP (J/F PART)	SOIC CLIP (J/F PART)	GROUND CONNECTOR ASSEMBLY (2 USED PER CLIP) (J/F PART)
14 pins	800052	817429	801878
16 pins	800060	817437	801886
18 pins	800078	-----	801894
20 pins	800086	817445	801902
24 pins	800094	817478	801910
28 pins	800102	821975	801928
40 pins	800110	-----	801936

The ground connector assemblies in Table C-2 use a ground wire cable (Fluke part number 801704) to attach to a nearby ground point on the UUT.

The flying lead set and the ground wire cable can also use either the hook clip (Fluke part number 757500) or the pincer clip (Fluke part number 845409) to attach directly to the UUT.

TESTING WITH SOLDERED-IN BOOT ROMS C-3.

In most cases, the UUT boot ROMs are removed from the UUT sockets and are replaced by the Pod's ROM Modules. When the ROMs are soldered, some method of disabling the UUT boot ROM must be designed into the UUT before the Pod can read or write to the UUT bus.

One method of accessing the UUT bus is to override the UUT timing control signal to the boot ROM (i.e., CE and/or OE). A simple jumper configuration may be adequate to switch the normal boot ROM CE connection to a ROM Module through a spare socket or a clip on the soldered boot ROM. An example of this method is shown in Figure C-2. The limitations to this method are that the Pod cannot RUN UUT at reset with the UUT code and there can be no ROM test of the UUT boot ROMS.

Another means of testing soldered ROMs is to provide a method of relocating the UUT boot ROM address space during testing. Using this method, whenever the ROM Modules are connected to the test access connector, the hardware on the UUT shifts the address of the boot ROM to a different location. An example of this method is shown in Figure C-3. This

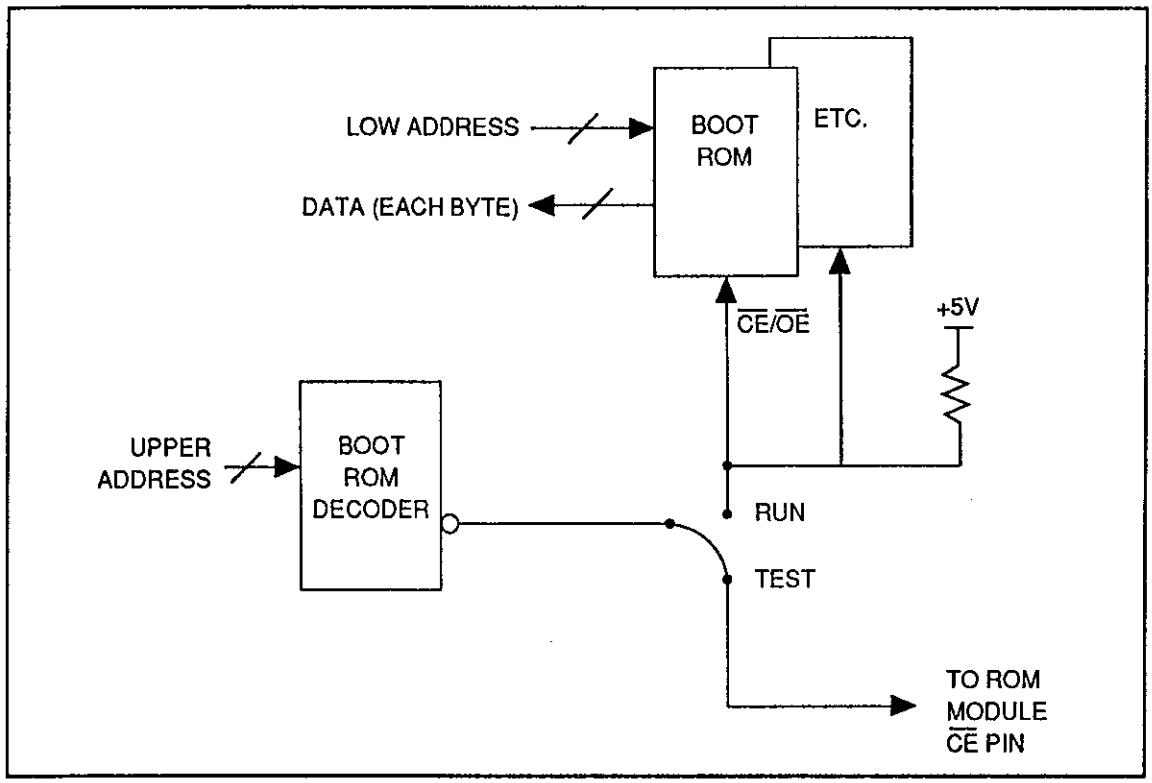


Figure C-2. Disabling the UUT Boot ROM

method allows the Pod to control the shifted address space for read accesses and ROM test, but still does not allow RUN UUT at reset with the UUT code.

The best method of testing a UUT with soldered-in ROMs is to use the previously described method while using relocatable UUT boot ROM code. This method allows the ROM Modules to access the normal boot ROM address (as in the method above), and also allows the Pod to perform RUN UUT at reset on the UUT code at its shifted address.

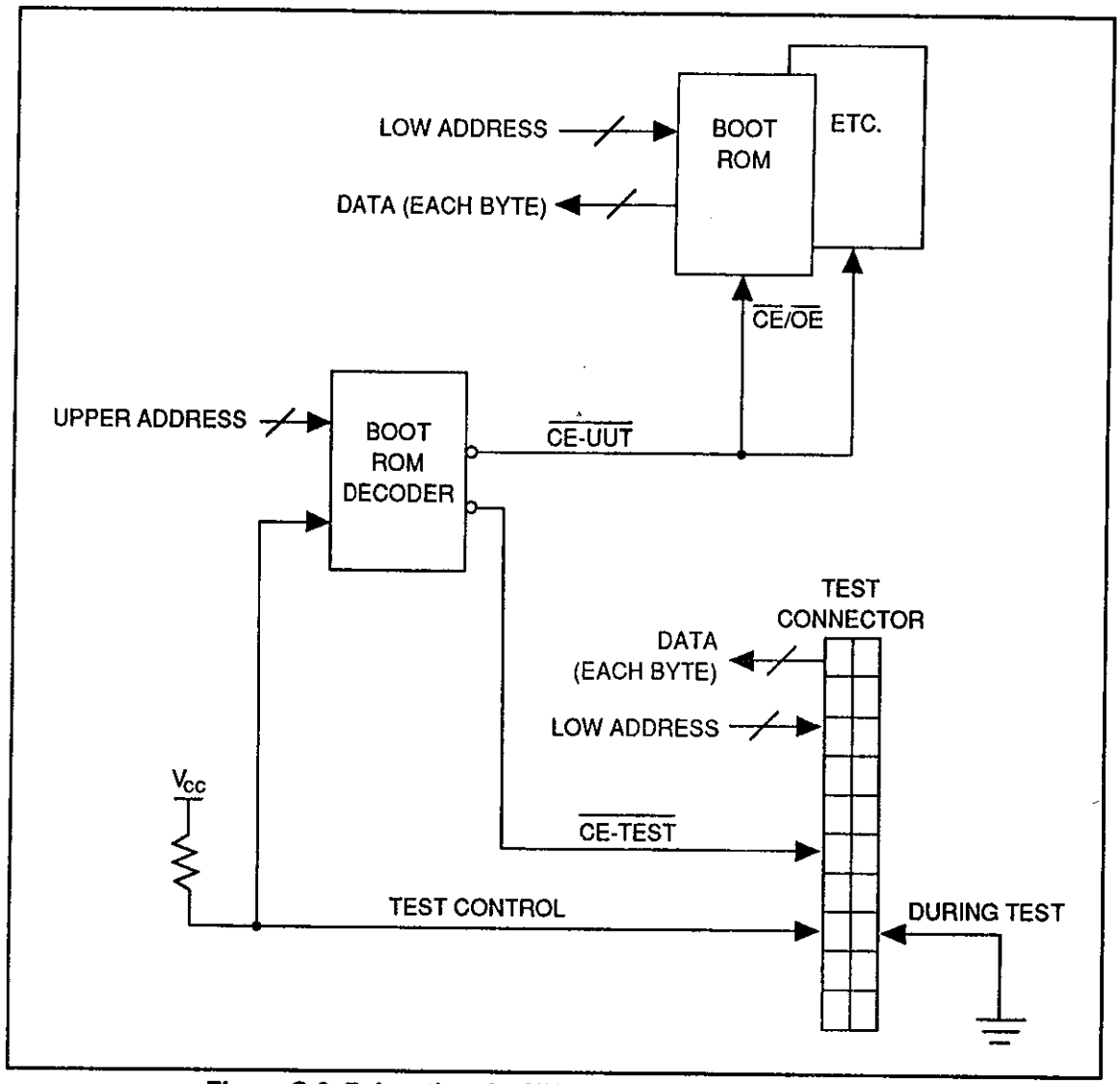


Figure C-3. Relocating the UUT Boot ROM Address Space

Appendix D
Pod Setup and Calibration

SETUP AND CALIBRATION PARAMETERS D-1.

Table D-1 contains a list of the Pod setup and calibration parameters, noting the default values that are entered when the 9132A-68020 database is loaded. The table includes an explanation of the function of each attribute. Each attribute is set by pressing the SETUP key on the mainframe and selecting SETUP POD.

Normally, the values of the parameters described in this Appendix are determined by using the Interactive Setup and Calibration routine (see Section 2 of this manual). However, each parameter can also be set by TL/1 commands (see Appendix E for details of the TL/1 statements) or front panel operations. Front panel setups are performed by pressing the SETUP key, selecting the POD softkey, and selecting the function and parameters needed. Choices for each parameter are listed down to the final level of the menu tree before the next parameter list begins.

Table D-1. Pod Setup and Calibration Parameters

REPORT POWER SUPPLY FAILURE
Values: ON, OFF
Default: ON

Specifies whether UUT power is monitored at the ROM Module connections. If so, power failure faults are reported if the UUT supply drops below about 3.5 volts.

REPORT INTR ACTIVE
Values: ON, OFF
Default: OFF

Not used.

REPORT FORCING SIGNAL ACTIVE
Values: ON, OFF
Default: ON

Specifies whether the Pod reports active forcing signals to the Mainframe. 68020 forcing signals are $\overline{\text{RESET}}$, $\overline{\text{BR}}$, $\overline{\text{BGACK}}$, $\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$.

REPORT CONTROL DRIVE ERROR
Values: ON, OFF
Default: ON

Not used.

Table D-1. Pod Setup and Calibration Parameters (cont)

REPORT ADDR DRIVE ERROR
Values: ON, OFF
Default: ON

Not used.

REPORT DATA DRIVE ERROR
Values: ON, OFF
Default: ON

Not used.

REPORT SPECIAL POD ERROR
Values: ON, OFF
Default: ON

Specifies whether the Pod's "special" errors are to be reported. These errors report conditions detected by the Pod which may be related to UUT faults, but do not otherwise fit into one of the predefined fault categories. This attribute should be left ON at all times.

TIMEOUT
Values: 0 to 9999999
Default: 5000000 microseconds (5 seconds)

Specifies the maximum time (in microseconds) that the Mainframe is to wait for response from the Pod following a command. In most cases there is no reason to lower this value. The value should be raised if the Pod is connected to a particularly slow UUT.

ENABLE
Values: None
Default: None

Not used.

INTRFACE ROM_MODS
Values: 4, 2, 1
Default: 1

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

INTRFACE BCYCLCLK
Values: SYNC_MOD, ROM_CE
Default: SYNC_MOD

Specifies desired "Bus Cycle Clock" signal source. Acceptable performance may be available with timing derived from the boot ROM enable signals, though the timing derived from the Sync Module is usually better. See the heading, Selecting the Bus Cycle Clock Source, further on in this Appendix for more information.

Table D-1. Pod Setup and Calibration Parameters (cont)

INTRFACE RST_LEN
Values: 0 to 9999999
Default: 1000

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

INTRFACE RST_POL
Values: LOW, HIGH
Default: LOW

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

INTRFACE XFER_ADR
Values: 0 to FFFFFFFF
Default: 0

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

INTRFACE CY_SPLIT
Values: 1, 2, 4
Default: 1

Specifies the bus width in bytes at the microprocessor divided by the number of ROM Modules. Unlike BURST_SZ, this setting does not care if the ROM Modules get selected or not. This setting is the ratio of access width at the microprocessor to that at the ROM Modules. See the heading, Setting the Burst Size and Cycle Split Setups, further on in this Appendix for more information.

INTRFACE BURST_SZ
Values: 1, 2, 4
Default: 1

Specifies the ratio of boot ROM addresses accessed to boot ROM enables. Change this setup if your UUT accesses multiple ROM locations during a single ROM enable and uses such "bursts" of data to respond to instruction fetches. See the heading, Setting the Burst Size and Cycle Split Setups, further on in this Appendix for more information.

INTRFACE DATAPRB
Values: YES, NO
Default: NO

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

Table D-1. Pod Setup and Calibration Parameters (cont)

INTRFACE ROM_BASE
Values: 0 to FFFFF000
Default: 0

Specifies the lowest address of the boot ROM space, and is only changed if the UUT has the means to relocate or alias the boot ROM to a location other than its normal location at the reset address.

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

Specifies the type of ROM used as the UUT boot ROM. (See Appendix A for information if you use the OTHER selection.)

CALIBRTN ADR_STIM
Values: 0 to 255
Default: 8

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

CALIBRTN RUN_UUT
Values: 0 to 255
Default: 4

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

RD_CAL
Values: 0 to 255
Default: 3

Specifies the number of microprocessor bus cycles expected between a trigger event and the appearance of the read cycle of interest on the UUT bus. $\overline{\text{PODSYNC}}$ is active during this bus cycle. UUT wait states and data bus width at the boot ROMs may have an effect on this value. There is a separate calibration for each address space:

UDATA_L	user data long	SDATA_L	supervisor data long
UDATA_W	user data word	SDATA_W	supervisor data word
UDATA_B	user data byte	SDATA_B	supervisor data byte
UPGM_L	user program long	SPGM_L	supervisor program long
UPGM_W	user program word	SPGM_W	supervisor program word
UPGM_B	user program byte	SPGM_B	supervisor program byte
UDEF_L	user defined long	CPU_L	CPU long
UDEF_W	user defined word	CPU_W	CPU word
UDEF_B	user defined byte	CPU_B	CPU byte

Table D-1. Pod Setup and Calibration Parameters (cont)

WR_CAL
Value: 0 to 255
Default: 3

Specifies the number of microprocessor bus cycles expected between a trigger event and the appearance of the write cycle of interest on the UUT bus. PODSYNC is active during this bus cycle. UUT wait states and data bus width at the boot ROMs may have an effect on this value. There is a separate calibration for each address space:

UDATA_L	user data long	SDATA_L	supervisor data long
UDATA_W	user data word	SDATA_W	supervisor data word
UDATA_B	user data byte	SDATA_B	supervisor data byte
UPGM_L	user program long	SPGM_L	supervisor program long
UPGM_W	user program word	SPGM_W	supervisor program word
UPGM_B	user program byte	SPGM_B	supervisor program byte
UDEF_L	user defined long	CPU_L	CPU long
UDEF_W	user defined word	CPU_W	CPU word
UDEF_B	user defined byte	CPU_B	CPU byte

SELECTING THE BUS CYCLE CLOCK SOURCE D-2.

The primary use of the bus cycle clock is to capture addresses present at ROM Module 1 at various times during testing. One important use of this is to capture addresses during the execution of bus test primitives. Only if a valid trace of the addresses is captured is the 9132A capable of properly analyzing the bus test operation.

Along with addresses, each step captures the state of the ROM CE/OE seen at ROM Module 1. This allows the 9132A to determine if the boot ROM was properly selected.

The 9132A allows the user to select either SYNC_MOD or ROM_CE as the bus cycle clock source through the SETUP menu (SETUP POD INTRFACE BCYCLCLK). SYNC_MOD specifies that the bus cycle clock should be derived from the timing signals present at the Sync Module connection. Each 9132A personality module contains custom circuitry to generate this signal. In the case of the 68020 Pod in this mode, the bus cycle clock corresponds directly to the processor's $\overline{AS}$ (address strobe) signal. ROM_CE specifies that the enabling combination of the CE (chip enable or chip select) and OE (output enable) signals, seen by ROM Module 1, be used to generate the bus cycle clock.

Depending on a UUT's particular design, either one, the other, or both bus cycle clock sources may be used. Figure D-1 contains three examples that each represent a different UUT design.

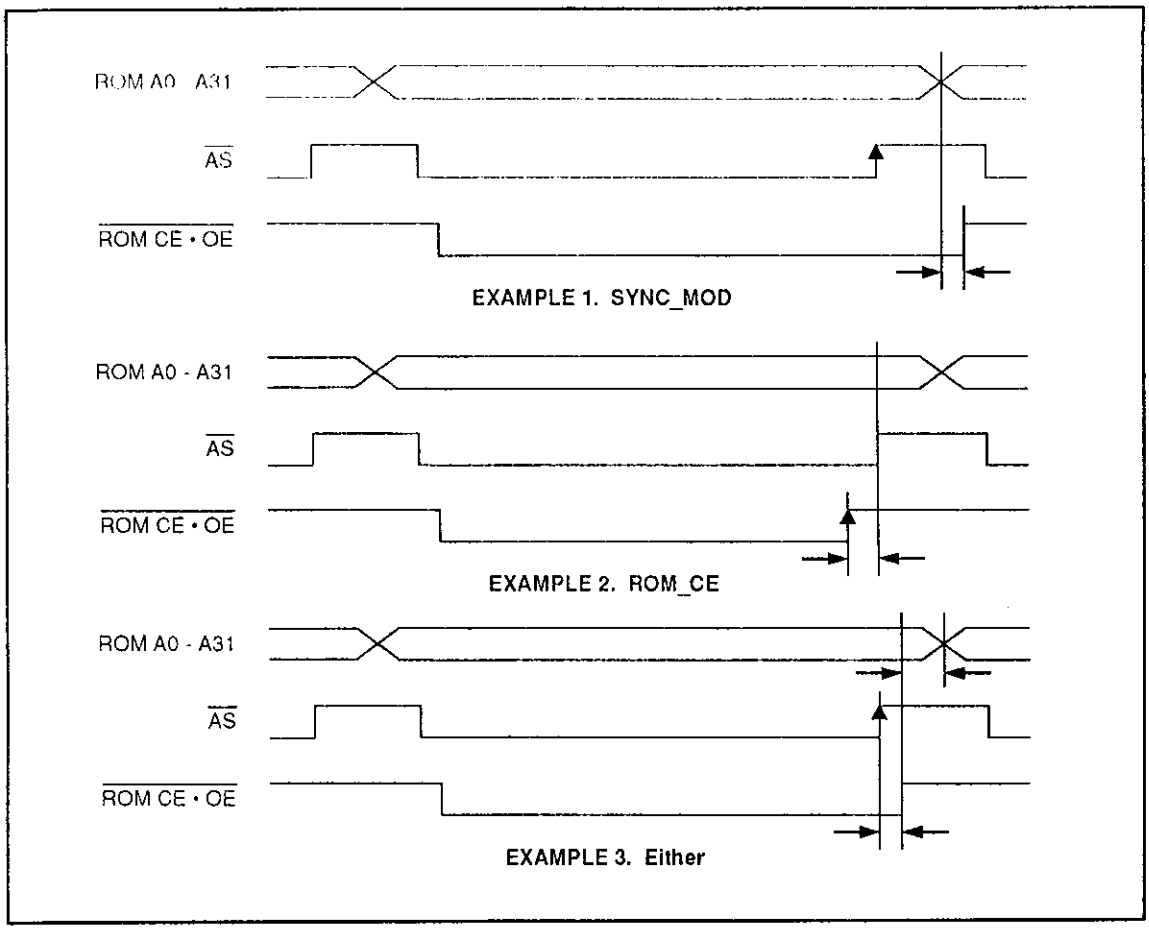


Figure D-1. BCYCLCLK Generation

Example 1 shows a UUT in which only SYNC_MOD may be used as the bus cycle clock source. The arrow on the trailing edge of AS shows the time at which addresses are captured. ROM_CE will not work in this example because the trailing edge of the ROM enable occurs when valid addresses are no longer present. Thus, this signal would not be suitable for capturing addresses.

Example 2 shows a UUT that must use ROM_CE as the bus cycle clock source. The arrow on the trailing edge of the ROM enable shows the time at which addresses are captured. SYNC_MOD mode will not work in this example because the ROM enable signal is no longer present when the trailing edge of AS occurs. This prevents the Pod from determining whether the boot ROM had actually been selected.

In example 3, either bus cycle clock source may be used. Addresses are valid at the trailing edge of either clocking signal and the ROM enable is valid at the trailing edge of AS.

SETTING THE BURST SIZE AND CYCLE SPLIT SETUPS D-3.

The 9132A Pod uses the burst size parameter to interpret the results of Bus Test. Burst size indicates whether the address trace values increment by the number of bytes of Boot ROMs present or by some multiple of that

number. Formally, burst size is defined as the number of ROM access cycles performed (ROM addresses accessed) per bus cycle clock. (See the heading, Selecting the Bus Cycle Clock Source, for a description of the bus cycle clock.)

For example, when two Boot ROMs are present, the address trace values normally step by an increment of 2 (i.e., 0, 2, 4, 6). The Boot ROMs are accessed once for each bus cycle clock and a burst size of 1 is used. Some UUTs may be designed such that multiple ROM accesses are performed and the combined results are supplied to the processor for each processor instruction fetch bus cycle. If the processor instruction fetch bus cycle results in a single bus cycle clock but the Boot ROMs were accessed twice in this time, then a burst size of 2 is indicated.

Cycle split is similar to burst size but is used during the STIM_ADR primitive to indicate to the Pod where in the address trace the STIM_ADR access occurred. As with Bus Test, the STIM_ADR function uses the bus cycle clock to clock the address trace. If SYNC_MOD is selected as the bus cycle clock source, the cycle split parameter is not used; there is a one-to-one correspondence between bus cycle clocks and addresses traced. However, if ROM_CE is used as the bus cycle clock source, cycle split indicates the ratio of ROM accesses (addresses) to processor bus cycles.

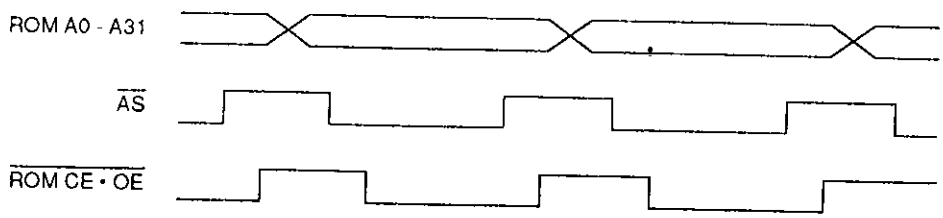
For example, if a UUT has only one Boot ROM but the processor fetches data 16 bits at a time, then the UUT must have some hardware that causes two ROM accesses to occur and the data to be assembled into a 16-bit quantity for each processor bus cycle. This UUT would require cycle split to be set to 2.

Typically if burst size is not 1, then cycle split and burst size are equal.

Example 1 in Figure D-2 contains signal timing for a UUT with a Boot ROM data width equal to the data size fetched by each processor bus cycle. The bus cycle clock is essentially the same regardless of whether SYNC_MOD or ROM_CE is selected; one bus cycle clock occurs for each processor bus cycle. Because ROM accesses, bus cycle clocks, and processor bus cycles all have a one-to-one correspondence, both BURST_SZ and CY_SPLIT are 1.

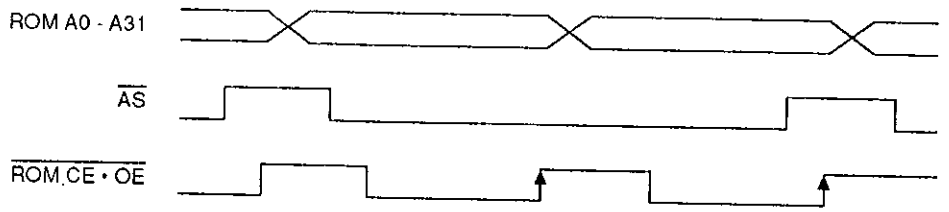
Examples 2 and 3 in Figure D-2 show signal timing for a UUT that accesses Boot ROM twice for each processor bus cycle (e.g., one Boot ROM in a UUT in which the processor fetches 16 bits at a time, or two Boot ROMs in which the processor accesses 32 bits at a time). Regardless of the bus cycle clock source, cycle split is 2 because there are two ROM accesses for each processor bus cycle. In example 2 in which the bus cycle clock is ROM_CE, BURST_SZ is 1 because there is one ROM address accessed for each bus cycle clock. In example 3, burst size is 2 because there are two ROM addresses accessed for each bus cycle clock.

Example 4 in Figure D-2 shows a UUT that performs two ROM accesses per processor bus cycle, but only generates one ROM enable strobe during that time. BURST_SZ is 2 since there are two ROM accesses per bus cycle clock and CY_SPLIT is 2 because there are two ROM accesses per processor bus cycle.



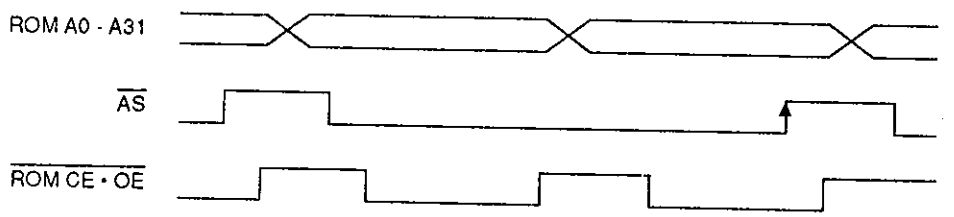
EXAMPLE 1

BCYCLCLK is SYNC_MOD or ROM_CE, BURST_SZ is 1, and CY_SPLIT IS 1.
There is one ROM access per bus cycle clock, and one ROM access per 68020 bus cycle.



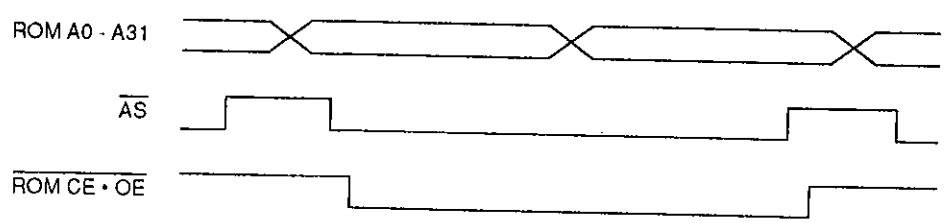
EXAMPLE 2

BCYCLCLK is ROM_CE, BURST_SZ is 1, and CY_SPLIT IS 2.
There is one ROM access per bus cycle clock, and two ROM accesses per 68020 bus cycle.



EXAMPLE 3

BCYCLCLK is SYNC_MOD, BURST_SZ is 2, and CY_SPLIT IS 2.
There are two ROM accesses per bus cycle clock, and two ROM accesses per 68020 bus cycle.



EXAMPLE 4

BCYCLCLK is SYNC_MOD or ROM_CE, BURST_SZ is 2, and CY_SPLIT IS 2.
There are two ROM accesses per bus cycle clock, and two ROM accesses per 68020 bus cycle.

Figure D-2. Burst Size and Cycle Split Settings

USING THE 68020 ROM_BASE SETUP D-4.

The 68020 processor begins execution after a reset by fetching an initial stack pointer value from address 0 and an initial program counter value from address 4. Once the program counter is loaded, execution continues at the address specified. The remaining addresses between 8 and 3FF are, by default, the location of the interrupt vector table. In general, boot ROM is located at address 0 upon reset of a 68020-based UUT. However, many UUTs are designed such that boot ROM execution is moved to another memory region and RAM is enabled at address 0. In this case, the software can control what values are located in the reset and interrupt vectors.

To accommodate this behavior, the 68020 Pod code executed on the UUT is designed to be relocatable. All program address references are either performed relative to the program counter, or are offset by the value located in the base address register. This value can be set using the Mainframe SETUP POD INTRFACE ROM_BASE selection. Since only the 13 least-significant UUT address lines are monitored by the Pod, ROM_BASE can be changed to any 8K byte address boundary without affecting the Pod's ability to monitor address lines. For example, valid ROM_BASE values include 0, 2000, 4000, etc.; upto a value of FFFF E000.

NOTE

The UUT must allow address aliasing between the original base address and the new base address for ROM_BASE relocation to function properly.

When the ROM_BASE setup is changed, the Pod reacts by immediately putting the new value in its base address register and by adjusting the program counter according to the new value. Once the relocation is performed, aliasing can be disabled.

Each time the 68020 is reset, the processor begins execution with the boot ROM assumed to be at address 0, and the ROM_BASE value is automatically applied after other initializations are completed. If the UUT address aliasing is affected by the processor reset, then it may be necessary to repeat the steps performed previously to disable the aliasing. (Actions that cause processor resets are described in Appendix G.)

USING THE XFER_CAL SETUP D-5.

The UUT communicates data with the Pod by performing write accesses at the XFER_ADR with the object data being placed on the processor's data bus. The Pod monitors the data lines through the Sync Module and captures the data at the proper time. This time is determined by the Pod monitoring bus cycle clocks and watching the addresses that appear at the ROM module connections. (Data transfer write accesses must take place without any bus errors or timeouts. Therefore, the XFER_ADR must be properly specified before any UUT communications are attempted. In the 68020 Pod, data transfers are done with Supervisor Data Long writes.)

Upon initialization in a new UUT (after the first time the Pod resets the UUT), an internal calibration sequence is executed that establishes the correct timing for UUT to Pod data transfers. The Pod commands the UUT to send a known data value several times until the correct bus cycle clock count is determined. This is accomplished through a series of writes at the XFER_ADR.

In some UUTs, most notably those that incorporate MMU hardware, this calibration immediately upon reset may not succeed. UUTs that incorporate MMU hardware may not allow Supervisor Data write accesses to the XFER_ADR (or to any address for that matter) until the MMU has been initialized. Initializing the MMU requires disabling the data transfer calibration and allowing the user to perform some write operations to the MMU before letting the data transfer calibration proceed.

To disable the data transfer calibration, perform a WRITE DATA xx TO VIRTUAL EXTADDR 2000000 ADDR 4C (where xx is a non-zero value that places a fixed calibration into the Pod and disables the automatic calibration performed on reset).

Use the following procedure to initialize a UUT with MMU upon power-up:

1. Use SETUP POD INTERFACE XFER_ADR to set the desired data transfer address.
2. Write non-zero data to virtual address 2000000, 4C.
3. Select address space option for MMU access.
4. Write values to the MMU that enable Supervisor Data writes at the XFER_ADDR.
5. Write 0 to virtual address 2000000, 4C.

After the procedure is concluded and a READ operation occurs, the Pod will perform the automatic data transfer calibration.

Appendix E
Using the 9132A Pod from TL/1 Programs

READING THE DATABASE VERSION NUMBER E-1.

The 9132A-68020 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.

An option exists under the Mainframe POD key that allows you to determine the software version number of the Pod and Personality Module. First press the POD key, press SOFT KEYS, select VERSION (F1), and press ENTER. The Mainframe displays both the Pod and Personality Module software version number.

Each part of the Pod and Personality Module software version can be read individually. The version number is returned as four hex digits of the form XXYY, where XX is the major version number and YY is the minor version number. For example, version number 2.3 is represented as 0203.

Begin by pressing the READ key, the VIRTUAL softkey, press the right arrow key (→), enter 200 0000 in the extended address field, and press the right arrow key. To read the Pod software version number, enter A4 in the address field and press ENTER. To read the Personality Module software version number, enter A8 in the address field and press ENTER.

TL/1 PROGRAMMING APPLICATIONS E-3.

Pod Address Space Options E-4.

Table E-1 shows which Pod address space options are available.

The following TL/1 program segment demonstrates how to change the Pod space:

```
program example1
        s = getspace space "USER", type "DATA", size "WORD"
        setspace space s
end example1
```

Table E-1. Pod Address Space Options

SPACE	TYPE	SIZE	OPS *	VIRTUAL ADDRESS	NOTES
USER	DATA	LONG	RWU	00000000 XXXXXXXX	Address must be multiple of 4.
USER	DATA	WORD	RWU	00000010 XXXXXXXX	Address must be multiple of 2.
USER	DATA	BYTE	RWU	00000020 XXXXXXXX	
USER	PROGRAM	LONG	RWU	00000001 XXXXXXXX	Address must be multiple of 4.
USER	PROGRAM	WORD	RWU	00000011 XXXXXXXX	Address must be multiple of 2.
USER	PROGRAM	BYTE	RWU	00000021 XXXXXXXX	
SUPERVSR	DATA	LONG	RWU	00000004 XXXXXXXX	Address must be multiple of 4.
SUPERVSR	DATA	WORD	RWU	00000014 XXXXXXXX	Address must be multiple of 2.
SUPERVSR	DATA	BYTE	RWU	00000024 XXXXXXXX	
SUPERVSR	PROGRAM	LONG	RWU	00000005 XXXXXXXX	Address must be multiple of 4.
SUPERVSR	PROGRAM	WORD	RWU	00000015 XXXXXXXX	Address must be multiple of 2.
SUPERVSR	PROGRAM	BYTE	RWU	00000025 XXXXXXXX	
USR_DEF		LONG	RWU	00000008 XXXXXXXX	Address must be multiple of 4.
USR_DEF		WORD	RWU	00000018 XXXXXXXX	Address must be multiple of 2.
USR_DEF		BYTE	RWU	00000028 XXXXXXXX	
CPU		LONG	RW	0000000C XXXXXXXX	Address must be multiple of 4.
CPU		WORD	RW	0000001C XXXXXXXX	Address must be multiple of 2.
CPU		BYTE	RW	0000002C XXXXXXXX	
UUT_ROM			RU	02000040 XXXXXXXX	"SIZE" not used, always byte.
ST_ROM			R	02000004 000XXXXX	"SIZE" not used, always byte.
OVERLAY			RWU	0200000C 0000XXXX	"SIZE" not used, always byte.

* Indicates which operations are allowed for each address option:
R = READ
W = WRITE
U = RUN UUT

Pod-Specific Setup Information E-5.

Pod-specific setup information is listed in Table E-2, along with data values, defaults, and ranges.

Table E-2. 68020 Pod Setup Parameters

POD SETUP	RANGE/KEY	DEFAULT	NOTES
INTRFACE ROM_MODS	1, 2, 4	1	
INTRFACE BCYCLCLK	SYNC_MOD, ROM_CE	SYNC_MOD	
INTRFACE RST_LEN	0-999999 (decimal)	1000	(microseconds)
INTRFACE RST_POL	LOW, HIGH	LOW	
INTRFACE XFER_ADR	0-FFFFFFFF	0	
INTRFACE CY_SPLIT	1, 2, 4	1	
INTRFACE BURST_SZ	1, 2, 4	1	
INTRFACE DATAPRB	NO, YES	NO	
INTRFACE ROM_BASE	0-FFFFF000	0	
ROM_TYPE	27256, 2716, 2732, 2764, 27128, 27512	27256	*
ROM_TYPE OTHER	0-FFFFFFFF	4201400	*
CALIBRTN ADR_STIM	0-255 (decimal)	8	
CALIBRTN RUN_UUT	0-255 (decimal)	4	

* If "ROM_TYPE OTHER" is specified, then a hexadecimal numeric value must be entered to describe the characteristics of the ROM (see Appendix A).

Table E-2. 68020 Pod Setup Parameters (cont)

POD SETUP	RANGE/KEY	DEFAULT	NOTES
RD_CAL UDATA_L	0-255 (decimal)	3	
RD_CAL UDATA_W	0-255 (decimal)	3	
RD_CAL UDATA_B	0-255 (decimal)	3	
RD_CAL UPGM_L	0-255 (decimal)	3	
RD_CAL UPGM_W	0-255 (decimal)	3	
RD_CAL UPGM_B	0-255 (decimal)	3	
RD_CAL UDEF_L	0-255 (decimal)	3	
RD_CAL UDEF_W	0-255 (decimal)	3	
RD_CAL UDEF_B	0-255 (decimal)	3	
RD_CAL SDATA_L	0-255 (decimal)	3	
RD_CAL SDATA_W	0-255 (decimal)	3	
RD_CAL SDATA_B	0-255 (decimal)	3	
RD_CAL SPGM_L	0-255 (decimal)	3	
RD_CAL SPGM_W	0-255 (decimal)	3	
RD_CAL SPGM_B	0-255 (decimal)	3	
RD_CAL CPU_L	0-255 (decimal)	3	
RD_CAL CPU_W	0-255 (decimal)	3	
RD_CAL CPU_B	0-255 (decimal)	3	
WR_CAL UDATA_L	0-255 (decimal)	3	
WR_CAL UDATA_W	0-255 (decimal)	3	
WR_CAL UDATA_B	0-255 (decimal)	3	
WR_CAL UPGM_L	0-255 (decimal)	3	
WR_CAL UPGM_W	0-255 (decimal)	3	
WR_CAL UPGM_B	0-255 (decimal)	3	
WR_CAL UDEF_L	0-255 (decimal)	3	
WR_CAL UDEF_W	0-255 (decimal)	3	
WR_CAL UDEF_B	0-255 (decimal)	3	
WR_CAL SDATA_L	0-255 (decimal)	3	
WR_CAL SDATA_W	0-255 (decimal)	3	
WR_CAL SDATA_B	0-255 (decimal)	3	
WR_CAL SPGM_L	0-255 (decimal)	3	
WR_CAL SPGM_W	0-255 (decimal)	3	
WR_CAL SPGM_B	0-255 (decimal)	3	
WR_CAL CPU_L	0-255 (decimal)	3	
WR_CAL CPU_W	0-255 (decimal)	3	
WR_CAL CPU_B	0-255 (decimal)	3	

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 an example of how to use the podsetup function
! with pod-specific setup parameters.
!
! Notes:
!
!   All character strings are case insensitive.
!
!   Pod-specific setups that consist entirely of selections from a set
!   of choices use one single-quoted argument, with the different
!   selections separated by spaces.  For example:
!
!       podsetup 'intrface rom_mods 2'
!
!   Pod-specific setups that take a numeric parameter use two arguments.
!   The first argument contains the selections leading up to the numeric
!   parameter, enclosed in single quotes.  The second argument is the
!   numeric parameter itself, unquoted.  For example:
!
!       podsetup 'intrface rst_len' 1000
!
!-------------------------------------------------------------------------

    podsetup 'intrface rom_mods 2'        ! UUT has 2 boot ROMs.

    podsetup 'intrface bcyclclk rom_ce'   ! Generate bus cycle clock
                                          ! from ROM 1 chip enable.

    podsetup 'intrface rst_len' 1000      ! Generate a 1000 uSec.
                                          ! reset.

    podsetup 'intrface rst_pol low'       ! UUT reset is active-low.

    podsetup 'intrface xfer_adr' $0       ! Transfer data using writes
                                          ! at $0.

    podsetup 'intrface cy_split 2'        ! Doubleword microprocessor
                                          ! fetches are split into 2
                                          ! boot ROM fetches.

    podsetup 'intrface burst_sz 1'        ! Boot ROMs are deselected
                                          ! between accesses.

    podsetup 'intrface dataprb no'        ! Bus Test should not ask
                                          ! user to probe D24 - D31.

    podsetup 'rom_type 27256'             ! Boot ROMs are standard
                                          ! type 27256.

    podsetup 'rom_type other' $6300400    ! Boot ROMs are non-standard
                                          ! type 231000.

    podsetup 'calibrtn adr_stim' 3        ! Address Stimulus cal = 3.

    podsetup 'calibrtn run_uut' 4         ! Run UUT calibration = 4.

    podsetup 'rd_cal udata_l' 2           ! READ USER DATA LONG cal = 2.
```

```
    podsetup 'rd_cal udata_w' 2          ! READ USER DATA WORD cal = 2.
    podsetup 'rd_cal udata_b' 2          ! READ USER DATA BYTE cal = 2.
    podsetup 'rd_cal sdata_l' 2          ! READ SUPERVISOR DATA LONG
                                         ! cal = 2.
    podsetup 'rd_cal sdata_w' 2          ! READ SUPERVISOR DATA WORD
                                         ! cal = 2.
    podsetup 'rd_cal sdata_b' 2          ! READ SUPERVISOR DATA BYTE
                                         !cal = 2.
    podsetup 'wr_cal udata_l' 2          ! WRITE USER DATA LONG cal = 2.
    podsetup 'wr_cal udata_w' 2          ! WRITE USER DATA WORD cal = 2.
    podsetup 'wr_cal udata_b' 2          ! WRITE USER DATA BYTE cal = 2.
    podsetup 'wr_cal sdata_l' 2          ! WRITE SUPERVISOR DATA LONG
                                         ! cal = 2.
    podsetup 'wr_cal sdata_w' 2          ! WRITE SUPERVISOR DATA WORD
                                         ! cal = 2.
    podsetup 'wr_cal sdata_b' 2          ! WRITE SUPERVISOR DATA BYTE
                                         ! cal = 2.
end setup
```

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

Pod Sync Calibration Data E-7.

The purpose of calibration is to insure that the probe and I/O Module sample signals on the UUT at a known point in time with respect to the UUT timing signals such as CLK and $\overline{AS}$. During calibration, delay lines in the Mainframe are adjusted so that the signals being sampled are correctly aligned in time with the clocking signal. The 9132A Pod monitors the 68020 timing signals and produces a $\overline{PODSYNC}$ signal that is sent to the Mainframe (with some delay). This $\overline{PODSYNC}$ signal corresponds to the address cycle or the data cycle of the 68020. (Figure 3-1 in this manual contains a timing diagram that shows the relationship between Pod signals and 68020 signals.)

The 68020 microprocessor uses the $\overline{AS}$ timing signal during the data cycle or during the address cycle to define basic bus cycle timing. Figure E-1 shows the relationship of UUT timing signals to $\overline{PODSYNC}$ for Data Sync mode. The calibration process determines the delay labeled "tcal", which may vary from one sync mode to another. If calibration is not performed, a default setting is used for the tcal value. After calibration is performed, the measured value for tcal replaces the default value.

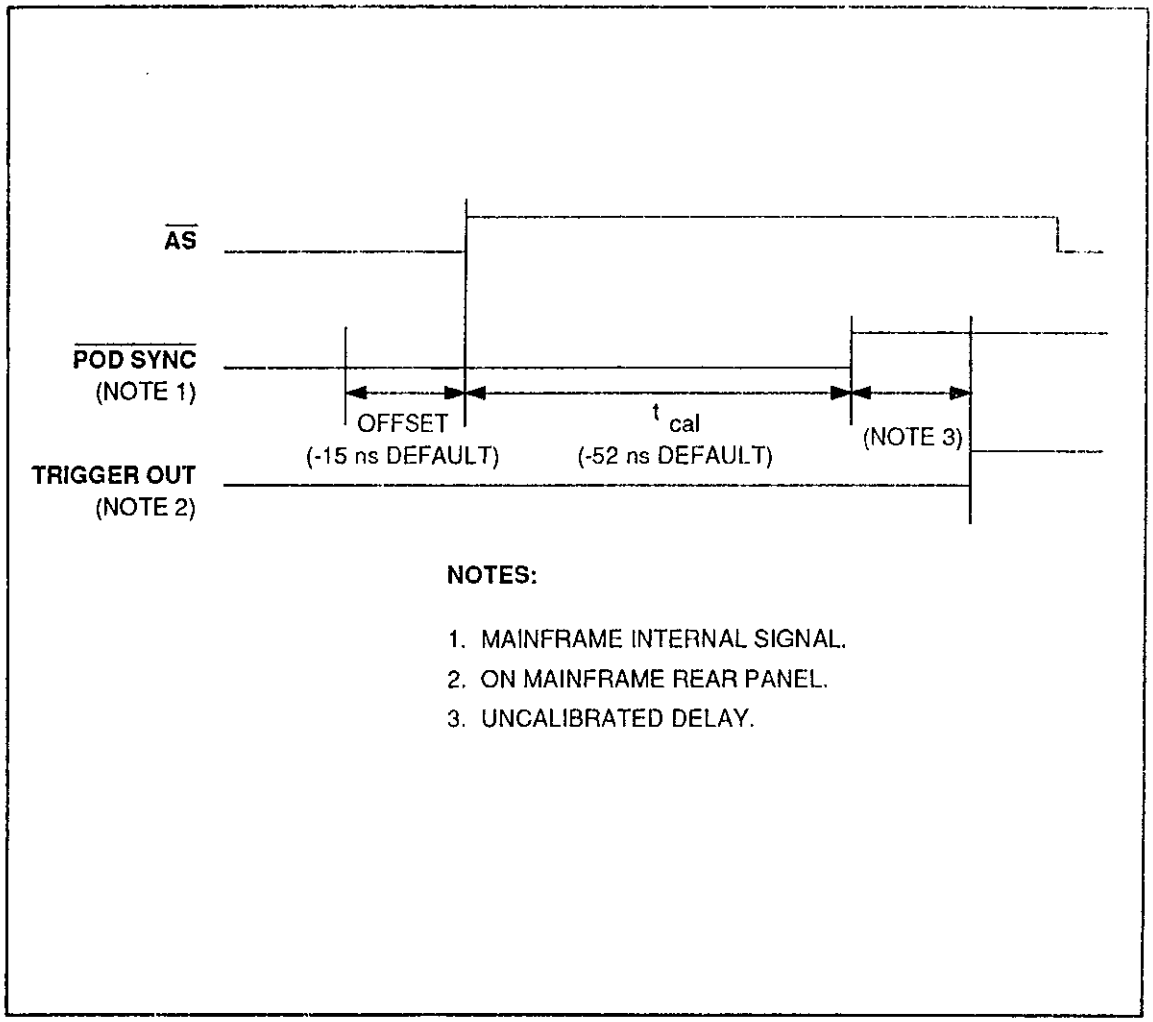


Figure E-1. 68020 Pod Data Sync Mode Calibration

To calibrate an I/O Module or probe to a 9132A-68020 Pod for a particular Pod sync mode, you are prompted to probe the 68020's $\overline{DS}$ or $\overline{AS}$ signal. For the Data Sync example shown in Figure E-1 $\overline{DS}$ has a rising edge in response to the clocked $\overline{DSACK0}$ and $\overline{DSACK1}$ signals at the microprocessor. This reference edge is used to adjust the delay lines for the probe or I/O Module signals relative to the $\overline{PODSYNC}$ signal internal to the Mainframe.

The "OFFSET" shown in Figure E-1 is a variable that allows you to move the data capture point in time with respect to the calibration point. The OFFSET value, in combination with the tcal value, determines exactly when a signal is sampled by the probe or I/O Module. Its default value is set to allow the probe or I/O Module to look at a point about 15 ns before the $\overline{AS}$ (or $\overline{DS}$) edge. Table E-3 shows the default values for each sync mode.

For greater accuracy on a specific UUT, you may calibrate the probe or I/O Module to a point on the UUT such as a ROM chip enable. In Address Sync mode the $\overline{PODSYNC}$ pulse always ends at the rising edge of $\overline{AS}$ (as shown in Figure 3-1). For UUTs with a high clock rate, you may want to calibrate Address Sync mode to the falling edge of $\overline{AS}$. In these cases, adjust the

OFFSET value using the TL/1 "setoffset" statement (to capture the address on the falling edge of $\overline{AS}$ with an OFFSET of zero.) See the "setoffset" and "getoffset" statements in the 9100-Series TL/1 Reference Manual for details. For convenience in calibrating to UUT signals, the calibration menu allows calibration to rising or falling edges. There is only one calibration each for Address Sync mode and Data Sync mode.

Table E-3. 68020 Pod Sync Calibration Data

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	$\overline{AS}$	RISING	-15 ns
DATA	$\overline{DS}$	RISING	-15 ns

Reserved Names in TL/1 Programs E-8.

A set of program names is reserved for the 68020 Pod TL/1 Support Programs that should not be used elsewhere in a TL/1 program. When any TL/1 program is called, three places are searched (in order) for a program of the correct name: the current UUT directory, then the current PODLIB, and finally the PROGLIB. As soon as a program with the correct name is found, it is executed. If a program appears with the same name as the reserved name, the wrong program may be executed.

The list of reserved names are:

HYP_RAM
QWK_RD
QWK_WR
B_TEST
B_DIAG
TEST_BUS
STIM_ADR
STIM_DAT
FRC_INT
SETUP
VERSION

Available Bus Test TL/1 Support Programs E-9.

The following list describes the available Bus Test TL/1 support programs for the 68020 Pod. For more information on fault condition and arguments, see Appendix F of the 9100-Series Technical User's Manual, and Appendix G and Appendix H of the 9100-Series TL/1 User's Manual.

NOTE

Do not use the built-in TL/1 "testbus" with the 68020 Pod. Operation with the built-in "testbus" function may produce unpredictable results.

- B_TEST

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

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

Arguments: None.

Faults: pod_misc_fault

message "ROMnPWR fault".
Bad UUT power was detected on ROM Module "n".

message "ROMnFUSE fault".
A blown fuse was detected on ROM Module "n".

message "SYNCFUSE fault".
A blown fuse was detected on ROM Module "n".

message "ROMMODn fault".
Though ROM Module "n" should exist, it was not detected.

Returns: The string "PASSED" if the test passes, or the string "FAILED" if a fault is found.

- TEST_BUS

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

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

normal fault message is raised and shown on the Mainframe display. If the CONT key on the Mainframe keypad is pressed, the program continues to diagnose further, if possible.

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

Arguments: open_already — This is set to "1" if the application display is already opened in the proper mode (B_TEST calls this routine with open_already = 1). When this program is called directly from a user's TL/1 program, this parameter should be set to "0" to let TEST_BUS open a window on the application display itself.

Faults:

pod_misc_fault
- message "ROMnPWR fault". Bad UUT power was detected on ROM Module "n".
- message "ROMnFUSE fault". A blown fuse was detected on ROM Module "n".
- message "SYNCFUSE fault". A blown fuse was detected on ROM Module "n".
- message "ROMMODn fault". Though ROM Module "n" should exist, it was not detected.

m_pod_no_reset
- message "no µP reset detected". No reset was detected by the Pod at the UUT microprocessor.

m_pod_slow_clock
- message "UUT Clock slow or stuck". The correct UUT clock signal was not detected by the Pod at the UUT microprocessor.

pod_forcing_active
- message "Forcing Signal xx pin yy is Active".

m_pod_stopped
- message "Microprocessor Stopped or Bad".

m_pod_rom1_cs
- message "ROM1 CS or OE is invalid". No chip select was seen at ROM Module 1 when the UUT microprocessor was reset.

m_pod_reset_addr
message "Bad Reset Address".
The correct reset address was not detected by ROM Module 1 after the UUT microprocessor was reset.

pod_data_tied
message "data line xx pin yy not drivable".

bus_addr_high_tied
message "addr line xx pin yy stuck high".

bus_addr_low_tied
message "addr line xx pin yy stuck low".

bus_addr_tied
message "addr line xx pin yy tied".

bus_data_high_tied
message "data line xx pin yy stuck high".

bus_data_low_tied
message "data line xx pin yy stuck low".

bus_data_tied
message "data line xx pin yy tied".

m_bus_addr_high
message "addr line xx pin yy was high, expected low".

m_bus_addr_low
message "addr line xx pin yy was low, expected high".

generic_fault
message "line stuck high".

message "line stuck low".

message "line toggling".

message "invalid level on line".

message "Warning: line always high".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

message "Warning: line always low".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

message "Warning: line toggling".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

Returns: Nothing.

- B_DIAG

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

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

Arguments: None.

Faults: pod_misc_fault

message "ROMnPWR fault".
Bad UUT power was detected on ROM Module "n".

message "ROMnFUSE fault".
A blown fuse was detected on ROM Module "n".

message "SYNCFUSE fault".
A blown fuse was detected on ROM Module "n".

message "ROMMODn fault".
Though ROM Module "n" should exist, it was not detected.

m_pod_no_reset

message "no µP reset detected".
No reset was detected by the Pod at the UUT microprocessor.

generic_fault

message "line stuck high".

message "line stuck low".

message "line toggling".

message "invalid level on line".

message "Warning: line always high".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

message "Warning: line always low".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

message "Warning: line toggling".
A warning is issued when the desired state is not certain. In most cases, the condition causing the message indicates a problem, but in other cases indicates normal operation.

Returns: Nothing.

- STIM_DAT

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

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

Arguments: DATA — The 32-bit data pattern that is placed at the reset address of the UUT microprocessor.

Faults: pod_misc_fault

message "ROMnPWR fault".
Bad UUT power was detected on ROM Module "n".

message "ROMnFUSE fault".
A blown fuse was detected on ROM Module "n".

message "SYNCFUSE fault".
A blown fuse was detected on ROM Module "n".

message "ROMMODn fault".
Though ROM Module "n" should exist, it was not detected.

m_pod_no_reset
message "no μP reset detected".
No reset was detected by the Pod at the UUT microprocessor.

m_pod_slow_clock
message "UUT Clock slow or stuck".
The correct UUT clock signal was not detected by the Pod at the UUT microprocessor.

pod_forcing_active
message "Forcing Signal xx pin yy is Active".

m_pod_stopped
message "Microprocessor Stopped or Bad".

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

- STIM_ADR

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

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

Arguments:		
	ADDR	The address at which the UUT microprocessor performs a fetch. This address must be within the UUT boot ROM address space.

Faults: pod_misc_fault
message "ROMnPWR fault".
Bad UUT power was detected on ROM Module "n".

message "ROMnFUSE fault".
A blown fuse was detected on ROM Module "n".

message "SYNCFUSE fault".
A blown fuse was detected on ROM Module "n".

message "ROMMODn fault".
Though ROM Module "n" should exist, it was not detected.

m_pod_no_reset
message "no μP reset detected".
No reset was detected by the Pod at the UUT microprocessor.

m_pod_slow_clock
message "UUT Clock slow or stuck".
The correct UUT clock signal was not detected by the Pod at the UUT microprocessor.

pod_forcing_active
message "Forcing Signal xx pin yy is Active".

m_pod_stopped
message "Microprocessor Stopped or Bad".

m_pod_rom1_cs
message "ROM1 CS or OE is invalid".
No chip select was seen at ROM Module 1 when the UUT microprocessor was reset.

m_pod_reset_addr
message "Bad Reset Address".
The correct reset address was not detected by ROM Module 1 after the UUT microprocessor was reset.

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

Other Available TL/1 Support Programs E-10.

The following list describes the other available Test TL/1 support programs for the 68020 Pod. For more information on fault condition and arguments, see Appendix F of the 9100-Series Technical User's Manual, and Appendix G and Appendix H of the 9100-Series TL/1 User's Manual.

- HYP_RAM

  This program embodies all the functions of the 68020 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 the top of boot ROM space + 4 to FFFF FFF8 and

divisible by 4 (longword space), the top of boot ROM space + 2 to FFFF FFFC and divisible by 2 (word space), or the top of boot ROM space + 1 to FFFF FFFE (byte space).

UPTO — This is the ending address of the HyperRAM test. UPTO may be any numeric value between the top of boot ROM space + 8 to FFFF FFFFC and divisible by 4 (longword space), the top of boot ROM space + 4 to FFFF FFFE and divisible by 2 (word space), or the top of boot ROM space + 2 to FFFF FFFF (byte space). UPTO's value must be greater than that of ADDR.

DELAY — This field produces a delay (in milliseconds) 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 milliseconds.

SEED — Pseudo-random number seed. If the value is 0, a different random seed is used each time the test is run. If a non-zero number is used, the sequence of random numbers is the same for each test. The default value is the numeric value 0.

Faults: test_aborted

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 "Space not supported".
The current address space that the Mainframe is in is not supported by the program.

reason "Illegal address range"
The stop address was less than the start address.

reason "Unknown status code $xx from pod"
Indicates problems sending data from the UUT to the Pod. Should not be seen in normal operation.

Returns: Nothing.

- QWK_RD

This program causes the Pod to implement a quick looping read function. Once QWK_RD is entered, the program returns at once with the value of the data found at the given address. Though only one value is returned, the Pod continues to perform reads at the specified address. A sync pulse is generated for each read, as specified by the current sync mode. Reading continues until a Pod access of any kind is initiated.

Arguments: ADDR — The address the Pod repeatedly reads.

Returns — The data at the specified address.

NOTE

Looping on the QWK_RD function may result in the loss of UUT hardware initialization. See Appendix G for more information.

- QWK_WR

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

Arguments: ADDR — The address the Pod repeatedly writes.

DATA — The value of the data to be written at the specified address.

Returns: Nothing.

NOTE

Looping on the QWK_WR function may result in the loss of UUT hardware initialization. See Appendix G for more information.

- FRC_INT

  This program forces an interrupt acknowledge cycle and returns the interrupt vector found on the data bus. (Interrupt acknowledge cycles are simulated by read accesses to CPU BYTE spaces as described under the heading, Simulating Interrupt Acknowledge, in Section 3 of this manual.) The vector data may be meaningless.

  Arguments: LEVEL — This parameter specifies the level of the interrupt acknowledge. LEVEL may be any value between 0 and 7.

9132A Pod Special Faults (pod_special) E-11.

All Pod Special faults are considered fatal, since they prevent the Pod from completing the current command. Because of this, Pod Special errors should not be disabled with the SETUP function.

The following Pod Special faults can be generated by the 9132A Pod. The number in the left-hand column is the "index" argument which is passed to the fault handler.

0 LOST CONTROL OF UUT MICROPROCESSOR

After sending a command to the microprocessor in the UUT, the Pod has received either an incorrect response or no response at all. This normally indicates one or more of the following problems:

- The ROM Modules are not plugged into the correct boot ROM sockets.
- The Sync Module is not properly connected to the UUT.
- One or more Pod setup selections are incorrect for the UUT.
- The UUT has a kernel fault that prevents Pod operations from functioning correctly. (Use the TEST BUS function to diagnose this type of fault.)
- The UUT microprocessor is defective.

1 UUT HAD AN INTERRUPT OR EXCEPTION

The UUT's microprocessor has received an unexpected non-maskable interrupt since executing the previous command. If this error persists, it may be necessary to disable the 68020 NMI source by connecting it to a non-asserted level. In rare cases, this fault can also be reported as a result of certain types of UUT kernel faults.

2 INTERNAL COMMAND ERROR

The UUT's microprocessor reports that it has received an invalid command from the Pod. This error is not expected during normal operation of the Pod. However, certain types of UUT kernel faults can cause this error to be reported. Use the TEST BUS function to diagnose the UUT kernel if this fault is reported.

3 INTERNAL ADDRESS ERROR

The UUT's microprocessor reports that it has received an invalid address from the Pod. This error is not expected during normal operation of the Pod. However, certain types of UUT kernel faults can cause this error to be reported. Use the TEST BUS function to diagnose the UUT kernel if this fault is reported.

4 UUT WAS RESET

The UUT's microprocessor reports that it has unexpectedly been reset since executing the previous command. If this error persists, it may be necessary to disable any UUT logic (for example, a watchdog timer) that might be producing undesired resets. In rare cases, this fault can also be reported as a result of certain types of UUT kernel faults.

5 DATA TRANSFER AT XFER_ADR FAILED

The Pod was unable to reliably transfer data from the UUT using the address specified in the INTRFACE XFER_ADR Pod setup selection (this transfer returns data after a UUT READ). This fault usually means the UUT microprocessor cannot write data to the given address. The fault can be caused by an incorrect or nonwritable XFER_ADR, or by a UUT kernel fault. If the XFER_ADR is determined to be correct, use the TEST BUS function to diagnose the UUT kernel.

6 INVALID HYPERRAM PARAMETER

One or more parameters for the HyperRAM test were given invalid values. In normal use, the HYP_RAM program checks all parameters and gives more detailed error messages if any of them are invalid. However, this fault may be reported if the HyperRAM test is invoked directly using RUN UUT VIRTUAL.

7 UUT MICROPROCESSOR HALTED

The Pod has detected that the UUT's microprocessor is not producing any bus cycles. This can be caused by any of the following problems:

- The ROM Modules are not plugged into the correct boot ROM sockets.
- The Sync Module is not properly connected to the UUT.
- One or more Pod setup selections are incorrect for the UUT.
- The UUT has a kernel fault which prevents Pod operations from functioning correctly. (Use the TEST BUS function to diagnose this type of fault.)
- The UUT microprocessor is defective.

8 ROM MODULE NOT IN SELFTEST SOCKET

An access has been attempted using the ST_ROM address option, but no ROM module is correctly inserted into the Pod self-test socket. Verify that the appropriate ROM module is oriented properly and is inserted fully into the self-test socket. If this error persists, check the fuse in the ROM module.

9 ROM MODULE / ROM TYPE MISMATCH

The ROM modules installed in the Pod have an inappropriate number of pins for the type of ROM that has been selected using the ROM_TYPE Pod setup item. Correct the ROM_TYPE selection, or install the proper type of ROM module.

10 INTERNAL RESPONSE ERROR

After sending a command to the microprocessor in the UUT, the Pod has received a response that is invalid. This error is not expected during normal operation of the Pod. However, certain types of UUT kernel faults can cause this error to be reported during the HyperRAM test. Use the TEST BUS function to diagnose the UUT kernel if this fault is reported.

Bit Definitions of Fault Masks E-12.

Tables E-4 through E-9 describe the mapping from specific 68020 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.

NOTE

Using the leading zeroes in the fault masks is no longer necessary, though programs will function correctly with the zeroes inserted. Masks of less than 64 characters are assumed to be right justified. The leading zeroes are displayed in these tables to demonstrate backward compatibility.

Table E-4. Address Signal Mapping to Fault Masks

Address signal fault mask format:

"00000000000000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

Address signal fault mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
0	A0	C4
1	A1	A2
2	A2	E12
3	A3	D13
4	A4	D12
5	A5	C13
6	A6	B13
7	A7	C12
8	A8	A13
9	A9	C11
10	A10	B12
11	A11	A12
12	A12	C10
13	A13	B11
14	A14	A11
15	A15	B10
16	A16	C9
17	A17	C8
18	A18	B8
19	A19	A8
20	A20	B7
21	A21	C7
22	A22	A7
23	A23	A6
24	A24	B6
25	A25	C6
26	A26	A5
27	A27	B5
28	A28	A4
29	A29	C5
30	A30	B4
31	A31	A3

Table E-5. Data Signal Mapping to Fault Masks

Data signal fault mask format:

"00000000000000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

Data signal fault mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
0	D0	K13
1	D1	K12
2	D2	L13
3	D3	L12
4	D4	M13
5	D5	M12
6	D6	M11
7	D7	L10
8	D8	N12
9	D9	N11
10	D10	M10
11	D11	L9
12	D12	N10
13	D13	M9
14	D14	N9
15	D15	L8
16	D16	M7
17	D17	N6
18	D18	M6
19	D19	L6
20	D20	N5
21	D21	M5
22	D22	N4
23	D23	L5
24	D24	M4
25	D25	N3
26	D26	M3
27	D27	L4
28	D28	N2
29	D29	M2
30	D30	L3
31	D31	N1

Table E-6. Interrupt Signal Mapping to Fault Masks

Interrupt fault mask format:

"00000000000000000000000000000000000000000000000000XXX0000000000"

Interrupt mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
10	$\overline{IPL0}$	J12
11	$\overline{IPL1}$	J13
12	$\overline{IPL2}$	H12

Table E-7. Miscellaneous Signal Mapping to Fault Masks

Miscellaneous fault mask format:

"00000000000000000000000000000000000000000000XXXXXXXXXXXXXX000000XX"

Miscellaneous mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
0	CLK	C2
1	$\overline{\text{RESET}}$	C1
8	ROM1PWR	(No power at ROM Module 1)
9	ROM2PWR	(No power at ROM Module 2)
10	ROM3PWR	(No power at ROM Module 3)
11	ROM4PWR	(No power at ROM Module 4)
12	ROM1FUSE	(ROM Module 1 fuse blown)
13	ROM2FUSE	(ROM Module 2 fuse blown)
14	ROM3FUSE	(ROM Module 3 fuse blown)
15	ROM4FUSE	(ROM Module 4 fuse blown)
16	SYNCFUSE	(Sync Module fuse blown)
17	ROMMOD1	(ROM Module 1 not present)
18	ROMMOD2	(ROM Module 2 not present)
19	ROMMOD3	(ROM Module 3 not present)
20	ROMMOD4	(ROM Module 4 not present)
21	PERMOD	(Personality Module not present)

Table E-8. Control Signal Mapping to Fault Masks

Control fault mask format:

"00000000000000000000000000000000000000000000XXXXXXXXXXXXX0X0X0XX0"

Control mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
1	$\overline{\text{RESET}}$	C1
2	$\overline{\text{AS}}$	L1
4	$\overline{\text{BG}}$	B2
6	$\overline{\text{HALT}}$	K2
8	$\overline{\text{RMC}}$	E2
9	$\overline{\text{IPEND}}$	F13
10	FC0	E1
11	FC1	F3
12	FC2	F2
13	SIZ0	F1
14	SIZ1	G2
15	R/$\overline{\text{W}}$	L2
16	$\overline{\text{DS}}$	M1
17	$\overline{\text{DBEN}}$	G3
18	$\overline{\text{ECS}}$	G1
19	$\overline{\text{OCS}}$	E13

Table E-3. Forcing Signal Mapping to Fault Masks

Forcing fault mask format:

"00000000000000000000000000000000000000000000000000000XXXXX0X0X0"

Forcing mask bit assignments:

MASK BIT	68020 SIGNAL	68020 PIN NUMBER
1	$\overline{\text{RESET}}$	C1
3	$\overline{\text{BR}}$	B3
5	$\overline{\text{BGACK}}$	A1
6	$\overline{\text{HALT}}$	K2
7	$\overline{\text{BERR}}$	J2
8	$\overline{\text{DSACK0}}$	H3
9	$\overline{\text{DSACK1}}$	J1

The following program summarizes the information in the previous tables and provides examples with TL/1 format and syntax.

```
program faults
!-------------------------------------------------------------------------
!
! This program shows examples of raising each of the built-in primitive
! faults requiring specific mask arguments associated with the 9132A-
! 68020 pod.
!
! When executed, this program stops as each fault is reported. Enter
! "CONT" to display successive faults.
!
!-------------------------------------------------------------------------
    !
    ! Raise built-in address drivability fault on A7.
    !
    fault pod_addr_tied mask "10000000"
    !
    ! Raise built-in data drivability fault on D11.
    !
    fault pod_data_tied mask "100000000000"
    !
    ! Raise built-in control drivability fault on ~AS.
    !
    fault pod_ctl_tied mask "100"
    !
    ! Raise built-in active forcing line fault on ~HALT.
    !
    fault pod_forcing_active mask "1000000"
    !
    ! Raise built-in active interrupt fault on IPL1.
    !
    fault pod_interrupt_active mask "100000000000"
    !
    ! Raise built-in miscellaneous fault on RESET.
    !
    fault pod_misc_fault mask "10"
end faults
```

RUN UUT PROGRAM EXAMPLES E-13.

The following examples demonstrate the different forms the "runuut" statement can take when using the 68020 Pod within TL/1 programs.

Runuut at the reset address (default):

```
setspace space (getspace space "SUPERVSR", type "PROGRAM",
   size "BYTE")
runuut addr $0
```

Runuut in the boot ROM space:

```
setspace space (getspace space "SUPERVSR", type "PROGRAM",
   size "BYTE")
runuut addr $560
```

Runuut with Overlay Memory enabled:

```
setspace space (getspace space "OVERLAY")
runuut addr $72C
```

Runuut with breakpoint specified:

```
setspace space (getspace space "SUPERVSR", type "PROGRAM",
   size "BYTE")
runuut addr $496, break $61A
```

POD VIRTUAL ADDRESSES E-14.

Table E-10 lists Pod virtual addresses that are accessed using the READ VIRTUAL and WRITE VIRTUAL functions. (These functions are not intended for general use, but appear in Fluke-provided TL/1 programs.)

Table E-10. Read and Write Virtual Addresses

EXTADDR	ADDR	DESCRIPTION
02000000	00000000 *	UUT reset length in microseconds
02000000	00000004 *	UUT reset polarity: 0 means LOW 1 means HIGH
02000000	00000008 *	Number of ROM modules: 0 means 1 ROM module 1 means 2 ROM modules 2 means 4 ROM modules
02000000	0000000C	Do not use
02000000	00000010	Do not use
02000000	00000014	Used by Bus Test
02000000	00000018	Used by Bus Test
02000000	00000028	Used by Bus Test
02000000	0000002C	Used by Bus Test
02000000	00000044 *	Bus cycle clock source: 0 means SYNC_MOD 1 means ROM_CE
02000000	00000048	Do not use
02000000	0000004C	Transfer calibration
02000000	00000050 *	Transfer address (XFER_ADR)
02000000	00000058 *	Burst size: 0 means Burst Size = 1 1 means Burst Size = 2 2 means Burst Size = 4
02000000	00000060	High-order 32 bits of virtual address for HyperRAM
02000000	00000064	Low-order 32 bits of start address for HyperRAM
02000000	00000068	Low-order 32 bits of stop address for HyperRAM
02000000	0000006C	Delay in milliseconds for HyperRAM

Table E-10. Read and Write Virtual Addresses (cont)

EXTADDR	ADDR	DESCRIPTION
02000000	00000070	Status returned by HyperRAM: 0 means the test passed 1 means the test failed
02000000	00000074	Failing address returned by HyperRAM
02000000	00000078	Expected data at HyperRAM failing address
02000000	0000007C	Actual data at HyperRAM failing address
02000000	00000080 *	Address Stimulus sync calibration value (ADR_STIM)
02000000	00000084	Used by Bus Test
02000000	00000088	Used by Bus Test
02000000	0000008C	Used by Bus Test
02000000	00000090	Used by Bus Test
02000000	00000094	Used by Bus Test
02000000	00000098	Used by Bus Test
02000000	0000009C	Do not use
02000000	000000A0 *	Numeric descriptor for currently selected ROM_TYPE
02000000	000000A4	Pod software version number: xxyy, where xx is the major revision number, and yy is the minor revision number
02000000	000000A8	Personality module version number: xxyy, where xx is the major revision number, and yy is the minor revision number
02000000	000000B4 *	Cycle split: 0 means Cycle Split = 1 1 means Cycle Split = 2 2 means Cycle Split = 4
02000000	000000B8	Bytes of unallocated memory available in the Pod
02000000	000000BC	Do not use
02000000	000000C0 *	DATAPRB: 0 means NO, do not probe data lines in Bus Test 1 means YES, probe data lines in Bus Test
02000000	000000C4	Run UUT sync calibration value (RUN_UUT)
02000000	000000CB	Breakpoint virtual address, low-order 32 bits
02000000	000000CC	Breakpoint virtual address, high-order 32 bits
02000000	000000D0	Breakpoint enabled flag: 0 means breakpoint is not enabled 1 means breakpoint is enabled
02000000	000000D4 *	Used by Bus Test
02000000	000000D8	Used by Bus Test
02000000	000000DC	Used by Bus Test
02000000	000000E0	Used by Bus Test
02000000	000000E4	Used by Bus Test
02000000	000000E8	Do not use
02000000	000000EC	Do not use
02000000	000000F0	Writing any value to this address causes the Pod to enter fast looping read or write mode, continually repeating the most recent READ or WRITE access to the UUT. This continues until the next Pod access.
02000000 through 02000000	00000100 through 0000013C	Do not use
02000000	00000200 *	Sync calibration for READ USER DATA LONG
02000000	00000204 *	Sync calibration for READ USER DATA WORD
02000000	00000208 *	Sync calibration for READ USER DATA BYTE
02000000	0000020C *	Sync calibration for READ USER PROGRAM LONG
02000000	00000210 *	Sync calibration for READ USER PROGRAM WORD
02000000	00000214 *	Sync calibration for READ USER PROGRAM BYTE
02000000	00000218 *	Sync calibration for READ USER DEFINED LONG
02000000	0000021C *	Sync calibration for READ USER DEFINED WORD
02000000	00000220 *	Sync calibration for READ USER DEFINED BYTE
02000000	00000224 *	Sync calibration for READ SUPERVISOR DATA LONG
02000000	00000228 *	Sync calibration for READ SUPERVISOR DATA WORD
02000000	0000022C *	Sync calibration for READ SUPERVISOR DATA BYTE
02000000	00000230 *	Sync calibration for READ SUPERVISOR PROGRAM LONG
02000000	00000234 *	Sync calibration for READ SUPERVISOR PROGRAM WORD
02000000	00000238 *	Sync calibration for READ SUPERVISOR PROGRAM BYTE
02000000	0000023C *	Sync calibration for READ CPU LONG
02000000	00000240 *	Sync calibration for READ CPU WORD

Table E-10. Read and Write Virtual Addresses (cont)

EXTADDR	ADDR	DESCRIPTION
02000000	00000244 *	Sync calibration for READ CPU BYTE
02000000	00000300 *	Sync calibration for WRITE USER DATA LONG
02000000	00000304 *	Sync calibration for WRITE USER DATA WORD
02000000	00000308 *	Sync calibration for WRITE USER DATA BYTE
02000000	0000030C *	Sync calibration for WRITE USER PROGRAM LONG
02000000	00000310 *	Sync calibration for WRITE USER PROGRAM WORD
02000000	00000314 *	Sync calibration for WRITE USER PROGRAM BYTE
02000000	00000318 *	Sync calibration for WRITE USER DEFINED LONG
02000000	0000031C *	Sync calibration for WRITE USER DEFINED WORD
02000000	00000320 *	Sync calibration for WRITE USER DEFINED BYTE
02000000	00000324 *	Sync calibration for WRITE SUPERVISOR DATA LONG
02000000	00000328 *	Sync calibration for WRITE SUPERVISOR DATA WORD
02000000	0000032C *	Sync calibration for WRITE SUPERVISOR DATA BYTE
02000000	00000330 *	Sync calibration for WRITE SUPERVISOR PROGRAM LONG
02000000	00000334 *	Sync calibration for WRITE SUPERVISOR PROGRAM WORD
02000000	00000338 *	Sync calibration for WRITE SUPERVISOR PROGRAM BYTE
02000000	0000033C *	Sync calibration for WRITE CPU LONG
02000000	00000340 *	Sync calibration for WRITE CPU WORD
02000000	00000344 *	Sync calibration for WRITE CPU BYTE
02000000	00002000	
through		
02000000	00003FFC	UUT address trace, used by Bus Test
All others		Do not use

* These locations are manipulated by the Mainframe SETUP function. If a write is performed at these locations, the Mainframe - Pod setup-state synchronization will be lost.

Table E-11 lists Pod virtual addresses that are accessed by the RUN UUT VIRTUAL function.

Table E-11. RUN UUT Virtual Addresses

EXTADDR	ADDR	DESCRIPTION
02000008	00000000	Executes go/no-go Bus Test
02000008	00000001	Executes Data Stimulus
02000008	00000002	Executes Address Stimulus
02000008	00000004	Executes Run UUT Calibration
02000008	00000005	Executes go/no-go Data Bus Test
02000008	00000100	Executes HyperRAM Test

68020 PART LIBRARY FOR GFI APPLICATIONS E-15.

The 9132A-68020 Master Userdisk contains a basic part description of the 68020 microprocessor in the file /HDR/PARTLIB/68020 (PART). This file is a starting point to include the 68020 part in a GFI database.

Appendix F
Self Test Failure Codes

Table F-1 contains a list of failure codes displayed by the Mainframe when the Pod Self Test fails. The numbers displayed by the Mainframe correspond to the type of failure described in the table.

Table F-1. Pod Self Test Failure Codes

FAILURE CODE	DESCRIPTION
1001	ROM module not in selftest socket.
1002	ROM power sense fault.
1003	RAM Module fault.
1004	Cannot arm the comparator.
1005	Comparator would not fire from clock on pin 22.
1006	Comparator would not fire from clock on pin 24.
1007 to 1028	ROM Module 1 pin fault.
1029	ARAM fault.
1030	RAM Module - force A and B fault.
1031	RAM Module - force A fault.
1032	RAM Module - force B fault.
1033	RAM Module - swap A fault.
1034	RAM Module - swap B fault.
1035	ROM module address or data path fault.
1036	Cannot determine ROM plug size.
1037	Bad address comparator output with all inputs set to don't care.
1038	Sync module data line 0 tied low.
1039	Sync module data line 1 tied low.
1040	Sync module data line 2 tied low.
1041	Sync module data line 3 tied low.
1042	Sync module data line 4 tied low.
1043	Sync module data line 5 tied low.
1044	Sync module data line 6 tied low.
1045	Sync module data line 7 tied low.
1046	Sync module sync chan. 0 tied low.
1047	Sync module sync chan. 1 tied low.
1048	Sync module sync chan. 2 tied low.
1049	Sync module sync chan. 3 tied low.
1050	Sync module sync chan. 4 tied low.
1051	Sync module sync chan. 5 tied low.
1052	Sync module sync chan. 6 tied low.
1053	Sync module sync chan. 7 tied low.
1054	Sync module data line 0 tied high.

Table F-1. Pod Self Test Failure Codes (cont)

FAILURE CODE	DESCRIPTION
1055	Sync module data line 1 tied high.
1056	Sync module data line 2 tied high.
1057	Sync module data line 3 tied high.
1058	Sync module data line 4 tied high.
1059	Sync module data line 5 tied high.
1060	Sync module data line 6 tied high.
1061	Sync module data line 7 tied high.
1062	Sync module sync chan. 0 tied high.
1063	Sync module sync chan. 1 tied high.
1064	Sync module sync chan. 2 tied high.
1065	Sync module sync chan. 3 tied high.
1066	Sync module sync chan. 4 tied high.
1067	Sync module sync chan. 5 tied high.
1068	Sync module sync chan. 6 tied high.
1069	Sync module sync chan. 7 tied high.
1070	Sync module data and/or sync lines tied.
1071	Microprocessor reset detector fault.
1072	Bad address sync clock from Personality Module.
1073	Bad data sync clock from Personality Module.
1074	Reset overdrive failed in attempting to drive low.
1075	Reset overdrive failed in attempting to drive high.
1076	Sync Channel 3 overdrive turned on when it should have been off.
1077	Sync Channel 3 overdrive failed to turn on when it should have.
1078	Bad Personality Module ROM.
2001	ROM or Sync Module not in self test socket. If in self test socket, check for blown fuse in ROM or Sync Module.
2002	Unexpected powerfail.
2003	No powerfail when expected.
2004	Pod does not report ABORT when ABORT is asserted.
2005	Pod reports ABORT unexpectedly.
2008	Pod responds unexpectedly when setting fault mask.
2009	Pod fails SYNC test/select.

APPENDIX G
Reset Connection

INTRODUCTION G-1.

The Sync Module UUT RESET line can be connected to any of several different points on an 68020-based UUT. This Appendix discusses some general guidelines. Resetting as much of the UUT as possible is desirable for functional test applications. Resetting as little of the UUT as possible is more convenient.

The Sync Module UUT RESET line generally cannot be directly clipped to the 68020 microprocessor. It should be connected to a point that allows the microprocessor $\overline{\text{RESET}}$ input to be synchronized to the CLK signal. It should also allow synchronization of the microprocessor CLK and of any UUT bus controller circuits that use CLK.

NOTE

Connection to a power supply monitor line (PWR GOOD) usually causes the Pod to reset the entire UUT, but on some power supplies can cause the power supply to conduct noise into UUT circuitry or cause the power supply to temporarily shut down. This may result in multiple resets at the microprocessor, and cause Bus Test failure.

FUNCTIONAL TEST CONSIDERATIONS G-2.

For a full functional test, the best connection position is one that resets as much of the UUT as possible, like a full system (i.e., power-up) reset. This connection allows the test to start as closely as possible to actual UUT reset start-up conditions. Since this connection usually allows UUT hardware to clear a fault condition which the software cannot (e.g., a $\overline{\text{BR}}$ request stuck on), this position is also the "safest."

Since this connection resets as much of the system as possible, the UUT probably requires initialization of hardware, such as dynamic RAM refresh controllers, after each RESET. The hardware initialization routines should be saved as programs and/or stored key sequences.

CONVENIENCE CONSIDERATIONS G-3.

For convenience, it is often preferable to connect the Sync Module RESET line to a point on the UUT at which the Pod can reset the microprocessor and

as little else as possible. With this method, it may not be necessary to reinitialize the hardware after each reset. This method allows you to run certain tests that might not be possible with a full UUT reset, such as:

- Using RUN UUT to initialize UUT peripheral chips. This is useful when first designing tests on a known good board, but is not a reliable method for functional tests or troubleshooting. A hardware fault may prevent RUN UUT operation from initializing the UUT properly.
- Leaving dynamic RAM refresh in operation after exiting from RUN UUT. This allows you to read test results in RAM after leaving RUN UUT.
- Leaving some peripherals initialized when entering RUN UUT, so later RUN UUT operations do not need to reinitialize them (the Pod performs a UUT reset on exit from RUN UUT).

TEST CONDITIONS THAT CAUSE A RESET G-4.

Certain tests conducted by the Pod reset the UUT. Once a reset occurs, some UUTs may require various components to be initialized before testing of the UUT can continue (for instance, the dynamic RAM controller on the UUT may need to be initialized). The tests and conditions that cause a UUT reset are:

- Bus Test.
- STIM_ADR.
- STIM_DAT.
- Entering RUN UUT.
- Exiting RUN UUT.
- At the first read, write, or HyperRAM access to the UUT after any of the above conditions, after a QWK_RD or QWK_WR, after UUT power has been removed and restored, or after certain types of UUT faults (e.g., "Lost control of UUT μP" or "active forcing line").

INDEX

U.S.GPO 570-000P.0.42307 5/91-100