NOTE

This manual documents the Model 9000A-80188 and its assemblies at the revision levels identified in Section 7. If your instrument contains assemblies with different revision letters, it will be necessary for you to either update or backdate this manual. Refer to the supplemental change/errata sheet for newer assemblies, or to the backdating information in Section 7 for older assemblies.

9000A-80188
Interface Pod

Instruction Manual

P/N 738005
AUGUST 1985

FLUKE®

WARRANTY

COVERAGE

Fluke warrants the 9000A-80186 Interface Pod to be free from defects in material and workmanship under normal use and service for a period of one (1) year and the Pod Cable for ninety (90) days from the date of shipment. This warranty extends only to the original purchaser and does not apply to any product that has been misused, altered, or has been subjected to abnormal conditions of operation.

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

SERVICE

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

DISCLAIMER

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

GETTING ANSWERS AND ADVICE

To enhance your use of this Pod, Fluke will be happy to answer you questions about applications and use. Address all correspondence to: JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WASHINGTON 98206, ATTN: Sales Department. European customers should contact FLUKE (Holland) B.V., P.O. BOX 5053, 5004 EB, TILBURG, THE NETHERLANDS.

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

Table of Contents

(continued on page ii)

TABLE OF CONTENTS, *continued*

(continued on page iii)

TABLE OF CONTENTS, *continued*

(continued on page iv)

TABLE OF CONTENTS, *continued*

List of Tables

List of Illustrations

Section 1
Introduction

THE PURPOSE OF THE INTERFACE POD 1-1.

The 9000A-80188 Interface Pod allows you to use any Fluke 9000-Series Micro-System Troubleshooter to troubleshoot equipment that uses an 80188 microprocessor.

The Micro-System Troubleshooter (referred to hereafter as the Troubleshooter) is used to service printed circuit boards, instruments and systems that use microprocessors. The 9000A-80188 Interface Pod (referred to as the Pod) replaces the 80188 microprocessor in the UUT and serves both as an interface to allow the Troubleshooter access to components on the UUT and as an emulator of the UUT's microprocessor.

In normal Troubleshooter/Pod operation, the Pod adapts the general-purpose architecture of the Troubleshooter to the specific pin layout of the 80188 microprocessor. This allows the Troubleshooter to exercise each of the microprocessor's signals and provides complete access to devices on the UUT that are connected to the microprocessor's bus, while at the same time monitoring activity on the UUT.

In the RUN UUT mode, the Pod's microprocessor is connected to the UUT (through buffers) to serve as a substitute microprocessor. It can be configured to have the same structure of interrupts, chip selects, timers, and DMA control as the UUT's microprocessor.

NOTE

It is assumed that the user of this manual is familiar with the basic operation of one of the 9000-Series Micro-System Troubleshooters.

PHYSICAL DESCRIPTION OF THE POD 1-2.

The Pod connects to the Troubleshooter through a round shielded cable, and connects to the unit-under-test (referred to as the UUT) through a ribbon cable and plug that's inserted into the UUT's microprocessor socket. The UUT's microprocessor is removed from the UUT and is replaced by the Pod ribbon cable plug.

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

The Pod consists of a pair of printed circuit board assemblies mounted within a break-resistant case. A clock-generator module is located near the end of the ribbon cable. The Pod contains the control software and supporting hardware that is required to do the following:

- Perform handshaking with the Troubleshooter.
- Receive and execute commands from the Troubleshooter.
- Report the condition of the UUT to the Troubleshooter.
- Exercise the UUT.

An 80188 microprocessor in the Pod performs all of the functions that are normally required by the UUT, and performs the Pod's functions as well. Figure 1-2 shows the communication between the Pod, the Troubleshooter, and the UUT.

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

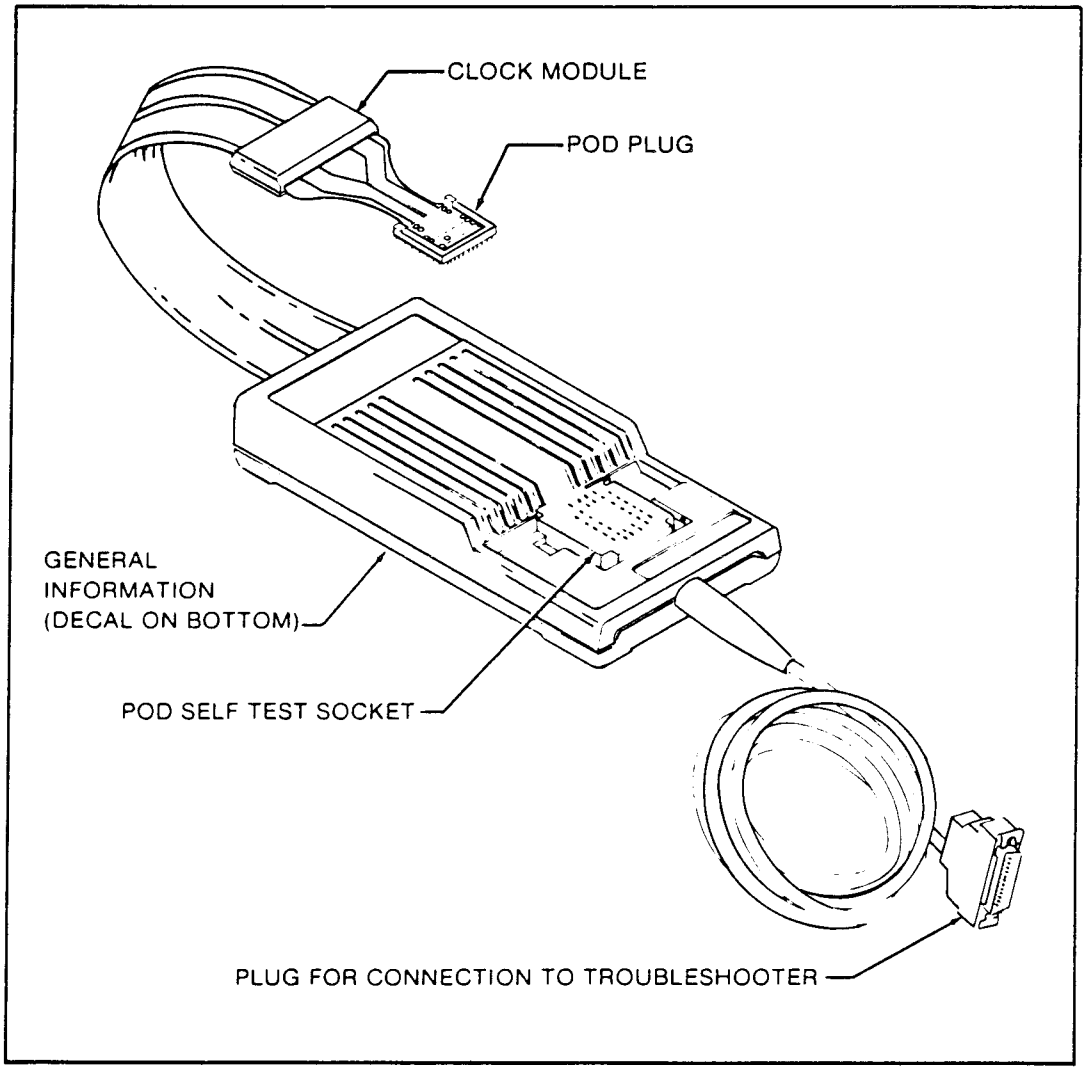


Figure 1-1. External Features of the 80188 Interface Pod

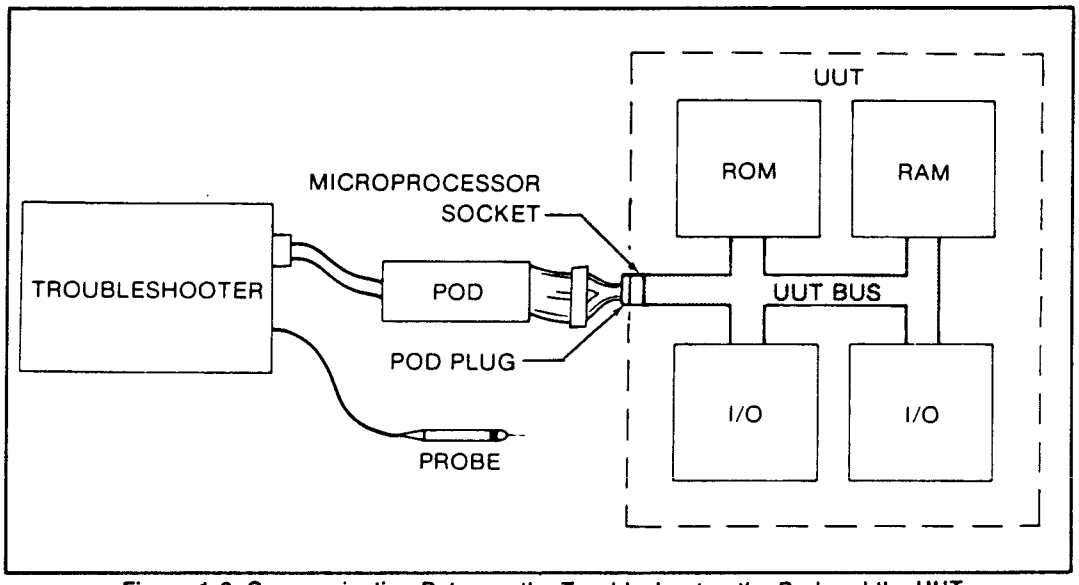


Figure 1-2. Communication Between the Troubleshooter, the Pod and the UUT

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

Over-voltage protection circuits on each line to the UUT (except the clock lines X1 and X2) guard against Pod damage that could result from the following:

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

A power-level sensing circuit monitors the voltage level of the UUT power supply. If UUT power rises above or drops below an acceptable level, the Pod notifies the Troubleshooter of a power-fail condition.

A self-test socket on the Pod enables the Troubleshooter to check Pod operation. The ribbon cable plug is connected to the self-test socket during self test operation, which enables the built-in self-test function and allows the Troubleshooter to investigate the Pod's internal functions.

CAUTION

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

POD SPECIFICATIONS 1-3.

Specifications for the Pod are listed in Table1-1.

USING THIS MANUAL 1-4.

This manual provides complete information for using the 80188 Pod, including installation and setup, operating, and troubleshooting instructions. The summary below explains briefly what kind of information is available in each of the sections:

Section 2 Contains installation instructions for connecting the Pod to the UUT and to the Troubleshooter and using the built-in self test to ensure that the Pod is working correctly. It also contains descriptions of the addressing scheme that is used to access both UUT addresses and special Pod Functions, and of the mapping system used by the Pod to identify status and control lines. Use the information in this section when you're first connecting the Pod, when you're performing periodic self tests on it, and when you're first becoming acquainted with the Pod before learning to use the Pod Functions.

Section 3 Provides descriptions of the 80188 signals, as they are implemented by the Pod. Use this Section to familiarize yourself with how the Pod handles the 80188's signals, then use this information as a reference. If you're already familiar with the 80188 microprocessor, you should still read this section to learn how its signals are handled by the Pod.

Section 4 Contains operating information for using the Pod and all of its built-in diagnostic functions and tests to troubleshoot UUTs. You will need to read the descriptions and examples in this section to learn how to use the Pod. The material is broken into two subsections:

Section 4A Describes Pod Functions that are used to access devices on the UUT.

Section 4B Describes Pod Functions that are used primarily for configuring the Pod.

Section 5 Theory of Operation information which you can read to get an understanding of how the Pod works and a background for extended troubleshooting of the Pod itself, if necessary.

Section 6 Procedures for diagnosing and correcting failures in the Pod. If you have a Pod failure, use the procedures in Section 6 (with the schematics in Section 8) to help locate the problem.

Section 7 Parts lists and information to use when ordering replacement parts for your Pod.

Section 8 Schematic diagrams to use if it becomes necessary to troubleshoot an inoperative Pod.

Appendices The Appendices contain miscellaneous material which may prove valuable when using the Pod, such as information about default values, reset procedures, and remote programming.

Table 1-1. 9000A-80188 Pod Specifications

ELECTRICAL PERFORMANCE		
Power Dissipation	10 watts max.	
Maximum External Voltage	Pin 59 (X1):	-1.5V to +6.5V
	Pin 58 (X2):	0V to +5V
	All other pins:	-7V to +12V
	Voltages listed above are continuous and are referenced to ground.	
MICROPROCESSOR SIGNALS*		
Clock Input X1 (Pin 59):		
Input Low Voltage	0.9V max.	
Input High Voltage	3.7V min.	
Input Current	1.0 μA max.	
All other signals:		
Input Low Voltage	0.8V max.	
Input High Voltage	2.0V min.	
Output Low Voltage	0.5V max. at rated current	
Output High Voltage	2.4V min. at -400 μa	
Tristate Output Leakage Current	±0.02 mA typical, +0.1 to -0.2 mA max.	
High-Level Input Current	20 μA max.	
Low-Level Input Current	-500 μA max.	
TIMING CHARACTERISTICS		
Maximum External Clock Frequency	8.0 MHz typical	
Insertion Delays to 80188 Signals		
INPUT SIGNALS	12 ns typical	
OUTPUT SIGNALS		
High-to-Low Transitions	24 ns typical	
Low-to-High Transitions	20 ns typical	
UUT POWER DETECTION		
Detection of Low Vcc Fault	Vcc<+4.5V	
Detection of High Vcc Fault	Vcc>+5.5V	
Pod Protection from UUT Low Power	Vcc<+3.4V**	
GENERAL		
Size	5.7 cm H x 14.5 cm W x 27.1 cm L (2.2 in H x 5.7 in W x 10.7 in L)	
Weight	1.5 kg (3.3 lbs)	
Environment		
STORAGE	-40°C to +70°C, RH <95% non-condensing	
OPERATING	0°C to +40°C, RH <95% non-condensing	
	+40°C to +50°C, RH <75% non-condensing	
Protection Class 3	Relates solely to insulation or grounding defined in IEC 348.	

*Signals are specified as they appear at the ribbon cable plug pins

**Pod outputs set to high-impedance, except X2 (Pin 58) and CLKOUT (Pin 56)

Section 2
Installation, Self Test, and Getting Started

INTRODUCTION 2-1.

This section contains directions for setting up your Pod and Troubleshooter to work with your 80188 based UUT. It also contains general information about addresses, status lines, and control lines that you need to be familiar with to use the Pod.

Before using the Pod, you will need to connect it to the Troubleshooter and to the UUT. Before connecting it to the UUT, you should also perform the built-in self test to ensure that the Pod is operating correctly. Once you get the Pod connected and it has passed the Self Test, you may also need to adapt the Pod to your UUT's characteristics before using it for testing. The procedures for connecting the Pod, for performing the Pod Self Test, and also for getting the Pod set up and working with your UUT are included below in the first part of this section.

The Pod uses a simple composite address system to accept and display the addresses that are used for accesses to UUT memory, I/O spaces, and Pod Functions. The Pod also maps the 80188 Status and Control lines into consistent bit patterns that are used uniformly throughout Pod operations. Before using the Pod, you should read the information in the later parts of this section to become familiar with the way that the Pod uses addresses and Status and Control line information.

CONNECTING THE POD TO THE TROUBLESHOOTER 2-2.

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

1. Check that the Troubleshooter is OFF.

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

PERFORMING THE POD SELF TEST 2-3.

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

1. Make sure that the Pod is connected to the Troubleshooter properly.

2. If the Pod is connected to a UUT, release the Pod cable from the microprocessor socket.

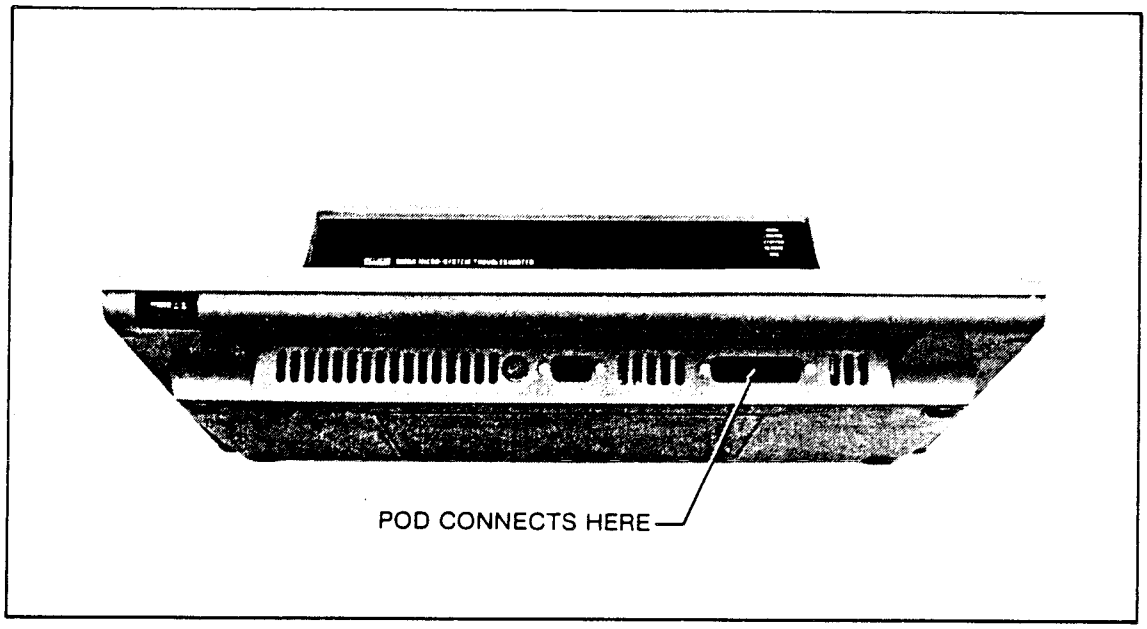


Figure 2-1. Connection of the Interface Pod to the Troubleshooter

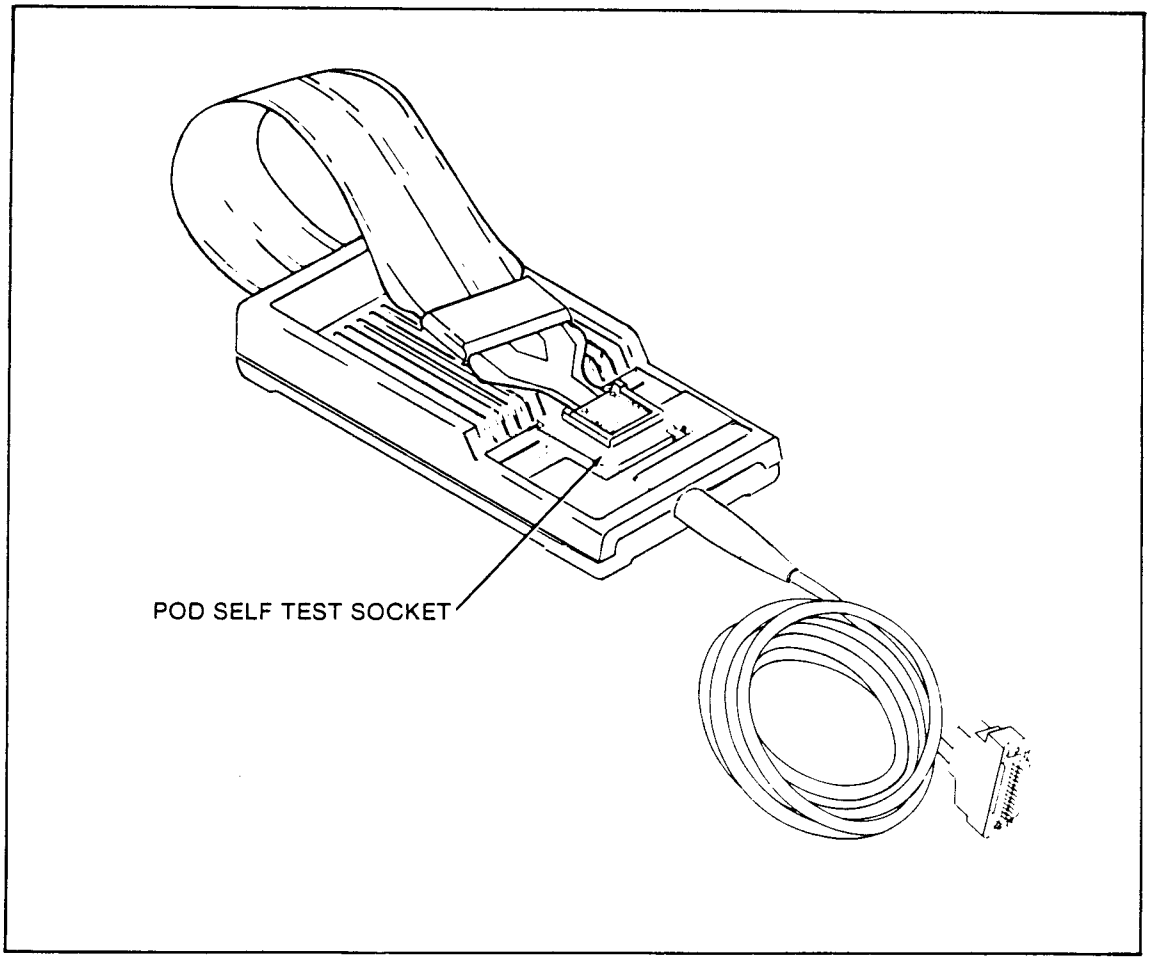


Figure 2-2. Pod Connected for Self Test

CAUTION

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

3. Insert the Pod cable into the Zero-Insertion Force (ZIF) self-test socket:

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

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

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

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

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

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

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

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

5. Press the BUS TEST key to initiate the Pod self test.

If the Troubleshooter displays the message *POD SELF TEST 80188 OK*, then the Pod is operating properly.

If the Troubleshooter displays the message *POD SELF TEST 80188 FAIL xx*, the Pod may not be operating properly. (The letters xx correspond to a failure code describing the error; the failure codes are listed in Section 6.) For information about diagnosing self test failures, refer to Section 6.

CONNECTING THE POD TO THE UUT 2-4.

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 WHICH 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 warning at the beginning of this section.

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

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

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

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

CAUTION

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

5. Insert the Pod plug into the UUT's microprocessor socket and secure it (using the same means used to secure the microprocessor). Make sure that pin 1 of the Pod plug is aligned with pin 1 of the microprocessor socket. If the UUT uses a Leadless Chip Carrier socket, then insert just the LCC Pod plug. If the UUT uses a pin-grid socket, then insert the Pod plug into a Pin-Grid Array adapter before attaching it to the UUT. See the self-test procedure above for instructions for attaching the Pin-Grid Array adapter.

CAUTION

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

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

CAUTION

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

7. Apply power to the UUT.

GETTING STARTED 2-5.

Introduction 2-6.

After you have connected the UUT, Pod, and Troubleshooter and, after power is applied to the Troubleshooter and UUT, several characteristics of the UUT may cause

conditions that prevent you from continuing with testing. If this happens, you must change the Pod configuration to suit your UUT's characteristics.

The following few paragraphs summarize the simple changes that you may need to make to the Troubleshooter setup and Pod configuration before you begin using the Pod. You may also need to configure the chip-select lines, DMA controller registers, and the RUN UUT entry address before using the RUN UUT mode or the Learn operation of the Troubleshooter. Changing the Pod configuration is described in detail in Section 4b.

NOTE

To use the Pod effectively, you will need to have schematics for the UUT, specifications for the number of Wait states required by memory on the UUT, and details of the required configuration for interrupts, DMA control, timers, and chip selects (such as might be provided by a listing of the UUT's program). You will use this information to configure the Pod to be like the UUT's microprocessor.

NOTE

Once you have determined how the Pod configuration needs to be set for your UUT, you can create a Troubleshooter program to make the changes automatically. Simply run this program at the beginning of each troubleshooting session.

NOTE

Other error conditions related to UUT defects are also possible. It is a good idea to begin using the Pod with a known-good UUT. That will eliminate any confusion between characteristics of the UUT that require a Pod configuration change, and UUT defects.

Changing Pod Characteristics 2-7.

Conditions on the UUT that are interfering with Pod operation will cause a message to be displayed on the Troubleshooter. The messages may occur either immediately after power is applied or after you try an operation (like BUS TEST). The instructions below offer simple solutions to the most common reasons why each condition may occur. Using these instructions, you should be able to identify the conditions which are causing the messages and change the Pod/Troubleshooter configuration to accommodate them. See the Troubleshooter Operator Manual for a complete description of each error.

If these instructions do not help in eliminating error conditions, repeat the self test and refer to the Troubleshooting instructions in Section 6 or contact Fluke Customer Service for advice.

UUT POWERFAIL — ATTEMPTING RESET

This message indicates that the voltage level at one or both of the UUT's microprocessor socket Vcc pins is below 3.5v dc. This message is also displayed if the Pod's UUT cable is not properly connected to either the UUT's microprocessor socket or the Pod's self-test socket, or if the Pod is not properly connected to the Troubleshooter.

ACTIVE FORCE LINE @ aaaa - LOOP?

This message tells you that forcing lines (microprocessor inputs that could force the microprocessor into a specific action when asserted) are active. Forcing lines may be active because of an error, or because of a design characteristic of the UUT. You can disable the reporting of this error and continue operation by selecting the Troubleshooter Setup function again and setting the Setup message *SET - TRAP ACTIVE FORCE LINE?* to NO.

BAD POWER SUPPLY @ aaaa - LOOP?

This message indicates that the voltage level at the UUT's microprocessor-socket Vcc pins is out of the 4.5V to 5.5V range (but not below 3.5V). If the UUT is intentionally using a low-level voltage, and you wish to ignore this error and continue, set the Troubleshooter's Setup command *SET - TRAP BAD PWR SUPPLY?* to NO.

POD TIMEOUT - ATTEMPTING RESET

This message is usually caused by status lines (input lines to the Pod microprocessor) called enableable status lines. Try reseting the UUT to begin with to eliminate the problem. If that does not work, disable the enable lines by setting the Troubleshooter Setup messages *SET - ENABLE EXTRDY?* and *SET - ENABLE HOLD?* to NO. This may be a temporary measure – just to get you going. You can go back at a later time and determine which of the enable lines is causing the problem. If one or the other is not causing the Pod to timeout, you can re-enable it to restore more thorough testing.

An alternate solution may be to change the *SET - TIMEOUT 200-CHANGE?* to a larger value, if the Pod Timeout is caused by the Troubleshooter not waiting long enough for the Pod to respond before reporting a Pod Timeout error.

NOTE

EXTRDY is a psuedo-status line that combines the function of the UUT microprocessor's SRDY and ARDY lines.

ACTIVE INTERRUPT @ aaaa - LOOP?

This message tells you that interrupts are active. Interrupts may be active because of an error, or because of a design characteristic of the UUT. You can disable the reporting of this error and continue operation by selecting the Troubleshooter Setup function again and setting the Setup message *SET - TRAP ACTIVE INTERRUPT?* to NO.

Entering in the Queue Status Mode by Accident 2-8.

The $\overline{RD}/\overline{QSMD}$ output line is held to a low logic level to force the Pod into the Queue Status mode. If you are testing a UUT that does not operate in the Queue Status mode, and the $\overline{RD}$ output line is accidently tied low, the Pod will go into the Queue Status mode. If this happens, the Pod will report address, data, or control drivability errors that do not actually exist on the UUT. To eliminate this problem, the error condition must be removed from the $\overline{RD}/\overline{QSMD}$ line on such a UUT.

Preparing for RUN UUT 2-9.

If you perform the RUN UUT function at the default entry address (F FFF0), the UUT's initialization programming should configure all of the Peripheral Control Block (PCB) registers in the Pod. If you do a RUN UUT at addresses other than the normal entry point though, you may have to manually configure the the entry address, the chip select definitions, interrupt configuration, and DMA controller specifications.

Whether you will need to configure these registers depends upon whether the function is used in your UUT, and whether the default value that is provided is suitable or not. Specifications are made by writing data to Pod Function addresses that are reserved for Pod use. See Pod Function Addresses below for a general description of using the Pod Function addresses. See Using the RUN UUT Function in Section 4A for a description of how to specify the required information.

Preparing for LEARN 2-10.

Before using the LEARN function, the Pod's functions may need to be configured to select the correct memory blocks on the UUT (if the defaults that are provided are not suitable). The chip select output lines are configured by writing data to Pod Function addresses that are reserved for Pod use. See Pod Function Addresses below for a general description of using the Pod Function addresses. See Section 4B for a description of how to specify the Chip Select definitions.

Changing the RESET Signal 2-11.

Some UUTs may need to have the RESET pulse applied at a high logic level (as an output of the microprocessor) when the Pod is reset by the Troubleshooter. After power-up, the default condition for the Pod is to NOT provide this signal, but you have the option of changing that. If your UUT needs the RESET signal to be supplied high to the UUT when the Pod is reset, see Configuring General Pod Characteristics in Section 4B for instructions on how to change the way that the reset signals of the Pod work.

Changing the Transparent Read Address 2-12.

The Pod does periodic Read operations to a specified address on the UUT to provide a stimulus for the UUT's refresh circuitry. If the default address used for this operation is not suitable for your UUT, you may need to change it. If your UUT needs to have a different address, see Configuring General Pod Characteristics in Section 4B for instructions on how to change it.

USING THE POD 2-13.

Once the Pod is connected to the Troubleshooter and the UUT, and the Pod has been configured to your UUT as described above, you can begin using the Troubleshooter/Pod team to investigate 80188 UUTs.

All of the standard diagnostic procedures that are built into the Troubleshooter work in the usual manner. In addition to these functions, there are many additional functions that are built into the Pod. These Pod functions allow you to do such things as perform high-speed memory tests, set up internal registers and masks, check for details about specific errors, alter default values of various parameters, and manipulate interrupt activity. Together, with the Troubleshooter's tests, they provide an extensive capability for diagnosing UUT defects.

The main thing to do when learning to use the 80188 Pod is to learn to control the various Pod functions. The Pod functions are implemented in the form of Pod Function addresses. Pod Function addresses are read/write locations that are not in the memory space of the UUT; they access diagnostic addresses within the Pod instead. The Pod Function Addresses are constructed with the usual 20-bit address information, plus additional bits to indicate specific Pod functions. More detailed information is located below under Pod Function Addresses.

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

ADDRESSES 2-14.

Introduction 2-15.

There are two types of addresses that are used with the 80188 Pod: UUT-access addresses and Pod Function addresses.

UUT-access addresses are used to Write data to or Read data from memory or I/O devices on the UUT. UUT-access addresses consist of two parts: the 20-bit physical address, which selects physical locations in the UUT, and 1 additional hex digit of information, which designates whether memory references are Normal or Direct Memory Access (DMA), or I/O or Memory accesses. Details about using accessing addresses on the UUT are described below under UUT Addresses.

Pod Function addresses allow you to use the many other functions that the Pod provides in addition to those that are built into the Troubleshooter. Pod Function addresses consist of two parts: a 20-bit physical address (an address that is meaningful to the Pod, not the UUT), and 2 additional hex digits of information which designate Pod functions. Information about using the Pod Functions is described below under Pod Function Addresses.

Figure 2-3 shows the general address structure that is used with the Pod. Specific formats for the address are described below in each topic.

NOTE

The most-significant four bits of the address (A28-A31) in the Troubleshooter are not used. If they are specified, they must be all set to 0 (hex 0XXX XXXX). It is not necessary to enter these unused bits (WRITE @ XXX XXXX is OK). They will not be displayed when address information is reported by the Troubleshooter.

NOTE

There are several address notation conventions that you should be aware of while using this manual:

- For ease of reading, the two halves of the address are separated with a space in this manual (HHHH LLLL). Do not try to enter addresses with a space (there's no "space" key on the Troubleshooter). The Troubleshooter does not display addresses with a space.

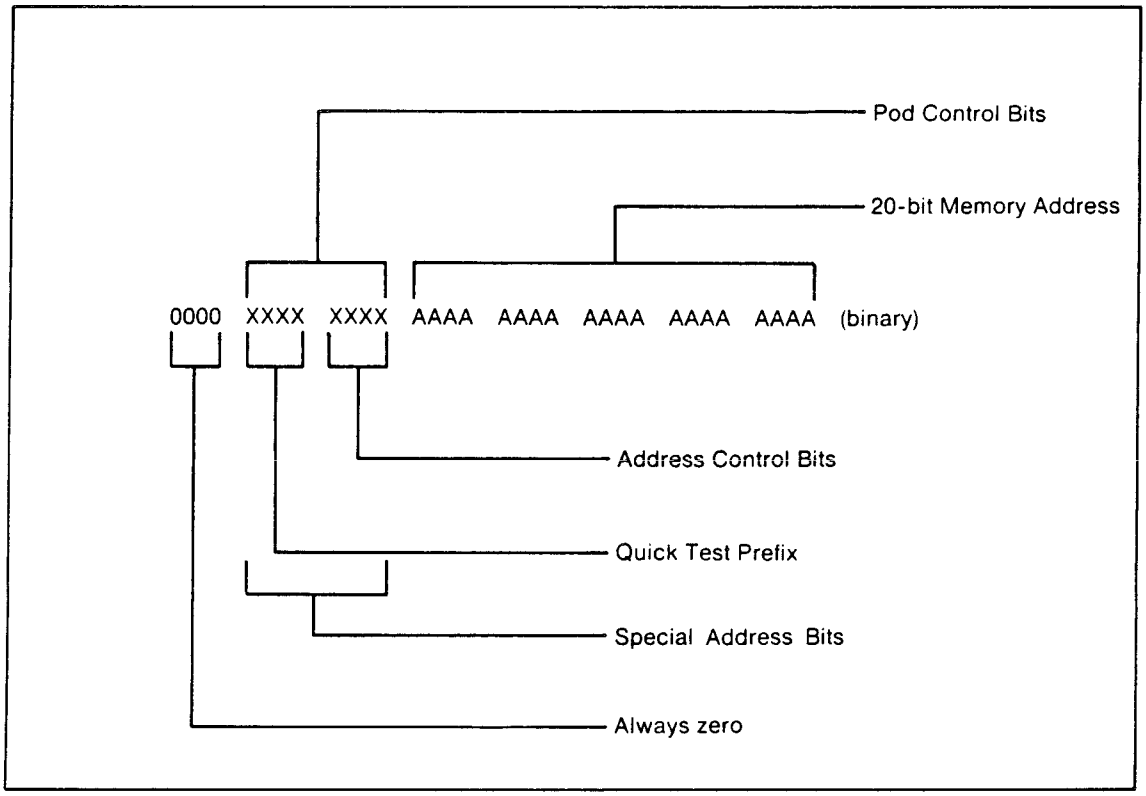


Figure 2-3. Addresses Used with the 80188 Pod

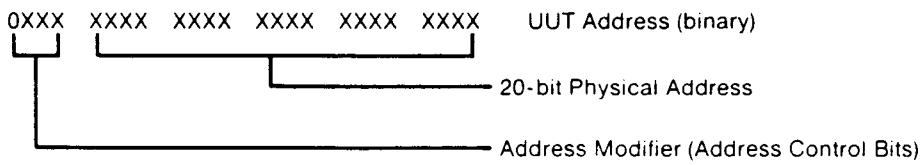
- X's are used to denote hex or binary digit spaces where the specific value may be any hex number. For example, the data value XX00 means that it's only important that the two least-significant digits be zero; the other two digits may be any value.

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

UUT Addresses 2-16.

INTRODUCTION 2-17.

UUT addresses consist of two parts: a 20-bit (5 hex digit) physical address and a single address modifier digit. The individual bits of the address modifier denote whether the address is Normal or DMA, or Memory or I/O access. Any time that a Troubleshooter Read or Write operation is used with a UUT address, it must consist of all six digits of information.



PHYSICAL ADDRESSES 2-18.

Introduction 2-19.

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

Memory Addresses 2-20.

The 80188's 20 physical address/data lines (AD0 -AD7, A8 - A15, and A16/S3 - A19/S6) allow it to address one megabyte (1,048,576 bytes) of memory. The 80188 only addresses bytes (8 bits of data). Each physical address is a byte address.

NOTE

In the 80188 Pod, addresses are always formed using the most-significant four bits of the SS Register of the 80188 microprocessor within the Pod. This method of forming the address is unimportant to the UUT to which the Pod is connected.

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

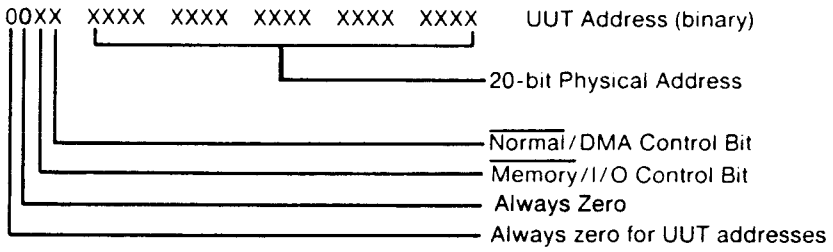
I/O Addresses 2-21.

The 80188 can address up to 64K (65,536) 8-bit ports. The I/O space is not segmented; to access a port address, the 80188 places the address on the lower 16 lines of the address bus. Address line A16 - A19 are output as zeros. I/O addresses are specified by an Address Control Bit (see below).

ADDRESS CONTROL BITS 2-22.

Introduction 2-23.

Four bits (one hex digit) of additional information are added to the normal 20-bit physical memory address to denote specific information about UUT accesses.



The additional bits are described below.

If this hex digit is not specified, the Pod will provide a default value. The default value is 0000 (binary); normal, memory accesses.

Normal/DMA Addressing 2-24.

Specifying the Normal/DMA bit (A20 in the "address") to be high causes the microprocessor's S6 control line to be set, indicating to the UUT that a DMA access is taking place. This allows "DMA style" transfers (not in response to a DMA request) that are actually normal UUT Read and Write operations with the S6 line forced high.

Even though the Pod does not do real DMA operations, this capability allows you to test DMA functions in a UUT. For example, *WRITE 1F FFB2* = XX specifies a deposit (a DMA Write operation) at location F FFB2.

When the $\overline{\text{Normal}}$/DMA bit is low (the default), normal Read and Write operations are performed.

$\overline{\text{Memory}}$/I/O ADDRESSING 2-25.

The $\overline{\text{Memory}}$/I/O bit (A21) determines whether the address is a memory reference or an operation with an I/O port. If A21 is low (the default), then the access is to memory. If A21 is high, then the access is to an I/O device.

NOTE

Only the lower 16 bits of the physical address are used for I/O operations. A16 - A19 are output low, and must be specified as zeros in the address.

For example, *WRITE @ 20 00C4* = XX specifies an I/O access at location C4.

Pod Function Addresses 2-26.

INTRODUCTION 2-27.

The Pod provides many additional functions in addition to those that are built into the Troubleshooter. These Pod functions are provided to allow you to set up internal registers and masks, to check for specific errors, to alter default values of various parameters, to manipulate interrupt activity, and to perform high-speed tests of memory. Table 2-1 shows the available Pod Function addresses.

The additional functions are used by reading and writing with Pod Function Addresses in the Pod. Pod Function Addresses are read/write locations that are not in the memory space of the UUT; they access diagnostic functions of the Pod. The Pod Function addresses themselves are not sent to the UUT—they are only a means to access the Pod functions.

SPECIAL FUNCTION ADDRESSES 2-28.

When the sixth digit of the address (bits A20-A23) is hexadecimal F, the address is a Special Function Address used by the Pod. These Special Function addresses provide a wide variety of diagnostic functions to use in a detailed investigation of a UUT.

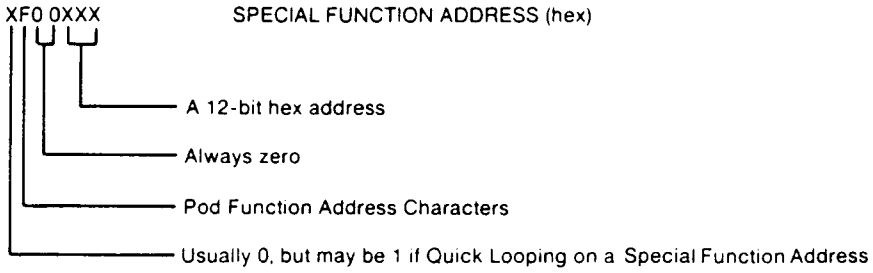


Table 2-1. Pod Function Addresses

ADDRESS	DESCRIPTION
Pod Control Addresses (Write/Readable)	
F0 0000	Selftest Address
F0 0002	Transparent Read Address (low byte)
F0 0003	Transparent Read Address (high byte)
F0 0004	Enable RESET output
Segment Register Contents for RUN UUT (Write/Readable)	
F0 0020	ES Register Contents (low byte)
F0 0021	ES Register Contents (high byte)
F0 0022	SS Register Contents (low byte)
F0 0023	SS Register Contents (high byte)
F0 0024	CS Register Contents (low byte)
F0 0025	CS Register Contents (high byte)
F0 0026	DS Register Contents (low byte)
F0 0027	DS Register Contents (high byte)
Error Reporting Addresses (Read Only)	
F0 0040	Last Error Summary
F0 0042	Last Control Errors (low byte)
F0 0043	Last Control Errors (high byte)
F0 0044	Last Forcing Line Errors
F0 0046	Last Active Interrupts
F0 0048	Last High (Nibble) Address Drivability Errors
F0 004A	Last Low (Word) Address Drivability Errors (low byte)
F0 004B	Last Low (Word) Address Drivability Errors (high byte)
F0 004C	Last Data Drivability Errors
F0 004E	Last INTA and TMR OUT Drivability Errors
F0 0050	Last CHIP SELECT Drivability Errors (low byte)
F0 0051	Last CHIP SELECT Drivability Errors (high byte)
F0 0052	Last Status (low byte)
F0 0053	Last Status (high byte)
Error Masks (Write/Readable)	
F0 0060	Error Summary Mask
F0 0062	Control Drivability Error Mask (low byte)
F0 0063	Control Drivability Error Mask (high byte)
F0 0064	Forcing Line Error Mask
F0 0066	Active Interrupt Error Mask
F0 0068	High (Nibble) Address Drivability Error Mask
F0 006A	Low (Word) Address Drivability Error Mask (low byte)
F0 006B	Low (Word) Address Drivability Error Mask (high byte)
F0 006C	Data Drivability Error Mask
F0 006E	INTA and TMR OUT Error Mask
F0 0070	CHIP SELECT Error Mask (low byte)
F0 0071	CHIP SELECT Error Mask (high byte)

Table 2-1. Pod Function Addresses (cont)

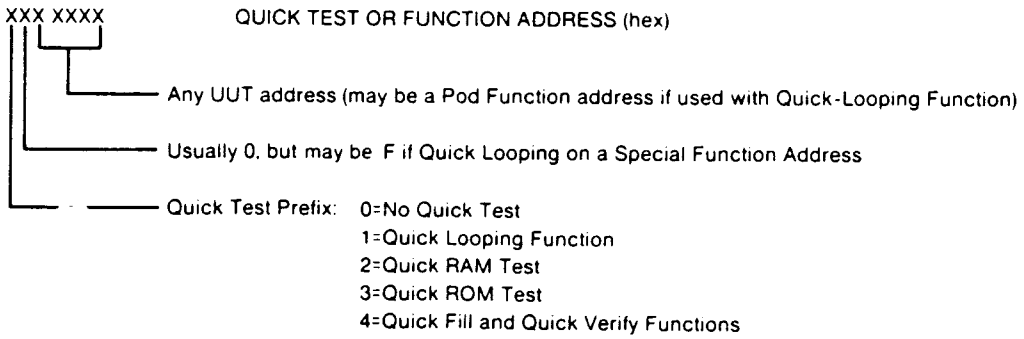
ADDRESS	DESCRIPTION
Interrupt Vector and Configuration Addresses (Write/Readable)	
F0 0080	Interrupt Configuration Address
F0 0082	Interrupt Vector 0 (Re-enable on Read)
F0 0084	Interrupt Vector 0 (No Re-enable on Read)
F0 0086	Interrupt 0 Cascade Address (low byte)
F0 0087	Interrupt 0 Cascade Address (high byte)
F0 0088	Interrupt Vector 1 (Re-enable on Read)
F0 008A	Interrupt Vector 1 (No Re-enable on Read)
F0 008C	Interrupt 1 Cascade Address (low byte)
F0 008D	Interrupt 1 Cascade Address (high byte)
Peripheral Control Block Addresses (Mostly Write/Readable)	
F0 0120 - F0 01FE	Peripheral Control Block Addresses

Pod Function Addresses are used by writing data to them to do such tasks as loading registers, defining parameters, or initiating Pod actions, or by reading them to retrieve information about Pod and UUT. Writing data to Pod Function Addresses F0 0020 and F0 0021, for example, defines the contents of the CPU's ES Register (the record of which is maintained in a location in the Pod's memory for diagnostic purposes for the RUN UUT function). The operation does not affect the contents of memory location 0 0020 in the UUT's memory.

All of the Pod Function Addresses are listed numerically in Table 2-1 to provide a complete reference, but since they are a means to use the unique functions that are provided by this Pod, they are described individually elsewhere in this manual. Functionally related groups of Pod Function Addresses also appear in other tables throughout the section.

QUICK TEST AND FUNCTION ADDRESSES 2-29.

The four bits in this portion of the address (A24-A27) indicate that one of the Pod's Quick Tests is to be used with the indicated physical address.



For example, *WRITE @ 3XX XXXX*=0 is an operation that is part of performing a Quick ROM Test.

If no Quick Test is wanted, the Quick Test Prefix (A24-A27) may be left blank and the default will be 0—No Test. (For example, *WRITE XX XXXX* = XX is OK.)

STATUS AND CONTROL LINES 2-30.

Introduction 2-31.

Pod status lines are input signals that are applied by the UUT to the UUT's microprocessor socket. Status lines are inputs that are not either data, clock, or power supply signals.

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

The Pod creates several additional psuedo-status and psuedo-control lines to augment the information that is normally reported to the Troubleshooter by the Pod.

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

The status and control lines provide much of the information that is used to diagnose defects in a UUT. You will use them frequently to determine the results of tests and to simulate microprocessor operations. It will pay to become familiar with how the status and control lines work with the Pod before using it to troubleshoot your UUTs.

NOTE

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

Figure 2-4 shows how the various lines interact between the Pod and the UUT.

Status Lines 2-32.

INTRODUCTION 2-33.

Most of the Status Lines are input lines to the UUT's microprocessor socket that indicate critical factors about system operation. Some additional status lines used are not real 80188 status lines, but PSEUDO-status bits that are generated within the Pod to indicate conditions that are important to the user. They are the same for both Normal mode and Queue Status mode operation. Status lines and PSEUDO-status lines are shown in Table 2-2.

STATUS LINE BIT ASSIGNMENTS 2-34.

Information about status lines is displayed as a map of bits, with each bit corresponding to a status line. The bits assigned to the specific status lines are shown in Table 2-2 and on the decal on the back of the Pod.

PSEUDO-STATUS LINES 2-35.

Several of the status lines reported during a Read Status operation are generated by the Pod and not by the UUT. These lines consist of PWR FAIL, INT VECT 0, INT VECT 1, ACTIVE INTR, and QSMD. These lines are described in detail in Section 3. Section 4B describes how to determine the source of the last interrupt (reported by ACTIVE INTR) using the Last Interrupt Pod Function address.

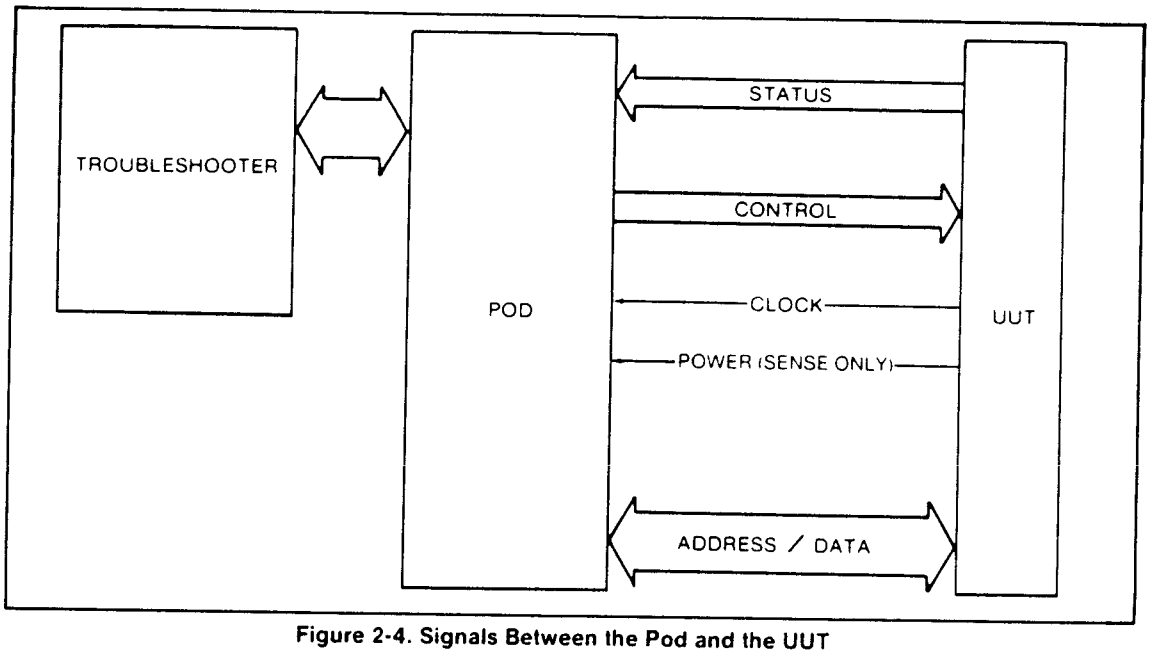


Figure 2-4. Signals Between the Pod and the UUT

Table 2-2. Status Lines

BIT	STATUS LINE	SPECIAL CHARACTERISTICS
0	ARDY	e, f
1	SRDY	e, f
2	HOLD	e, f
3	$\overline{\text{RES}}$	f
4	DRQ0	f
5	DRQ1	f
6	TMR IN 0	
7	TMR IN 1	
8	$\overline{\text{TEST}}$	
9	NMI	i
10	$\overline{\text{QSMD}}$	p
11	ACTIVE INTR	p
12	INTR VECTOR 0	p
13	INTR VECTOR 1	p
14	(not used)	
15	POWER FAIL	p

Special Characteristics Definitions

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

FORCING LINES 2-36.

Introduction 2-37.

Forcing lines are a special category of status lines that, when asserted, could force the UUT's microprocessor into some specific state or action. When the Pod is plugged into the UUT's microprocessor socket in place of the microprocessor, the Troubleshooter reports activity on these lines with the *ACTIVE FORCE LINE (@ aaaaa)-LOOP?* message.

The forcing lines of the 80188 Pod consist of ARDY, SRDY, HOLD, $\overline{RES}$, DRQ0, and DRQ1.

NOTE

An active forcing line is reported as a high bit in a status error word, regardless of the actual logic level of the signal. For example, if the $\overline{RES}$ line (bit 3) is active (low), an active Reset is reported as ACTIVE FORCE LINE, STS BTS 0000 0000 0000 1000, while a READ @ STS operation would show the true logic levels by reporting 0000 0101 0000 0000. An NMI interrupt, on the other hand (if the Troubleshooter's active interrupt trap is enabled) is reported as ACTIVE INTERRUPT, STS BTS 0000 0010 0000 0000, and a READ @ STS operation will report 0000 0111 0000 0000.

NOTE

It is possible to disable the reporting of active forcing lines by selecting the Troubleshooter Setup function message SET-TRAP ACTIVE FORCE LINE? NO. The Pod will still monitor the lines, but the Troubleshooter will not interrupt its operation to display the ACTIVE FORCE LINE error message on the display. Sometimes it is useful to disable the reporting of active forcing lines, particularly if the information is not needed by the operator.

User Enableable Forcing Lines 2-38.

The 80188 has several status lines which the operator can individually enable or disable using the Troubleshooter's Setup function: ARDY, SRDY, and HOLD. If these user-enableable lines are disabled (using the Troubleshooter Setup function), their inputs to the Pod microprocessor are disabled, but the Pod continues to monitor their condition. If they are asserted, the Pod reports to the Troubleshooter that a forcing line is active. Pressing the YES key on the Troubleshooter enables the status line; pressing the NO key disables the status line.

NOTE

Both ARDY and SRDY are enabled or disabled at the same time by the combined setup message ENABLE EXTRDY. If ARDY and SRDY are enabled, both must be low before an externally generated wait state can occur. If ARDY and SRDY are disabled, both must be captured low on the UUT (during UUT accesses) to generate an ACTIVE FORCING LINE error message. For information about how the User Forcing Line Mask affects the reporting of these lines as active forcing lines, see Pod Function Addresses.

When these UUT-generated lines are disabled, they are prevented from affecting Troubleshooter and Pod operation. For example, a HOLD line that is stuck high would cause the 80188 within the Pod to remain in a HOLD state, preventing normal Troubleshooter/Pod operation. When the Setup function of the Troubleshooter is used to disable this input to the Pod, the HOLD signal is prevented from reaching the 80188 within the Pod, allowing the Troubleshooter/Pod interactions to take place normally.

NOTE

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

NOTE

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

NOTE

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

Disabled Forcing Lines 2-39.

The $\overline{\text{RES}}$, DRQ0, and DRQ1 lines are forcing lines that are hardware-disabled except during RUN UUT. They are monitored and reported as active forcing lines if they are active during non-RUN UUT operation.

INTERRUPT LINES 2-40.

Some of the status lines are interrupt signals. Interrupt lines of the 80188 Pod consist of INT0, INT1, INT2, INT3, and NMI. The NMI input is disabled by hardware except during operation in the RUN UUT mode. Although disabled, the NMI input is routinely checked by the Pod software and reported to the Troubleshooter if held high by the UUT. NMI is also reported as a status bit (bit 9).

NOTE

Reporting of the active interrupt error message is disabled at power on. It is possible to enable the reporting of active interrupt lines by selecting the Troubleshooter Setup function message SET-TRAP ACTIVE INTERRUPT? YES. Sometimes it is useful to enable the reporting of active interrupts if you need the information.

Control Lines 2-41.

INTRODUCTION 2-42.

Control lines are Pod output lines that provide information to the UUT. Like status lines, some of the control lines used with the Pod are actually Pod-generated PSEUDO-control bits that provide information about Pod/UUT signals.

The control lines and some of their characteristics are listed in Table 2-3.

CONTROL LINE BIT ASSIGNMENTS 2-43.

Information about control lines is displayed as a map of bits, with each bit corresponding to a control line. The bits assigned to the specific control lines are shown in Table 2-3 and on the decal on the back of the Pod.

For example, when performing Bus Test or various other Troubleshooter functions, the Troubleshooter may detect that one or more control lines are not drivable. Suppose that during a BUS TEST the Troubleshooter detects that the $\overline{\text{LOCK}}$ line is not drivable. The Troubleshooter displays the message *CTL ERR 0000000 0000000 1-LOOP?* The zeros and ones correspond to the bit numbers assigned to the control lines as listed in Table 2-3. The first bit, Bit 0, is set to 1 because the $\overline{\text{LOCK}}$ line was detected as not drivable.

WRITING CONTROL LINES 2-44.

The 80188 Pod has several control lines that the Troubleshooter can set high or low with the WRITE CTL function. This feature is used by BUS TEST to check lines which cannot be toggled by normal read and write operations. It is also useful for helping troubleshoot these lines.

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

Table 2-3 Control Lines

BIT	CONTROL LINE	SPECIAL CHARACTERISTICS
0	$\overline{\text{LOCK}}$	w
1	RESET	w
2	HLDA	w
3	ALE (QS0)*	w (QS0 only)
4	$\overline{\text{WR}}$ (QS1)*	w (QS1 only)
5	$\overline{\text{RD}}$	
6	$\overline{\text{DEN}}$	
7	DT/$\overline{\text{R}}$	
8	$\overline{\text{S0}}$	
9	$\overline{\text{S1}}$	
10	$\overline{\text{S2}}$	
11	S6	
12	S7	
13	TMR OUT ERROR	p
14	INTA ERROR	p
15	CHIP SEL ERROR	p

w - writeable control line
p - pseudo-control line

*QS0 and QS1 replace ALE and $\overline{\text{WR}}$ when the Pod is in the Queue Status mode.

When the Pod operates in the Normal mode, the user-writable control lines consist of $\overline{LOCK}$, RESET, and HLDA. When the UUT operates in the Queue Status mode the QS0 and QS1 lines are also writable control lines. (In the Normal mode, the QS0 and QS1 lines are used instead as ALE and $\overline{WR}$, which are not writable.)

NOTE

The Pod will not report drivability errors on the $\overline{RD}$ signal when the Pod is in the Queue Status mode. (The $\overline{RD}$ signal is not produced by the Pod in the Queue Status mode.)

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

NOTE

Two of the writable control lines are active high (bit 3, HLDA, and bit 2, RESET), and one is active low (bit 0, $\overline{LOCK}$). To set the three lines to their inactive states during the WRITE@ CTL operation, use WRITE@ CTL = 001.

NOTE

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

PSEUDO CONTROL LINES 2-45.

Three of the Control lines are artificial signals that are generated by the Pod to provide clues to possible failures. The three PSEUDO-control lines, INTA ERROR, TMR OUT ERROR, and CHIP SEL ERROR are described in detail in Section 3. Refer to Section 4B for information about using the Last INTA, TIMER OUT, and Last Chip Select Driveability Error Pod Function addresses to diagnose errors reported by these pseudo-control lines.

Section 3
Information About Pod Signals

INTRODUCTION 3-1.

This section contains information about microprocessor signals, including descriptions of the 80188 pin assignments and of the signals as they are implemented by the Pod. It also includes descriptions of the Pseudo-status and Pseudo-control signals that are created by the Pod itself.

This information, particularly that which describes unique characteristics of the signals used by the Pod, will be useful during troubleshooting.

MICROPROCESSOR SIGNALS 3-2.

Table 3-1 lists all of the 80188 microprocessor signals and provides a brief description of how each signal is handled by the Pod. Some of the lines are classified by the Pod as Status Lines, Forcing Lines, Control Lines, or User-Enableable Lines. For information about these signal classifications, see Status and Control Lines in Section 2.

Figure 3-1 shows the 80188 Pod and microprocessor pin assignments.

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

POD-GENERATED SIGNALS 3-3.

Introduction 3-4.

Pseudo-status and pseudo-control lines are Pod-created read-only information bits that convey information about Pod or UUT operation that is not otherwise available from the microprocessor- or UUT-generated signals. (Status and Control lines are used by a microprocessor to exchange information with the rest of a system. See Status and Control Lines in Section 2.)

These psuedo lines are not actual lines that connect to the microprocessor, the UUT, or the Pod. The Pod creates the information solely for display on the Troubleshooter. The Troubleshooter displays them along with microprocessor/ UUT generated status and control lines when information is requested (for example, by READ @ STS). These lines provide you with added information about the results of Pod functions. For example, PWR FAIL is a Pod-generated status line that shows that power at the UUT's microprocessor socket is out of the range of 4.5 to 5.5 volts—information that is not obtainable using only the microprocessor's lines.

Table 3-1. 80188 Pod Signal Descriptions

SIGNAL NAME	PIN	DESCRIPTION
RESET	57	In an 80188-based system, RESET is an asserted-high output signal that indicates that the 80188 CPU is being reset. For troubleshooting purposes, the Pod treats the RESET signal as a Writeable Control line (see Table 2-3). Normally, the Pod does not assert the RESET signal at the UUT if the Pod and its microprocessor are being reset by the Troubleshooter. However, the Pod may be configured so that it DOES assert the RESET output signal under such a condition. (See Changing Pod Characteristics in Section 4.)
$\overline{RES}$	24	In an 80188-based system, the $\overline{RES}$ signal is an asserted-low Schmitt-trigger input to the microprocessor. When asserted, the $\overline{RES}$ signal suspends all operations of the microprocessor, and resets all internal registers. Instruction execution does not begin again until the $\overline{RES}$ signal is returned to a high logic level. The Pod treats the RES signal from the UUT as an asserted-low Forcing (Status) line (see Table 2-2). Except during RUNUUT, the RES input from the UUT is always disabled from reaching the Pod's microprocessor.
X1 X2	59 58	These signals are connections to an external crystal (both X1 and X2) or external clock oscillator (X1 only). The applied frequency is divided in half by the Pod's microprocessor and used for the internal microprocessor clock. **CAUTION** **The X1 and X2 clock lines do NOT have the input overload protection that all other signal connections between the Pod and the UUT have. Voltages on these signal lines greater than +5.5V dc or less than -0.5V dc may damage components within the Pod.**
CLKOUT	56	CLKOUT is a square-wave output signal that makes the internal clock of the 80188 microprocessor available to the remainder of the system. The frequency of the CLKOUT signal is one half the frequency of the signal(s) being applied to the X1 and X2 pins (see above). The CLKOUT signal of the Pod is connected from the Pod's microprocessor to the UUT without being buffered, and is therefore not put into a high-impedance state during a low power condition on the UUT. The CLKOUT signal does, however, pass through the Pod's hybrid protection networks. (See Section 5, Theory of Operation, for more information about the hybrid protection networks.) The drivability of the CLKOUT signal is not checked by the Pod.

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
$\overline{\text{TEST}}$	47	In an 80188-based micro-system, $\overline{\text{TEST}}$ is an input signal used to suspend instruction execution if it is sampled high upon execution of the WAIT instruction. Except during the RUN UUT mode, the UUT's $\overline{\text{TEST}}$ signal is always disabled from affecting the operation of the Pod's microprocessor. $\overline{\text{TEST}}$ is reported by the Pod as a Status line (see Table 2-2).
TMR IN 0 TMR IN 1	20 21	These two Timer Input signals can be used as either clock or control signals for the internal peripheral timers of the 80188 microprocessor. These two inputs are always enabled to the Pod's microprocessor—thus allowing operation of the timers in the Pod's microprocessor at all times. The Pod treats these two signals as Status lines (see Table 2-2).
TMR OUT 0 TMR OUT 1	22 23	These two signals are outputs of the 80188 microprocessor's internal peripheral timers. The Pod always enables these outputs to the UUT (except for a short time during BUS TEST) — thus allowing operation of the timers in the Pod's microprocessor at all times. The Pod treats these two outputs as control lines (see Table 2-3). These outputs are put into a high-impedance state by the Pod during a low power condition (UUT power supply voltage below 3.5V).
DRQ0 DRQ1	18 19	These two DMA-Request signals are activated by devices requesting one of the two DMA transfer channels of the 80186 microprocessor. Except during RUN UUT mode, DRQ0 and DRQ1 are disabled from reaching the Pod's microprocessor. The Pod treats DRQ 0 and DRQ 1 as Forcing (Status) lines (see Table 2-2).
NMI	46	In an 80188-based system, NMI is an asserted-high, non-maskable, edge-triggered interrupt to the microprocessor. Except during RUN UUT mode, NMI is always disabled from reaching the Pod's microprocessor. NMI is reported by the Pod as a Status line (see Table 2-2).
INT0 INT1	45 44	In an 80188-based system, INT0 and INT1 are input signals to the microprocessor for requesting maskable interrupts. Depending upon how the Pod is configured, INT0 and INT1 may or may not be allowed to interrupt the Pod's microprocessor during troubleshooting operations. See Configuring Interrupts in Section 4 for more information.
INT2/$\overline{\text{INTA0}}$ INT3/$\overline{\text{INTA1}}$	42 41	In an 80188-based system, INT2/$\overline{\text{INTA0}}$ and INT3/$\overline{\text{INTA1}}$ can be configured as either maskable interrupt inputs, interrupt-acknowledge outputs, or handshake lines for an external Programmable Interrupt Controller (PIC). The Pod can be configured to provide all possible functions of these lines during troubleshooting operations as well as during the RUN UUT mode.

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
A19/S6 A18/S5 A17/S4 A16/S3	65 66 67 68	A16-A19 are the four most-significant address output signals of the 80188 microprocessor (and of the Pod). These signals are time-multiplexed with the Bus Cycle Status output signals S3-S6. The 80188 microprocessor (and the Pod) generates A16-A19 during T1 of a bus cycle, and S3-S6 during T2, Tw, T3, and T4 of the same bus cycle. S6 is held low by the Pod to indicate that a bus cycle is a Normal access; S6 is held high to indicate that the bus cycle is a DMA transfer. (See Pod Addresses in Section 2 for more information.) S6 is reported by the Pod as a Control Line (see Table 2-3). S3-S5 are always held low by the Pod, and are not checked for drivability.
S7	64	S7 is always high.
AD0-AD7	17,15,13, 11,8,6,4,2	These are multiplexed address and data signals of the 80188 microprocessor (and the Pod). The address output signals (A0-A7) are generated during T1 of a bus cycle. Data is placed on the bi-directional data bus (D0-D7) during T2, Tw, T3, and T4 by an external device during a READ bus cycle, or by the microprocessor (and the Pod) during a WRITE bus cycle.
A8-A15	16,14,12, 10,7,5,3,1	These are additional address signals.
ALE/QS0	61	In the Normal mode of the 80188 microprocessor (and the Pod), the ALE (Address Latch Enable) output is provided to latch the address signals (A0-A19) into external devices. In the Queue Status mode, QS0 combines with QS1 to provide information about the 80188 microprocessor's internal instruction queue. (When the Pod is in the Queue Status mode, QS0 and QS1 provide information about the internal instruction queue of the Pod's microprocessor.) ALE/QS0 is reported by the Pod as a Control Line. When the Pod is in the Queue Status mode, QS0 is a user-writable control line. (See Table 2-3.)
$\overline{WR}$/QS1	63	In the Normal mode of the 80188 microprocessor (and the Pod), the $\overline{WR}$ (Write) output strobe is provided to enable data into external devices during WRITE bus cycles. In the Queue Status mode, QS1 combines with QS0 to provide information about the 80188 microprocessor's internal instruction queue. (When the Pod is in the Queue Status mode, QS0 and QS1 provide information about the internal instruction queue of the Pod's microprocessor.) $\overline{WR}$/QS1 is reported by the Pod as a Control Line. When the Pod is in the Queue Status mode, QS1 is a user-writable control line. (See Table 2-3.)

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
$\overline{RD}/\overline{QSMD}$	62	In the Normal mode of the 80188 microprocessor (and the Pod), the $\overline{RD}$ (Read) output strobe is provided to enable data from external devices into the microprocessor (and the Pod) during READ bus cycles. If the $\overline{RD}/\overline{QSMD}$ line is tied low on a UUT when the Pod is being reset (either by the Troubleshooter or by a power-up), the Pod will go into the Queue Status mode. In the Queue Status mode, the Pod provides the QS0 and QS1 output signals (see above). If the RD/QSMD line is not tied low, the Pod will remain in the Normal mode. In the Normal mode, the Pod provides the ALE, $\overline{RD}$, and $\overline{WR}$ output strobes. $\overline{RD}$ is reported by the Pod as a Control Line (see Table 2-3). $\overline{QSMD}$ is reported as a Status Line (see Table 2-2.)
ARDY SRDY	55 49	ARDY (Asynchronous Ready) and SRDY (Synchronous Ready) microprocessor (and the Pod). Assertion of either of these signals indicates that a data transfer from an addressed memory space or and I/O device can be completed. If both signals are held low at the proper times, Wait States will be inserted into the current bus cycle until either signal becomes asserted (high). The Pod treats these inputs as Enableable Status lines; They are normally enabled to reach the Pod's microprocessor. If enabled, and held or "stuck" low on the UUT, these signals will suspend Pod operations and a POD TIMEOUT will occur. ARDY and SRDY can be disabled from reaching the Pod's microprocessor using the Troubleshooter's SETUP function. (See Status Lines in Section 2 for more information.) When disabled, these signals are treated as Forcing lines by the Pod (see Table 2-2).
$\overline{LOCK}$	48	When asserted (low), the $\overline{LOCK}$ output signal of the 80188 microprocessor (and the Pod) denotes that other other system bus controllers may not have control over the multiplexed address/data bus. $\overline{LOCK}$ is reported by the Pod as a Control Line (see Table 2-3), and is user-writable.
$\overline{S0}$ $\overline{S1}$ $\overline{S2}$	52 53 54	$\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ are output signals of the 80188 microprocessor (and the Pod) that define the type of bus cycle being performed. The bus cycle type is defined according to the following table (1 = Logic High, 0 = Logic Low): **$\overline{S2}$ $\overline{S1}$ $\overline{S0}$ Type of Bus Cycle** 0 0 0 Interrupt Acknowledge 0 0 1 I/O Read 0 1 0 I/O Write 0 1 1 Halt 1 0 0 Instruction Fetch 1 0 1 Memory Read 1 1 0 Memory Write 1 1 1 No Bus Cycle S0, S1, and S2 are reported by the Pod as Control Lines (see Table 2-3).

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
HOLD	50	HOLD is an asserted-high input to the 80188 micro-processor (and the Pod). When asserted (high), HOLD denotes that another bus controller is requesting control of the local bus. The 80188 microprocessor responds to a HOLD input by asserting the HLDA output (see below) and by placing the microprocessor's bus into a high-impedance state. The HOLD input is treated by the Pod as an Enableable Status Line, and is normally enabled to reach the Pod's microprocessor. When the HOLD input is enabled and becomes asserted, the Pod responds by asserting its HLDA output (see below) and by placing the Pod's external bus into a high-impedance state (see Special Signal States in this section). If the HOLD input is enabled, and held or "stuck" high on the UUT, operation of the Pod can suspend, causing a POD TIMEOUT. The HOLD line can be disabled from reaching the Pod's microprocessor using the SETUP function of the Troubleshooter. When disabled, the HOLD input is reported as a Forcing Line by the Pod (see Table 2-2).
HLDA	51	The HLDA output of the 80188 microprocessor (and the Pod) is asserted (high) to acknowledge that an external device may drive the microprocessor's bus. The HLDA output is asserted in response to assertion of the HOLD input; HLDA then goes low shortly after HOLD is driven low. When HLDA becomes asserted, the microprocessor's bus is placed in a high-impedance state. (See Special Signal States in this section for a list of the Pod's signals that are placed in a high-impedance state in response to an asserted HLDA signal.) Note that if the HOLD input to the Pod is disabled (see above), the HLDA output is never asserted by the Pod in response to an asserted HOLD input. HLDA is reported by the Pod as a Control Line (see Table 2-3). HLDA is user-writable. NOTE The Pod's bus is NOT put into a high-impedance condition when HLDA is asserted by the Pod during a WRITE CTL UUT access.
CHIP SELECT OUTPUTS	(SEE BELOW)	The 80188 microprocessor (and the Pod) has 13 asserted-low Chip Select output signals. These outputs are used to select specific memory or I/O devices within distinct address spaces in an 80188-based system. The address space in which each Chip Select line becomes asserted (low) is software programmable (in the Pod as well as in the 80188 microprocessor). See Configuring Chip Selects in Section 4B for more information. The Pod collectively treats the Chip Select outputs as a "Pseudo Control Line" called CHIP SELECT ERROR (see Table 2-3). If the Pod dectects that one or more of the Chip Select outputs is not drivable, the Pod reports a Control Error on the CHIP SELECT ERROR control line. Which Chip Select line(s) caused the error can be determined by a READ at a special "Pod Function" address (see Determining Errors in Section 4B).

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
		The function of the individual Chip Select outputs is described in the paragraphs below.
$\overline{UCS}$	34	The $\overline{UCS}$ (Upper Memory Chip Select) output is asserted (low) whenever an access is made to the upper portion of memory. The address range of the $\overline{UCS}$ line is software programmable to be from 1K (1024) bytes wide (from FFC00 to FFFFF) to 256K (262,144) bytes wide (from C0000 to FFFFF).
$\overline{LCS}$	33	The $\overline{LCS}$ (Lower Memory Chip Select) output is asserted (low) whenever an access is made to the lower portion of memory. The address range of the $\overline{LCS}$ line is software programmable to be from 1K (1024) bytes wide (from 0 to 3FF) to 256K (262,144) bytes wide (from 0 to 3FFFF).
$\overline{MCS0}$ $\overline{MCS1}$ $\overline{MCS2}$ $\overline{MCS3}$	38 37 36 35	One of these Mid-Range Memory Chip Select outputs ($\overline{MCS0}$-$\overline{MCS3}$) becomes asserted when an access is made to the area of memory for which it has been programmed.
$\overline{PCS0}$ $\overline{PCS1}$ $\overline{PCS2}$ $\overline{PCS3}$ $\overline{PCS4}$ $\overline{PCS5}$/A1 $\overline{PCS6}$/A2	25 27 28 29 30 31 32	One of these Peripheral Chip Select output signals ($\overline{PCS0}$-$\overline{PCS6}$) becomes asserted when an access is made to the area of memory or I/O space for which it has been programmed. Each peripheral Chip Select line is valid over a 128-byte block of memory or I/O space; the starting address of the address block over which the Peripheral Chip Select lines become asserted is software programmable. $\overline{PCS5}$ and $\overline{PCS6}$ can also be software programmed to provide Address bits 1 (A1) and 2 (A2) during each bus cycle. Note that the Peripheral Chip Select lines are the only Chip Select lines that can be programmed to be valid in the I/O address space of the 80188 microprocessor (and the Pod).
DT/$\overline{R}$	40	DT/$\overline{R}$ (Data Transmit/Receive) is an output signal of the 80188 microprocessor (and the Pod) that indicates whether data flow through an external data bus transceiver is being transferred into the 80188 (DT/$\overline{R}$ low) or being transferred out onto the external data bus (DT/$\overline{R}$ high). DT/$\overline{R}$ is reported by the Pod as a Control Line (see Table 2-3).
$\overline{DEN}$	39	$\overline{DEN}$ (Data Enable) is an output signal of the 80188 microprocessor (and the Pod) that enables the outputs of external data bus transceivers. $\overline{DEN}$ is asserted during each memory or I/O access and is not asserted whenever DT/$\overline{R}$ changes state. $\overline{DEN}$ is reported by the Pod as a Control Line (see Table 2-3).

Table 3-1. 80188 Pod Signal Descriptions (cont)

SIGNAL NAME	PIN	DESCRIPTION
VCC1 VCC2	9 43	These pins are connections to the 80188 microprocessor's +5 volt power supply. The Pod does not receive its operating power from these pins (the Pod is powered by the Troubleshooter). The Pod monitors these pins for the correct power supply voltage (+5V dc). If the voltage on either of these pins is not within the range of +4.5V dc to +5.5V dc, the Pod will set the PWR FAIL status line (see Table 2-2) and report a BAD POWER SUPPLY to the Troubleshooter. If the voltage on both pins falls below +3.5V dc, the Pod will place all of its output signals (except CLKOUT and X2) into a high-impedance state, and the Pod will suspend its operations. This will cause a UUT POWER FAIL - ATTEMPTING RESET message to be displayed by the Troubleshooter when an operation requiring communication with the Pod is attempted.
GND GND	26 60	These pins are ground connections to the 80188 microprocessor.

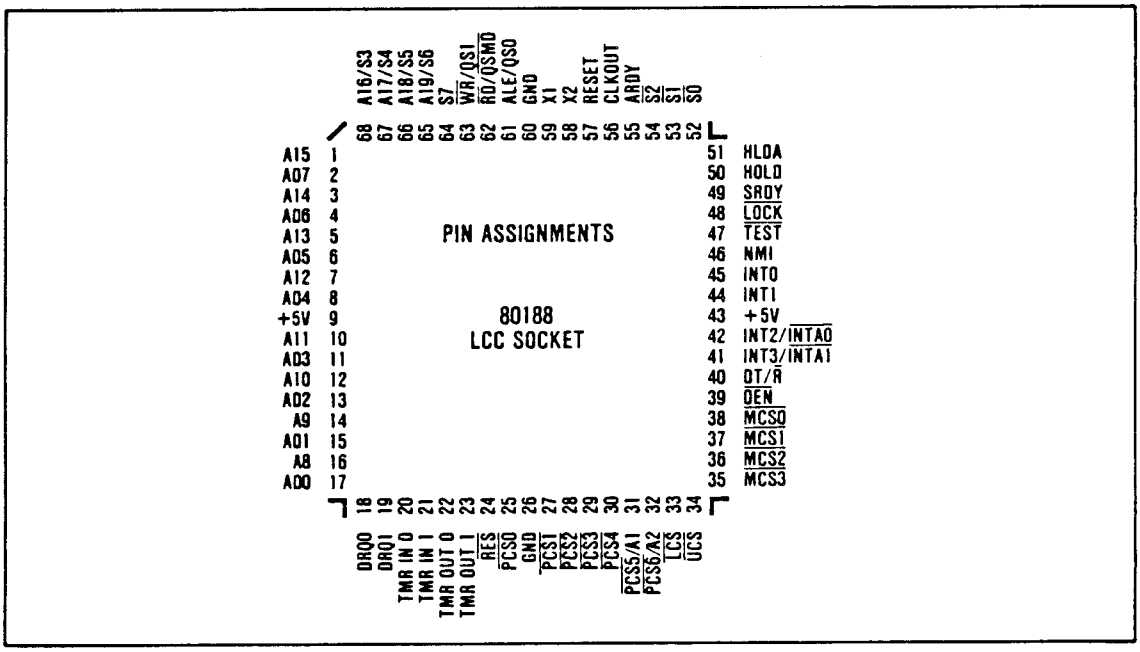


Figure 3-1. 80188 Pod and Microprocessor Pin Assignments

NOTE

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

The psuedo-status and psuedo-control lines are described below. Bit assignments for the psuedo-status and psuedo-control lines are shown in Tables 2-3 and 2-4.

Psuedo-Status Lines 3-5.

INTRODUCTION 3-6.

Several of the status lines reported during a Read Status operation are generated by the Pod and not by the UUT. These lines consist of PWR FAIL, INT VECT 0, INT VECT 1, ACTIVE INTR, and $\overline{\text{QSMD}}$.

POWER FAIL (PWR FAIL) 3-7.

The PWR FAIL status line is set high by the Pod whenever UUT power supply voltage on either of the 80188's Vcc pins drops below 4.5 volts or rises above 5.5 volts.

INTERRUPT VECTORS (INT VECT 0, INT VECT 1) 3-8.

These psuedo-status bits tell you that the Pod has performed an interrupt-acknowledge sequence in response to an interrupt from one of two possible interrupt channels. The INT VECT bits also indicate that the Pod has stored interrupt-type information (and, possibly, a cascade address) that was generated by an external Programmable Interrupt Controller.

The INT VECT 0 line is set whenever an interrupt acknowledge routine is performed by the INT0, INT2/$\overline{\text{INTA0}}$ pins, or whenever an interrupt-acknowledge routine is performed in iRMX* mode and the external Programmable Interrupt Controller does not select the internal interrupt controller of the Pod's microprocessor to provide the interrupt vector. (See the 80188 literature for explanations of these processes.)

The INT VECT 1 bit is set when an interrupt-acknowledge routine is performed by the INT1, INT3/$\overline{\text{INTA1}}$ pins (when the INT1 and INT3/$\overline{\text{INTA1}}$ lines are configured in that mode).

NOTE

INT VECT 0 and INT VECT 1 are never set unless the corresponding channel of the Pod's Interrupt Controller has been programmed to the proper mode. When either interrupt channel is programmed to have two direct inputs (as opposed to an INT/$\overline{\text{INTA}}$ pair) the corresponding INT VECT status bit is never set. Also, INT VECT 1 is never set when the interrupt controller is programmed in iRMX mode.

INT VECT 0 and INT VECT 1 are reset by using appropriate Pod Function addresses (see Interrupt Acknowledge Addresses).

ACTIVE INTERRUPTS (ACTIVE INTR) 3-9.

This is a pseudo-status bit that informs the user that one or more external interrupts is active. There can be from two to five external interrupt sources (including NMI), depending upon how the Pod's microprocessor is programmed. This bit can be masked from being reported as active. For more information, including instructions for determining the actual source of the interrupt, see Troubleshooting Interrupts in Section 4. See Section 4B for informatin about using the Last Active Interrupt Pod Function address to obtain more information about active interrupts.

$\overline{\text{QUEUE STATUS MODE}}$ (QSMD) 3-10.

The $\overline{\text{QSMD}}$ status bit tells the user which mode the Pod's microprocessor is in. If the $\overline{\text{QSMD}}$ bit is low, the Pod is in the Queue Status mode. If $\overline{\text{QSMD}}$ is high, the Pod is in the Normal Mode.

* iRMX is a registered trademark of the Intel Corporation.

The Pod uses the $\overline{QSMD}$ status bit to determine whether or not it should report drivability errors on the $\overline{RD}/\overline{QSMD}$ line. If the $\overline{QSMD}$ bit is low, the Pod is in Queue Status mode and $\overline{RD}/\overline{QSMD}$ will not be checked for drivability errors.

NOTE

If the Pod is in the Queue Status mode, the Pod will also output the QS0 and QS1 signals instead of ALE and $\overline{WR}$. In Queue Status mode, the QS0 and QS1 lines are writable control lines; the drivability of QS0 and QS1 is not checked except during a BUS TEST or a WRITE CTL operation.

If $\overline{QSMD}$ is high, the Pod is in Normal mode, the Pod produces the $\overline{RD}$, $\overline{WR}$, and ALE signals and will check the drivability of these lines during normal operations.

Pseudo-Control Lines 3-11.

INTRODUCTION 3-12.

Three of the Control lines are artificial signals that are generated by the Pod to provide clues to possible failures. The three psuedo-control lines are INTA ERROR, TMR OUT ERROR, and CHIP SEL ERROR. For information about locating and using the information provided by these lines, see Determining Errors in Section 4B. These three psuedo-control signals are used to report drivability errors to the Troubleshooter, which, if not disabled, will display a CTL ERROR message.

INTA ERROR 3-13.

INTA ERROR is a pseudo-control line that is set when one or more of the INTA lines is not driveable. (There can be 0, 1, or 2 $\overline{INTA}$ lines depending upon how the Pod is programmed by the user.) The Pod keeps track of which lines are and are not configured as $\overline{INTA}$ lines, and does not check the drivability of these interrupt lines if they are not configured as $\overline{INTA}$ lines. See Section 4B for information about using the Last $\overline{INTA}$ and TIMER OUT Driveability Error Pod Function address to isolate the source of the $\overline{INTA}$ error.

TMR OUT ERROR 3-14.

TMR OUT ERROR is a pseudo-control line that is set when one or more of the TMR OUT lines is not driveable. See Section 4B for information about using the Last INTA and TIMER OUT Driveability Error Pod Function address to isolate the source of the TMR OUT error.

CHIP SEL ERROR 3-15.

CHIP SEL ERROR is a pseudo-control line that gets set when one or more of the Chip Select lines is not driveable. See Section 4B for information about using the Last Chip Select Driveability Error Pod Function address to locate more information about reporting chip select errors.

SPECIAL SIGNAL STATES 3-16.

When the Pod detects certain conditions that indicate abnormal power levels, reset/initialization periods, or "Hold" states, it responds by driving selected signal lines to a high-impedance state. This protects UUT circuitry from damage by signals that might be present during those times. The specific conditions that initiate a reset are:

- The power level at the UUT's microprocessor socket drops below the 3.5 volt limit.
- A HOLD signal is asserted.
- The UUT sends an $\overline{RES}$ signal to the Pod (and the Pod's microprocessor) during RUN UUT.
- The Troubleshooter Resets the Pod (and the Pod's microprocessor).

The lines that are set to high-impedance states are:

AD0 - AD7
A8 - A15
A16/S3 - A19/S6
S7
$\overline{S0}$
$\overline{S1}$
$\overline{S2}$
DT/$\overline{R}$
$\overline{RD}$/$\overline{QSMD}$
$\overline{DEN}$
$\overline{WR}$

In addition, these other lines are set to a high-impedance state only when power drops below the 3.5 volt limit:

ALE/QS0
HLDA
$\overline{INTA0}$
$\overline{INTA1}$
TMR OUT 0
TMR OUT 1
CHIP SELECTS

POD DRIVE CAPABILITY 3-17.

As a driving source on the UUT bus, the Pod provides equal to or better than normal 80188 current drive capability. All Pod inputs and outputs (except the clock lines X1 and X2) are TTL-compatible.

Section 4
Operating Information

INTRODUCTION 4-1.

This Section contains descriptions of how the unique functions of the 9000A-80188 Interface Pod work. It also contains operating procedures that you will need to know to use it. The Section is divided into two Subsections:

4A USING POD FUNCTIONS	Section 4A describes the way that you use the various Pod functions to test a UUT. These tests all involve active read and write operations to check components and functional circuits (such as memory) on the UUT.
4B CONFIGURING THE POD	Section 4B describes the Pod Functions that may be used to configure the Pod so that its characteristics correspond to those required by the UUT. It also describes the Pod Function addresses that you can use to extract detailed UUT-error analysis information from the Pod after UUT operations are completed.

Together with the Troubleshooter Operator and Programming manuals, this information provides complete instructions for operating the Troubleshooter and Pod with 80188-based systems.

Section 4A
Using Pod Functions

INTRODUCTION 4A-1.

This section describes how to use Pod Functions to test components and circuitry on a UUT. It includes these subjects:

Testing RAM Quickly	How to use the Pod's Quick RAM tests to perform high-speed evaluation of blocks of RAM memory.
Testing ROM Quickly	How to use the Pod's Quick ROM test to do high-speed testing of blocks of ROM memory.
Using the Ramp Function	Using the Ramp function to create probe signatures for analysis of data signals on the UUT.
Using the Pod with an Oscilloscope	How to use the Pod's Quick Looping function to enhance an oscilloscope display, and how the different sync modes work.
Testing Interrupt Circuitry	Using the Pod to evaluate Interrupt circuitry on the UUT.
Testing DMA Circuitry	Using the Pod to check a UUT's DMA functions.
Using the RUN UUT Mode	How to exercise the UUT by using the Pod to emulate its microprocessor.

Before using the information in this Section, you should have already connected the Pod to the Troubleshooter and UUT, and configured it to your UUT's characteristics (as described in Sections 2 and 4B).

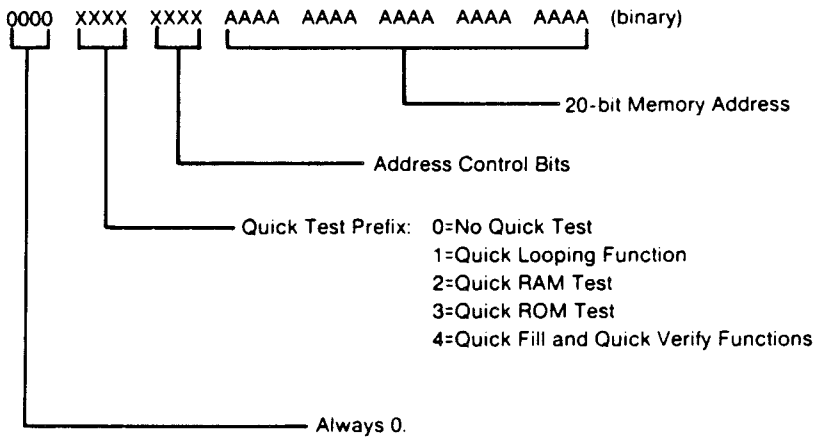
TESTING RAM QUICKLY 4A-2.

Introduction 4A-3.

The Pod provides several Quick Tests and Functions that allow you to conduct high-speed tests and verifications on RAM memory in the UUT. They allow you to do simple Read/Write and Pattern Verify tests on large portions of memory. The

diagnostics performed by the Pod during the execution of these Quick functions are less rigorous than the diagnostics performed during the execution of the Troubleshooter's built-in tests. They should be used when high speed testing is worth some sacrifice in thoroughness. Each function is described below.

Since the software routines that control these operations reside in the Pod and not in the Troubleshooter like the other tests and functions, you select these functions by writing to and reading from Pod Function addresses in the format shown below. The normal 20-bit address and Pod control bits are used to specify the physical address and access type. In addition to those 6 hexadecimal digits, another hex digit is placed in the seventh hex digit of the "address" to denote which of the Quick Functions or Tests are to be used. This simple process is described in detail in the following paragraphs.



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

Appendix A contains a program that, when loaded into a 9010A or 9005A Troubleshooter and executed, makes the Quick functions seem more like the Troubleshooter's built-in functions—you simply enter the address and data information according to displayed prompts. The program is provided in two forms: as a normal Troubleshooter source program and as a source program for the optional 9010 Language Compiler.

The Quick RAM Test 4A-4.

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

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

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

- The Pattern Verification test simply verifies that memory still contains the data that was last written in the Quick RAM test. Because dynamic RAM may often retain data for a long time (as much as a minute) without being refreshed, the Pattern Verification test is provided to check that the memory refresh is working and that the memory is retaining data. If problems with dynamic RAM are suspected, it is suggested that the pattern verification test be executed more than a minute after a Quick RAM test has been performed.

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

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

1. Define the starting address by a *WRITE@ 2XX XXXX*=0, where XX XXXX is the first address to be tested by the Quick RAM test.

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

   YY YYYY — Is the desired ending address. The ending address must be greater than the starting address.

   Z — Is the desired increment. If Z is omitted or specified as 0, the address increment defaults to 1.

   N — Is the test specification. N may be either

   1 = Normal Quick RAM test
   2 = Pattern Verification test

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

NOTE

The address-control bits (A20-A22) must be the same values for both the starting and ending address. Otherwise, the test will return an abort code A2 (illegal address in command).

To determine if the Quick RAM Test is still in progress, or what the test results are, you should perform a Read at the ending address.

- Press READ @ ENTER to command a READ operation at the last-entered address.

The Troubleshooter will display a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 4A-1.

Several read-only Pod Function addresses in the 200 00XX range contain additional information about the Quick RAM Test, including records of addresses used and any errors detected. The Pod Function addresses for the Quick RAM test are described in Table 4A-1.

NOTE

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

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

WRITE @ 201 5000=0
WRITE @ 201 5FFF=1 (or =11)

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

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

READ @ ENTER

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses in the 2000 00XX range. For example, you can find the low byte of the address where the error occurred by

READ @ 200 2008

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

READ @ 200 2012

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

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

WRITE @ 201 5FFE=2 (or =12)

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

Table 4A-1. Quick RAM Test

ACTION	MEANING
WRITE @ 2XX XXXX = 0	XX XXXX = starting address.
WRITE @ 2YY YYYY = ZN	YY YYYY = ending address Z = increment 0, 1 = byte addresses N = function type 1 = RAM test 2 = Pattern Verify 3 = RAM Test then Pattern Verify
READ @ ENTER	Returns status as follows: CODE | MEANING 00 | No test requested A0 | Aborted, new command entered A1 | Aborted, illegal data in cmd A2 | Aborted, illegal adr in cmd A3 | Aborted, Pod timeout occurred B0 | Busy, performing RD/WR check B1 | Busy, performing adr dcd check B2 | Busy, performing pattern verify C0 | Complete, no errors F0 | Failed, read/write error F1 | Failed, address decoding error F2 | Failed, pattern verify error
READ @ 200 0000	Low byte of the starting address
READ @ 200 0001	Second byte of the starting address
READ @ 200 0002	Third byte of the starting address
READ @ 200 0003	High byte of the starting address
READ @ 200 0004	Low byte of the ending address
READ @ 200 0005	Second byte of the ending address
READ @ 200 0006	Third byte of the ending address
READ @ 200 0007	High byte of the ending address
READ @ 200 0008	Low byte of the error address
READ @ 200 0009	Second byte of the error address
READ @ 200 000A	Third byte of the error address
READ @ 200 000B	High byte of the error address
READ @ 200 000C	Data expected at error address
READ @ 200 000E	Actual data returned from error address
READ @ 200 0010	Returns most recent code (same code as returned by READ @ ENTER)
READ @ 200 0012	Hex mask of error bits*
READ @ 200 0014	Returns increment and function type

*The hex mask of error bits is an Exclusive OR of the expected and actual data. Therefore, it is not a complete mask. For an address decoding error, the mask is meaningless.

READ @ ENTER

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

The Quick Fill and Quick Verify Functions 4A-5.

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

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

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

- The Quick Fill function will write the data that is contained in the starting address to all of the addresses in the block.

- The Quick Verify function will read data from all of the addresses in the block an compare each one to the data contained in the starting address. Errors will be reported via the Pod Function addresses described below.

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

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

1. Specify the data to be used by a *WRITE @ XX XXXX* = DD, where:

   XX XXXX Is the starting address (including control bits).

   DD Is the data to be filled throughout the defined address block.

   NOTE

   This step is only used with the Quick Fill function.

2. Define the starting address by a *WRITE@ 4XX XXXX*=0, where XX XXXX is the starting address.

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

YY YYYY	Is the desired ending address. The ending address must be greater than the starting address.
Z	Is the desired increment. If Z is omitted or specified as 0, the address increment defaults to 1.
N	Is the test specification. N may be either 1 = Quick Fill function 2 = Quick Verify function 3 = Quick Fill and Quick Verify functions

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

NOTE

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

NOTE

The address-control bits (A20-A22) must be the same values for both the starting and ending address. Otherwise, the test will return an abort code A2 (illegal address in command).

To determine if the Quick Fill or Quick Verify function is still in progress, or what the results are, you should perform a Read at the ending address.

- Press READ @ ENTER to command a READ operation at the last-entered address.

The Troubleshooter will display a code indicating the status of the test or the results. The status codes and their meanings are shown in Table 4A-2.

Several read-only Pod Function addresses in the 400 00XX range contain additional information about the Quick Fill and Quick Verify function, including records of addresses used and errors. The Pod Function addresses for the Quick Fill and Quick Verify functions are also described in Table 4A-2.

NOTE

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

Table 4A-2. Quick Fill and Verify Function

ACTION	MEANING
WRITE @ 4XX XXXX = 0	XX XXXX = starting address.
WRITE @ 4YY YYYY = ZN	YY YYYY = ending address Z = increment 0, 1 = byte addresses N = function type 1 = Fill 2 = Verify 3 = Fill then Verify
READ @ ENTER	Returns status as follows:
READ @ 400 0000	Low byte of the starting address
READ @ 400 0001	Second byte of the starting address
READ @ 400 0002	Third byte of the starting address
READ @ 400 0003	High byte of the starting address
READ @ 400 0004	Low byte of the ending address
READ @ 400 0005	Second byte of the ending address
READ @ 400 0006	Third byte of the ending address
READ @ 400 0007	High byte of the ending address
READ @ 400 0008	Low byte of the error address
READ @ 400 0009	Second byte of the error address
READ @ 400 000A	Third byte of the error address
READ @ 400 000B	High byte of the error address
READ @ 400 000C	Data written by fill
READ @ 400 000E	Actual data returned from error address
READ @ 400 0010	Returns most recent code
READ @ 400 0012	Hex mask of error bits
READ @ 400 0014	Returns address increment and function type

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

The following example demonstrates the simple procedure used to conduct a combined Quick Fill and Quick Verify function. To specify the functions over the RAM addresses 1 2000 through 1 2FFF with the default address increment of 1 and data 55, do the following two operations:

WRITE @ 1 2000=55
WRITE @ 401 2000=0
WRITE @ 401 2FFE=3

Line 1 inserts the data to be used into the first address. Line 2 defines the starting address to be 1 2000. Line 3 defines the ending address to be 1 2FFE and the test to be a

combined Quick Fill and Quick Verify function with an increment of 1 (by default). It also causes the function to start.

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

READ @ ENTER

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses.

TESTING ROM QUICKLY 4A-6.

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

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

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

1. Define the starting address by a *WRITE@ 3XX XXXX*=0, where XX XXXX is the first address to be used by the Quick ROM test.

2. Define the ending address and address increment by a single *WRITE@ 3YY YYYY*=Z1, where:

   YY YYYY — Is the desired ending address. The ending address must be greater than the starting address.

   Z — Is the desired increment. If Z is omitted or specified as 0, the address increment defaults to 1.

   1 — Denotes "Ending Address".

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

NOTE

The address-control bits (A20-A22) must be the same values for both the starting and ending address. Otherwise, the test will return an abort code A2 (illegal address in command).

To determine if the Quick ROM Test is still in progress, or what the test results are, you should perform a Read at the ending address. The Troubleshooter will display a code indicating the status of the test or the test results. The status codes and their meanings are shown in Table 4A-3.

- Press READ @ ENTER to command a READ operation at the last entered address.

Several read-only Pod Function addresses in the 300 00XX range contain additional information about the Quick ROM Test, including records of addresses used and any errors detected. The Pod Function addresses for the Quick ROM test are described in Table 4A-3.

NOTE

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

Table 4A-3. Quick ROM Test

ACTION	MEANING
WRITE @ 3XX XXXX = 0	XX XXXX = starting address.
WRITE @ 3YY YYYY = Z1	YY YYYY = ending address Z = increment 0, 1 = byte addresses
READ @ ENTER	Returns status as follows:
READ @ 300 0000	Low byte of starting address
READ @ 300 0001	Second byte of starting address
READ @ 300 0002	Third byte of starting address
READ @ 300 0003	High byte of starting address
READ @ 300 0004	Low byte of ending address
READ @ 300 0005	Second byte of ending address
READ @ 300 0006	Third byte of ending address
READ @ 300 0007	High byte of ending address
READ @ 300 000C	Checksum (low byte)
READ @ 300 000D	Checksum (high byte)
READ @ 300 000E	Hex mask of bits detected as inactive
READ @ 300 0010	Returns most recent code
READ @ 300 0014	Returns address increment

CODE	MEANING
00	No test requested
A0	Aborted, new command entered
A1	Aborted, illegal data in cmd
A2	Aborted, illegal adr in cmd
A3	Aborted, Pod timeout occurred
B0	Busy, test in progress
C0	Complete, no errors
C1	Complete, inactive bits detected

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

WRITE @ 30F C000=0
WRITE @ 30F DFFF=1(or 11)

Line 1 defines the starting address of the Quick ROM test to be F C000. Line 2 defines the ending address to be F DFFF, and the test to been done with an increment of 1. It also causes the test to start.

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

READ @ ENTER

If an error occurs, you can check for more detailed information about the failure by reading the associated Pod Function addresses.

To test the other ROM, you would use a similar command, but start the test at the higher address.

WRITE @ 30F E000=0
WRITE @ 30F FFFF=1 (or 11)

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

READ @ ENTER

NOTE

The only error that will be indicated is one for stuck bits. The test will always return a checksum value to addresses 300 000C and 300 000D, but there will be no indication whether this is the correct value. You must compare this calculated value with a known good value to determine if the Quick ROM test passed.

USING THE RAMP FUNCTION 4A-7.

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

1. Press PROBE SYNC D to select the data sync mode for the probe.
2. Probe the selected data-related signal on the UUT.
3. Press READ PROBE to clear the probe.

4. Press RAMP to do the Ramp function.

5. Press READ PROBE to read the probe.

6. Compare the displayed signature with the expected signature.

USING THE POD WITH AN OSCILLOSCOPE 4A-8.

Introduction 4A-9.

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

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

The Troubleshooter's Trigger Output can be used to synchronize your oscilloscope with specific cycles of Pod operations. The Probe and Synchronization Modes topic below describes the synchronization modes that are provided by the 80188 Pod, for the Troubleshooter's probe as well as an oscilloscope.

Using the Quick-Looping Function 4A-10.

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

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

For example, if the operator specifies the operation *READ @ 120 F000,* the Pod will perform a Quick-Looping Read operation at the address 20 F000 (an I/O access at 0 F000). If the operator specifies the operation *WRITE @ 100 B007* = 2F, the Pod will perform a Quick-Looping Write operation at address 0 B007, writing the data 2F.

The Quick-Looping function may be used with read or write operations at any valid address, and any of the special interrupt function addresses listed later under Interrupt Handling. The Quick-Looping function is not intended to be used with any of the other troubleshooting functions or tests.

If both error reporting and the Quick Looping feature are desired, you may apply the ordinary Troubleshooter Looping function to the Quick-Looping read or write, such as *READ @ 1XX XXXX LOOP.* The Troubleshooter will command read operations

at address XX XXXX at the normal looping speed with full error reporting. For every ordinary read operation, the Pod will interject a few Quick-Looping read operations (with no error reporting) which will enhance oscilloscope viewing.

Probe and Scope Synchronization Modes 4A-11.

INTRODUCTION 4A-12.

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

A = Address Sync
D = Data Sync
I = Interrupt Acknowledge
F = Free-Run

ADDRESS SYNC 4A-13.

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

described above. For probe response, the probe will latch on the signal level present at its tip at the time of the second or high-going scope trigger pulse.

DATA SYNC 4A-14.

If data sync is selected, the scope trigger will pulse low at the start of the data portion of the UUT access (the end of the address portion). It will pulse high at the end of the data cycle (the trailing edge of the Data Enable pulse).

INTERRUPT-ACKNOWLEDGE SYNC 4A-15.

If interrupt-acknowledge sync is selected, the scope trigger will pulse low at the start of the interrupt-acknowledge cycle, and it will pulse high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle.

NOTE

The interrupt-acknowledge sync mode is selected by the "1" key on the Troubleshooter (since there is no "I" key).

FREE-RUN 4A-16.

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

At power-on, the probe is in the free-run mode, and the scope trigger output pulses are synchronized to the address cycle.

USING THE SYNC MODES 4A-17.

One recommended method for using the scope synchronization is to synchronize on the negative edge of the scope trigger using the address sync mode. This will trigger the scope at a time slightly before the UUT access, allowing you to see the entire UUT access cycle. If you cannot verify that the scope is synchronized to the correct edge, use the pulse stimulus mode of the probe and display the signal on the scope. Since the probe stimulus pulses only occur during the UUT access (when the address sync mode is selected), you can easily check whether the scope is synchronized to the correct edge. After verifying the correct synchronization, use the scope to look at the $\overline{\text{DEN}}$ signal during a looping Read operation. If the probe's red indicator is on, then sync is timed with the address cycle. If the probe's green indicator is on, then the sync is timed with the data access. If both indicators are on, the sync is free-running. The relationship of these signals and the UUT access is described and illustrated in Section 5.

If the signal image on the scope is dim because of a low repetition rate, use the Quick-Looping function to increase the repetition rate and make the signal easier to see.

TESTING INTERRUPT CIRCUITRY 4A-18.

Introduction 4A-19.

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

The Pod can be configured to match the structure of the UUT's interrupts. If your UUT uses interrupts, you will probably need to set up the interrupt configuration for your UUT manually (unless the default values that are provided are adequate). Information below will help you change the interrupt configuration. (Use the interrupt-configuration routines that are normally provided by the UUT's software as a guide.)

The 80188 Pod provides two distinct methods of structuring interrupts: Normal mode, where interrupts are passed from external devices to the 80188's internal interrupt controller, and iRMX mode, where the internal interrupt controller is configured to be a slave to an external device. Using the interrupts in the two different modes is explained separately below.

Normal Mode Interrupts 4A-20.

INTRODUCTION 4A-21.

The Pod provides special functions to allow you to control how interrupts are handled by the UUT and the Pod. Using these functions consists of writing data to and reading data from several Pod Function addresses.

Pod Function addresses in the F0 008X range are used to configure the Pod's interrupt-handling characteristics. The F0 01XX (PCB) addresses do not need to be used (they are only used during RUN UUT operations).

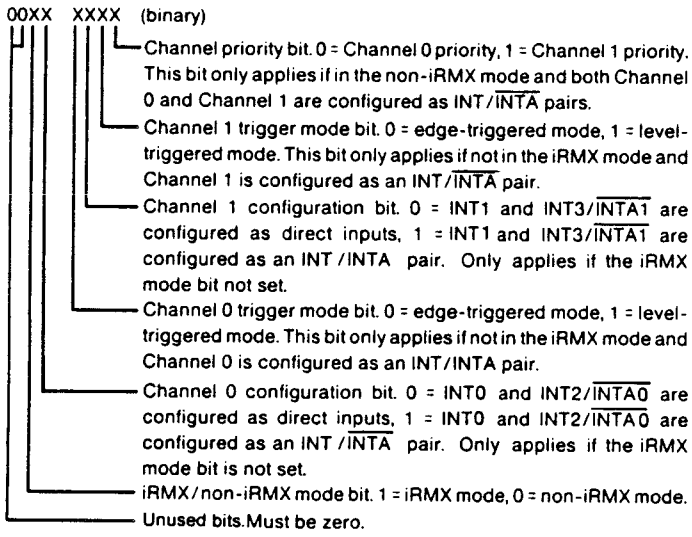
Instructions for controlling interrupts are provided in the paragraphs below. Included is information about configuring the different external interrupt lines and functions, reading the interrupt type and cascade address information that is provided by the system when an interrupt occurs, and forcing interrupt-acknowledge routines.

CONFIGURING INTERRUPTS

4A-22.

The Interrupt Configuration Pod Function address provides a simple method for configuring the Pod's interrupt handling. Writing a single data byte to this address completely specifies the interrupt handling configuration.

To configure the Pod to the interrupt-handling scheme that is required by your UUT, write a data byte to the Pod Function address F0 0080, coded as follows:



For example,

A *WRITE @ F0 0080*=1D (1 1101 binary) configures both channel 0 (INT0 and INT2/INTA0) and channel 1 (INT1 and INT3/INTA1) as interrupt/interrupt-acknowledge pairs (both bits 4 and 2 of the data are set). Channel 0 is programmed to the edge-triggered mode (bit 3 of the data is set), and channel 1 is programmed to the edge-triggered mode (bit 1 of the data is not set). Since bit 0 is set, channel 1 has priority.

A *WRITE @ F0 0080*=20 (10 0000 binary) configures the Pod to the iRMX mode.

A *WRITE @ F0 0080*=0 configures the Pod to its default interrupt mode, with all interrupt lines configured as inputs. No interrupt-acknowledge sequences occur in this mode.

A *READ @ F0 0080* returns the current interrupt configuration of the Pod.

The default value contained in Pod Function address F0 0080 after power-up is 00 (non-iRMX mode, INT0 and INT2/INTA0 configured as direct inputs).

NOTE

When either INT2 or INT3 are configured for direct input, the corresponding interrupt acknowledge lines are disabled by the Pod. The Pod Function addresses below, otherwise used for reading interrupt information and forcing interrupt-acknowledge routines, will not work. Attempts to read interrupt vector addresses will always return 00.

NOTE

A BUS TEST will check the drivability of any lines that are configured as Interrupt Acknowledge lines. Once you have the Pod configured to produce any interrupt-acknowledge signals, a BUS TEST is a good way of making sure that the $\overline{INTA}$ lines are drivable, instead of forcing an interrupt-acknowledge routine to check the drivability of an $\overline{INTA}$ line. (UUT-generated interrupt-acknowledge cycles will not cause the Pod to check the drivability of the $\overline{INTA}$ lines.)

ENABLING INTERRUPTS 4A-23.

A Write operation to the Pod Function address F0 0080 described above enables interrupts on any channel that is configured as an INT/$\overline{INTA}$ pair. The Write operation also clears both of the INTR VECTOR status bits and clears both sets of interrupt types and cascade addresses.

NOTE

To eliminate conflicts, there is no Troubleshooter Setup function to enable or disable interrupts as there is in Interface Pods for other microprocessors. All interrupt control is accomplished via the Pod Function addresses listed here.

Reading the interrupt types at F0 0082 and F0 0088 as described below will re-enable the respective interrupts and clear the corresponding INT VECT status bits.

READING INTERRUPT INFORMATION 4A-24.

Introduction 4A-25.

When the Pod and the UUT are configured to have interrupt/interrupt-acknowledge channels, upon receipt of external interrupts on such a channel, the UUT can generate two pieces of information: an interrupt type and a cascade address. The Pod has Pod Function addresses that allow you to view this information.

Reading Interrupt Types 4A-26.

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

The interrupt type can be determined by reading the Pod Function addresses listed below. Note that reading either of two addresses, F0 0082 or F0 0088, also clears the associated status bits and re-enables the interrupts. The other two addresses, F0 0084 and F0 008A, leave the status bits untouched.

NOTE

Until the interrupt configuration is set, these addresses are illegal.

READ @ F0 0082	Returns the type that is pointed to by INTR VECTOR 0. Reading this Pod Function address re-enables INT0 and clears the INTR VECTOR 0 status bit.

READ @ F0 0084	Also returns the type that is pointed to by INTR VECTOR 0, but does not clear the INTR VECTOR 0 bit or re-enable INT 0.
READ @ F0 0088	Returns the type that is pointed to by INTR VECTOR 1. Reading this Pod Function address re-enables INT1 and clears the INTR VECTOR 1 status bit.
READ @ F0 008A	Also returns the type that is pointed to by INTR VECTOR 1, but does not clear the INTR VECTOR 1 bit or re-enable INT 1.

These and other the Pod Function addresses that are used for interrupt handling are listed in Table 4A-4.

NOTE

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

Reading Cascade Addresses 4A-27.

When the UUT has multiple slave Programmable Interrupt Controllers (PIC's), the Master PIC provides a cascade address during the interrupt-acknowledge routine to indicate which slave originated an interrupt. If the UUT is so designed, the PIC can put the cascade address on the UUT's address bus, and the Pod can capture it during INTA cycles.

This 16-bit cascade address can be found by reading two of the four Pod Function addresses F0 0087 (INT0 - high byte), F0 0086 (INT0 - low byte), F0 008D (INT1 - high byte), or F0 008C (INT1 - low byte). Reading these addresses does not re-enable the interrupts or clear the respective INTR VECTOR X status bits.

FORCING INTERRUPT ACKNOWLEDGE ROUTINES 4A-28.

To help with testing interrupts, the Pod Function addresses allow you to force interrupt-acknowledge routines. You can force an interrupt-acknowledge routine by writing any data to the Pod Function addresses that contain the interrupt-type information.

Interrupt-acknowledge routines may be forced without an actual interrupt having occurred. All that's required is for the interrupt channel to be configured as an interrupt pair (channel 0, for example, needs to be set up as an INT0/$\overline{\text{INTA0}}$ pair). The forced interrupt clears previous types and cascade addresses and replaces them with new information.

The Pod Function addresses for forcing interrupts are:

WRITE @ F0 0082 = xx	Force an interrupt-acknowledge routine on Channel 0
WRITE @ F0 0084 = xx	Force an interrupt-acknowledge routine on Channel 0
WRITE @ F0 0088 = xx	Force an interrupt-acknowledge routine on Channel 1
WRITE @ F0 008A = xx	Force an interrupt-acknowledge routine on Channel 1

For example, to force an interrupt-acknowledge routine on channel 1, write any data to F0 0088 or F0 008A.

The interrupt-type and cascade address information captured by the 80188 Pod can be read as described above under Reading Interrupt Information.

iRMX Mode Interrupts 4A-29.

INTRODUCTION 4A-30.

Some UUTs are designed with circuitry that operates the microprocessor in the iRMX interrupt mode. The internal Interrupt Controller in the 80188 Pod may be configured to be compatible with the iRMX-mode UUTs. When the Pod is configured to the iRMX mode, the internal interrupt controller of the Pod's microprocessor is treated as a slave controller to an external Peripheral Interrupt Controller (PIC). The external interrupt lines of the Pod are used to perform the necessary handshaking functions with the external PIC. This requires the Pod to handle these external interrupt lines differently than in the normal mode.

Interrupts are configured to the iRMX mode by setting bit 5 of the Interrupt Configuration Pod Function address (WRITE @ F0 0080 = 20).

iRMX INTERRUPT SIGNALS 4A-31.

The iRMX mode uses the same lines and Pod Function addresses as the normal mode, only with somewhat different functions. Table 4A-4 describes how the various interrupt lines are used in the normal mode and in the iRMX mode.

NOTE

The INT0 line must be enabled for the lines to function in the iRMX mode. The INT0 line is enabled by selecting the iRMX mode (via a WRITE @ F0 0080=20) and re-enabled by reading the interrupt-acknowledge type address (F0 0082).

Table 4A-4. Interrupt Handling

READ/WRITABLE ADDRESS	DESCRIPTION
F0 0080	Interrupt Configuration Address
F0 0082	Interrupt Vector 0 (Re-enable on Read)
F0 0084	Interrupt Vector 0 (No Re-enable on Read)
F0 0086	Interrupt 0 Cascade Address (low byte)
F0 0087	Interrupt 0 Cascade Address (high byte)
F0 0088	Interrupt Vector 1 (Re-enable on Read)
F0 008A	Interrupt Vector 1 (No Re-enable on Read)
F0 008C	Interrupt 1 Cascade Address (low byte)
F0 008D	Interrupt 1 Cascade Address (high byte)

INITIATING INTERRUPTS IN THE iRMX MODE 4A-32.

A WRITE @ F0 0088 or F0 008A will cause the Pod's INT3/$\overline{\text{INTA1}}$ line to be set high as an interrupt-acknowledge signal to the external master PIC. The PIC may or may not respond with an interrupt on the INT0 line. If an interrupt is active on INT0, the

Pod will respond to the interrupt with an iRMX mode interrupt-acknowledge routine on the INT0/$\overline{\text{INTA0}}$ channel. This iRMX mode interrupt-acknowledge routine samples the INT1 line (used by the Pod in iRMX mode as a SLAVE SELECT input). This sampling occurs on the falling edge of T2 of the second clock cycle of the interrupt-acknowledge routine, and the results are used to determine whether or not the interrupt vector is to be provided by an external PIC or the Pod microprocessor's internal PIC.

NOTE

After the Pod performs an iRMX-mode interrupt-acknowledge sequence, the Pod exits the iRMX mode until the INT0 line is re-enabled by a READ @ F0 0082. During the time that the Pod has left the iRMX mode, the INT2/$\overline{\text{INTA0}}$ and INT3/$\overline{\text{INTA1}}$ lines will be pulled high by the Pod.

A WRITE @ F0 0088 or F0 008A=0 will cause the Pod's INT3/$\overline{\text{INTA1}}$ line to be reset to a low condition when the Pod actually goes into the iRMX mode. When the Pod is initially configured in the iRMX mode (by a WRITE @ F0 0080=20), the data at the address F0 0088 and F0 008A is 0, and the INT3/$\overline{\text{INTA1}}$ line is set low.

A READ @ F0 0088 or F0 008A will return the data at these addresses.

PROCESSING INTERRUPTS 4A-33.

If INT1 (SLAVE SELECT) is high during the interrupt-acknowledge routine, an external device (usually a slave PIC) caused the interrupt and placed the interrupt type information on the data bus. The Pod will perform its normal interrupt-acknowledge routine, and capture the externally generated interrupt type and cascade address. The INTR VECTOR 0 status bit will be set, and the interrupt type can be read at addresses F0 0082 and F0 0084, as in the "INT/$\overline{\text{INTA}}$ pair" mode. The cascade address can be read at F0 0086, also as in the "INT/$\overline{\text{INTA}}$ pair" mode.

If INT1 (SLAVE SELECT) is low during the interrupt-acknowledge routine, the master PIC selected the Pod's interrupt controller as the interrupting device. The master PIC expects the Pod's interrupt controller to place the interrupt type on its own internal data bus. No external device is selected to place interrupt-type information on the external data bus. The information captured on the data bus by the Pod during the interrupt-acknowledge routine is invalid. The Pod executes the interrupt-acknowledge routine normally, except that it clears all bits of the data at both of the interrupt type addresses (F0 0082 and F0 0084) for Channel 0 to zero. This informs the user that the interrupt-type information that was captured by the Pod is invalid.

NOTE

If INT0 is disabled, incoming interrupts on this line will be reported as with any active interrupt. A high or low level on INT1 (SLAVE SELECT), however, will not generate an active-interrupt message.

FORCING INTERRUPTS 4A-34.

An iRMX mode interrupt-acknowledge routine can be forced by the user via a Write operation to address F0 0082 or F0 0084 (with any data) regardless of whether INT0 and/or INT1 (SLAVE SELECT) are enabled. This interrupt-acknowledge

routine will erase any unread interrupt type and cascade address and replace it with the new interrupt type and cascade address.

CHANNEL 1 POD FUNCTION ADDRESSES NOT USED 4A-35.

The Pod Function address F0 008C, devoted to Channel 1 interrupts, is not used in the iRMX mode. A Read operation at address F0 008C will return 00.

Using the Interrupt-Acknowledge Sync 4A-36.

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

NOTE

The interrupt-acknowledge sync mode is selected by the "1" key on the Troubleshooter (since there is no "I" key).

If interrupt-acknowledge sync is selected, the scope trigger will pulse low at the start of the interrupt-acknowledge cycle, and it will pulse high at the end of the interrupt-acknowledge cycle. An interrupt-acknowledge sequence consists of two bus cycles. The interrupt-acknowledge sync pulse starts at the beginning of the first bus cycle of the sequence and ends at the end of the "send interrupt acknowledge" bus cycle. See Using the Pod with an Oscilloscope earlier in this Section for more information about the available sync modes.

TESTING UUT DMA CIRCUITRY 4A-37.

Introduction 4A-38.

In the RUN UUT mode, the Pod provides the complete DMA capability of the 80188 microprocessor. It will handle both internal DMA operations (internal to the CPU) and external DMA operations. Specific instructions for using both types of DMA Controllers are provided in the following paragraphs. See Appendix H for detailed information about the DMA registers.

The Pod also allows you to investigate external DMA circuitry on the UUT by simulating DMA accesses. See Simulating DMA Accesses below.

DMA Operations During RUN UUT 4A-39.

INTRODUCTION 4A-40.

Before using the RUN UUT mode with a UUT that uses the CPU's internal DMA Controller, you must make sure that the DMA Controller registers are loaded with the correct contents. You can do this by either using an entry address for RUN UUT that will access the UUT's initialization programming (which the UUT would normally use to load the registers) or by predefining contents by writing data to Pod Function addresses.

The Pod provides default values for the DMA Controller registers that essentially turn the DMA mechanism off. These default values are shown in Table 4A-5.

Table 4A-5. DMA Control Addresses

FUNCTION	CHANNEL 0	DEFAULT	CHANNEL 1	DEFAULT
Control Word				
(high byte)	F0 01CB	00	F0 01DB	00
(low byte)	F0 01CA	00	F0 01DA	00
Transfer Count				
(high byte)	F0 01C9	00	F0 01D9	00
(low byte)	F0 01C8	00	F0 01D8	00
Dest. Pointer				
(upper 4 bits)	F0 01C6	00	F0 01D6	00
(middle 8 bits)	F0 01C5	00	F0 01D5	00
(lower 8 bits)	F0 01C4	00	F0 01D4	00
Source Pointer				
(upper 4 bits)	F0 01C2	00	F0 01D2	00
(middle 8 bits)	F0 01C1	00	F0 01D1	00
(lower 8 bits)	F0 01C0	00	F0 01D0	00

CONFIGURING DMA CONTROL REGISTERS 4A-41.

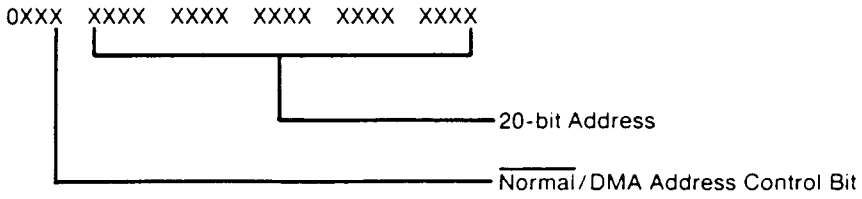
The DMA Control Registers are changed by writing to a group of Pod Function addresses. Table 4A-5 shows the Pod Function addresses that correspond to each DMA Controller Register. If you want to change the contents of the DMA Channel 0 Control Word to 0001, for example, enter *WRITE@ F0 01CB*=00 and *WRITE@ F0 01CA*=01. See Appendix H for more information.

EXTERNAL DMA OPERATIONS 4A-42.

External DMA controllers normally assert the HOLD line to make the 80188 CPU release control of the bus during DMA operations. If such external DMA operations are expected during normal operations of the Pod, the Pod should have the HOLD line enabled, so that the Pod will correctly respond and allow the external DMA controller to have control. Make sure that the Troubleshooter's Setup command SET -ENABLE HOLD? is set to YES. If the HOLD line is not enabled, the Pod will not change its lines to a high-impedance state in response to a HOLD signal and will not respond with a Hold Acknowledge (HLDA) signal. The absence of the HLDA signal will prevent DMA operation from occurring.

Simulating DMA Accesses for Troubleshooting 4A-43.

Even though the Pod does not provide DMA operation when it is not in the RUN UUT mode you may still be able to exercise DMA circuitry by writing and reading to and from DMA addresses. To simulate a DMA access, set the $\overline{\text{Normal}}$/DMA Address-Control Bit when you specify the address. (See Addresses in Section 2 for information about Address-Control Bits.)



For example, to do a fetch (DMA Read) from memory location 6 88FE, enter

READ @ 16 88FE

A deposit (DMA Write) to 0 002B would be done by entering

WRITE @ 10 002B=xx

When the $\overline{\text{Normal}}$/ DMA Address-Control Bit is high, the Control Line S6 is driven high by the Pod during memory accesses. The UUT may use the S6 signal to determine that the operation is a DMA reference, so that the corresponding circuitry can be enabled.

USING THE RUN UUT MODE 4A-44.

Introduction 4A-45.

The RUN UUT mode allows the Pod to emulate the UUT's microprocessor by executing a program directly from the UUT's memory. Before using the RUN UUT mode, you may first need to configure the RUN UUT entry address and registers in the Peripheral Control Block (PCB), such as chip select definitions and DMA Controller register specifications. The UUT's program listing may be a useful guide to configuring these items.

The RUN UUT Entry Address 4A-46.

INTRODUCTION 4A-47.

When you select RUN UUT, you may either use the default execution address that is supplied by the Pod, or you may explicitly specify the address where execution begins. Each of these cases are described in the following paragraphs. The Pod Function addresses that pertain to the RUN UUT mode are listed in Table 4A-6.

NOTE

Interrupts are always disabled when the Troubleshooter enters the RUN UUT mode.

USING THE DEFAULT EXECUTION ADDRESS 4A-48.

If the RUN UUT mode is selected and you do not specify the execution address, the Pod supplies the address 000F FFF0 as the execution address. Execution at this address sets the segments registers to their reset values: 0000 for the ES, SS, and DS registers, and FFFF for the CS register; the offset address is set to 0000.

NOTE

The default address for RUN UUT is the entry address for the microprocessor's Reset software. The UUT's programming may alter the values of the Chip Select Registers and other contents of the Peripheral Control Block registers. When the Pod exits from the RUN UUT mode, the Chip Select Registers and other Peripheral Control Block registers are returned to their previous configuration.

Table 4A-6. RUN UUT Control Addresses

FUNCTION	ADDRESS	DESCRIPTION
RUN UUT Type 1 (Using the Default Execution Address)	F FFF0	This is the RUN UUT default address that the Pod supplies if the operator does not specify an execution address. RUN UUT execution at F FFF0 sets all segment register to their reset values (0000 for ES, SS, and DS, and FFFF for CS) and sets the offset to 0.
RUN UUT Type 2 (Specifying the Execution Address)	YZZZZ	The RUN UUT execution address may be specified by entering the address as RUN UUT @ YZZZZ, the CS register will equal Y000, and the offset will be ZZZZ.
RUN UUT Type 3 (Specifying the Segment Register Contents)	F0 XXXX	The Segment Registers may be defined prior to a RUN UUT by writing the data to the following Pod Function addresses: F0 0020 = ES Register (low byte) F0 0021 = ES Register (high byte) F0 0022 = SS Register (low byte) F0 0023 = SS Register (high byte) F0 0024 = CS Register (low byte) F0 0025 = CS Register (high byte) F0 0026 = DS Register (low byte) F0 0027 = DS Register (low byte) After the desired values are written to the above addresses, execute RUN UUT at the address F0 XXXX, where XXXX is the desired offset address. The specified contents are loaded into the segment registers. As usual, the RUN UUT execution addresses are formed using the CS register: CS register contents are shifted left four bits, then added to the offset for the execution address.

SPECIFYING THE RUN UUT ADDRESS 4A-49.

You may select the address where RUN UUT execution begins by using any of three methods: changing the default RUN UUT entry address using the Troubleshooter's Setup command, explicitly defining the RUN UUT address, or by using Special Function addresses within the Pod.

- You may change the RUN UUT default address with the Troubleshooter Setup function by entering the desired address for the Setup message *SET-RUN UUT@ FFFF0-CHANGE?*

- You also may change the RUN UUT entry address by explicitly entering the beginning address when using the RUN UUT command. For example, entering RUN UUT @ 0000FE will cause the RUN UUT operation to begin at the address 00FE.

- Using the Special Function address F0 XXXX for RUN UUT will cause the RUN UUT to begin with the segment registers intact. Using this method, you can specify a RUN UUT to begin anywhere in memory, without the segment registers being reset when RUN UUT begins. For example, to perform a RUN UUT at A BCDE, first load the CS register with A000 by a WRITE @ F0 0024=00 and a WRITE @ F0 0025=A0, then specify a RUN UUT using the F0 XXXX address, RUN UUT @ FF BCDE.

Whenever any address other than the default address (F FFF0) is specified, the Pod considers the offset address to be equal to the four least significant digits, and the CS register contents to equal the fifth hexadecimal digit followed by three zeros. Consider the following examples:

After the following operations to set up the appropriate registers,

```
WRITE @ F00024=00
WRITE @ F00025=A0
RUN UUT @ F0BCDE
```

these are the results:

ADDRESS WHERE RUN UUT BEGINS	CS REGISTER CONTENTS	OFFSET
000A BCDE	A000	BCDE

For comparison, note that the default entry address for RUN UUT produces these values:

ADDRESS WHERE RUN UUT BEGINS	CS REGISTER CONTENTS	OFFSET
000F FFF0 (default address)	FFFF	0000

The CS register contents are set to the reset value and the offset is 0000.

Peripheral Control Block 4A-50.

INTRODUCTION 4A-51.

All of the Peripheral Control Block (PCB) registers may be programmed before you do a RUN UUT operation. This allows you to set up registers that would otherwise be configured by the UUT's software, and of course, allows you to manipulate the registers for diagnostic purposes.

Programming the Peripheral Control Block registers involves writing data to Pod Function addresses, which are summarized (with their default values) in Table 4B-2. Refer to Appendix H for detailed information about programming the PCB registers.

Before using the RUN UUT mode, you may define the contents of the Segment registers, Timer registers, DMA Controller registers, and Interrupt registers.

SPECIFYING SEGMENT REGISTER CONTENTS 4A-52.

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

1. Write the desired 16-bit value to the following Pod Function addresses before selecting RUN UUT (these Pod Function addresses and their default values are also listed in Table 4A-6):

   F0 0020 = ES register initial contents (low byte)(default=00)
   F0 0021 = ES register initial contents (high byte)(default=00)
   F0 0022 = SS register initial contents (low byte)(default=00)
   F0 0023 = SS register initial contents (high byte)(default=00)
   F0 0024 = CS register initial contents (low byte)(default=FF)
   F0 0025 = CS register initial contents (high byte)(default=FF)
   F0 0026 = DS register initial contents (low byte)(default=00)
   F0 0027 = DS register initial contents (high byte)(default=00)

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

2. After the desired values are written to the above addresses, specify RUN UUT at the address F0 XXXX (XXXX equals the offset address). When RUN UUT begins, the initial contents of the segment registers will equal the values at the Pod Function addresses.

NOTE

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

SETTING UP TIMERS 4A-53.

The design of your UUT may require that you define the operation of the Timers before using the RUN UUT mode. The default state of the timers is set to disable all timer functions. See Configuring the Timers in Section 4B for complete information.

PREDEFINING DMA CONTROLLER REGISTERS 4A-54.

You may need to define the operation of the internal DMA controller before using the RUN UUT mode. See Testing DMA Circuitry earlier in this Section for complete information.

DEFINING INTERRUPT HANDLING 4A-55.

You may need to define the operation of interrupts before using the RUN UUT mode. See Testing Interrupt Circuitry in this Section for complete information.

Section 4B
Configuring the Pod

INTRODUCTION 4B-1.

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

The Pod provides Pod Function addresses to allow you to set up the Pod to the same configuration as the microprocessor that it replaces. These include such procedures as configuring the chip select lines, the interrupt structure, and internal timer operation. Many of these items need to be set (usually, using the UUT's program listing as a guide) before using the Pod in the RUN UUT mode and, occasionally, before you can access components on the UUT with Read and Write operations.

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

Topics in this Section include:

Configuring Chip Selects	How to set up the Pod's Chip Select registers for use with your UUT.
Configuring Timers	How to set up the Pod's internal Timer registers for use with your UUT.
Configuring General Pod Characteristics	How to adapt the Pod to accommodate your UUT's specific characteristics.
Configuring Interrupts	Configuring the interrupt structure of the Pod.
Masking Errors	How to suppress the reporting of unimportant or known errors by the Pod.
Determining Errors	How to poll the Pod for specific information about previous errors.

CONFIGURING CHIP SELECTS 4B-2.

Introduction 4B-3.

The 80188 microprocessor has several programmable Chip Select lines. All of these need to be programmed to define which memory or peripheral addresses that they will enable. If your UUT uses these Chip Select lines, you will need to have them

programmed properly in the Pod so that the correct memory or peripheral blocks are enabled. That will ensure that all of the available memory and all of the peripherals are actually tested, and that the accesses have the correct timing. Otherwise, erroneous Read/Write errors may be reported due to the Pod trying to select non-existent devices or using the wrong delay time (number of Wait states), and some sections may be inadvertently skipped and not tested.

Normally, programming the Chip Selects would be done by the UUT's software as part of an initialization routine. When you test a UUT, you can handle these Chip Select lines by manually defining the Chip Select specifications (by writing to several of the Pod's Pod Function addresses) or by using the default Chip Select definitions that are provided by the Pod. These procedures are both described below.

NOTE

The 80188 microprocessor addresses the PCB (Peripheral Control Block) in word accesses. The Pod, on the other hand, reads and writes to these locations using byte accesses. The control word for each function is therefore composed of a high and low byte—the low byte being at an even address, the high byte at the next odd address. For example: the UMCS register is at offset A0, with a default value of C03B, this would be read by the Troubleshooter and Pod as READ @ F0 01A0 = 3B and READ @ *F0 01A1* = C0.

NOTE

Some 80188 UUTs do not use the Chip Select lines. Instead, they use a decoder device to expand the range of memory available beyond that which is possible using the Chip Selects. If your UUT does not use the Chip Select lines, you may still need to check the default values for the proper number of Wait states at a given address.

The chip select specifications take immediate effect and remain in operation during normal Pod operation, as well as during the RUN UUT mode.

NOTE

If the Chip Selects are altered by the UUT's software during the RUN UUT mode, such as they might be if the initialization software is executed, they will return to the previously selected values when the normal RUN UUT is finished. See Appendix I for a description of a special RUN UUT procedure that retains the UUT's register values upon completion.

Programming Chip Selects 4B-4.

INTRODUCTION 4B-5.

Manually programming Chip Selects is a two-fold process: defining the allocation of memory or peripheral blocks for each Chip Select line and specifying the number of Wait states to be used for accesses to each memory block. These values are defined by writing the data to Pod Function addresses in the Pod.

ALLOCATION 4B-6.

You may define the valid address range of any of the 80188's Chip Select lines by writing data to Pod Function addresses in the Pod which correspond to the Chip Select Registers in the microprocessor. Refer to Appendix H for information about the data format and application. Table 4B-I lists the Pod Function addresses used to control Chip Selects.

4B-1. Chip Select Special Addresses

SPECIAL ADDRESS	REGISTER	DEFAULT	DEFAULT WAIT STATES
F0 01A9	MPCS Register	A0	3
F0 01A8	MPCS Register	FB	
F0 01A7	MMCS Register	81	3
F0 01A6	MMCS Register	FB	
F0 01A5	PACS Register	40	3
F0 01A4	PACS Register	3B	
F0 01A3	LMCS Register	3F	3
F0 01A2	LMCS Register	FB	
F0 01A1	UMCS Register	C0	3
F0 01A0	UMCS Register	3B	

WAIT STATES 4B-7.

Part of each Chip Select definition is the number of Wait states to be used in each bus cycle that uses that chip select line. The number of Wait states is determined by the response speed of the memory or peripheral in the particular block that's being used. The Wait states are inserted into the bus cycle. The Wait states extend the length of the bus cycle so that the Pod's microprocessor can communicate with slowly responding devices.

If you use too many unnecessary Wait states or not enough Wait states, you may introduce timing problems that produce Read/Write errors. You need to use the literature from both the memory/peripheral manufacturer and the microprocessor manufacturer to determine the optimum number of Wait states for your application. Slower devices will require more Wait states (up to 3). Faster ones will not require as many, and the fastest will not need any at all. Not all of the memory and peripherals in a UUT will always have the same required number of Wait states—some may need different specifications.

USING CHIP SELECT DEFAULTS 4B-8.

The 80188 microprocessor normally only defines the address range of the Upper Memory Chip Select. That part of memory normally contains the initialization programming that will be used after power-up to configure the remainder of the lines. After power-up, the Pod provides default values for all of the available Chip Select lines, not just the Upper Memory Chip Select. These default values are shown in Table 4B-2. The default values have been selected to cover as much memory space as possible, in the largest blocks.

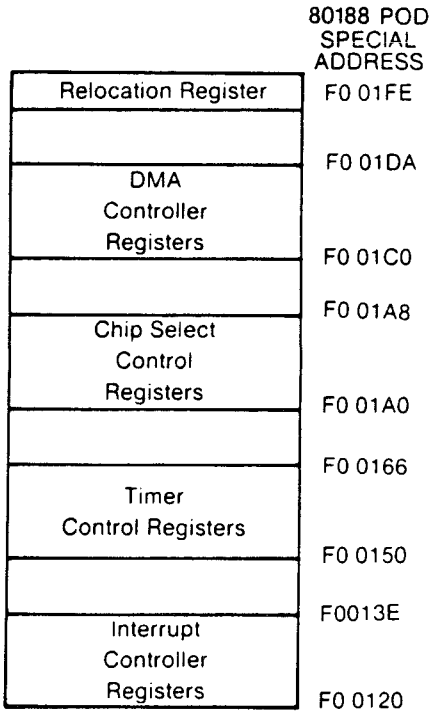
Each one of the default values allows three wait states during bus cycles to accommodate the slowest memories. The external Reading is also enabled by default.

You may be able to use the default values, even if they do not coincide with arrangement that is used by your UUT. By checking the table of defaults, you may be able to use the default arrangement to access the memory in your UUT. For example, you may be testing a UUT that uses its $\overline{\text{MCS}}$ Chip Select line to enable a certain block of memory. This Chip Select line is normally defined by the UUT's software to be the address range 20000-2FFFE. The Pod has this $\overline{\text{MCS}}$ line predefined by default to be the range 80000-8FFFE. You can access the memory on the UUT without redefining the $\overline{\text{MCS}}$ Chip Select line by reading and writing to addresses in the 80000-8FFFE range instead. (Refer to Table C-1 in Appendix C for the default Chip Select register values.)

CONFIGURING THE INDIVIDUAL CHIP SELECT LINES 4B-9.

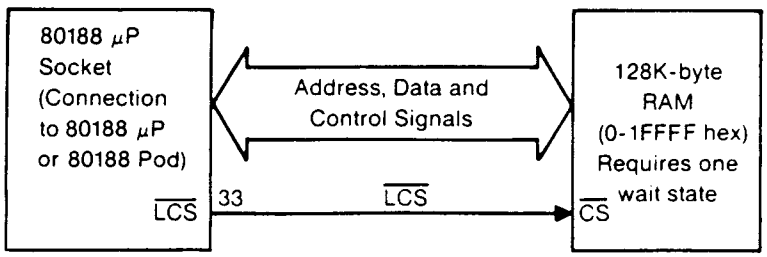
The Pod has its own Peripheral Control Block (PCB), like that of the 80188 microprocessor. The PCB of the Pod controls the behavior of the Internal Peripherals (including the Chip Select Generator and the Wait State Generator) of the Pod's microprocessor. The PCB registers of the Pod are accessed by Read and Write operations at Special Function Addresses (addresses that are not UUT addresses).

The locations of the Pod's PCB registers in the Pod's Special Function Address map are shown below. Note that the registers are arranged in groups according to which peripheral device they control, and that the beginning and ending address of each group of registers are shown. The Chip Select Control registers form one of these groups; they are located at the addresses F0 01A0 through F0 01A8.

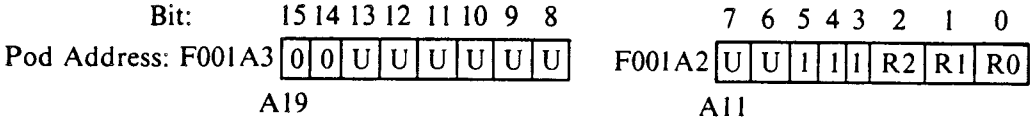


There are five Chip Select Control registers; one register controls the $\overline{UCS}$ line, one controls the $\overline{LCS}$ line, and three others control all of the Peripheral Chip Select and Mid-range Chip Select lines.

Detailed information about programming the Chip Select registers is located in Appendix H. The following paragraphs demonstrate how to program a Chip Select register for the block of RAM shown below.

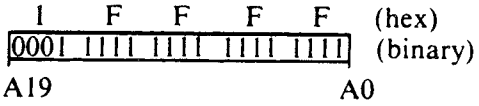


This RAM requires the $\overline{LCS}$ line to be configured to go low over the first 128K bytes (0-1FFFF) of the Pod's memory map. There must also be one Wait state inserted into all (UUT access) bus cycles that address the $\overline{LCS}$ line. The LMCS register of the Pod's PCB is used to control the $\overline{LCS}$ line. (The LMCS register is one of the Chip Select Control registers.) The bits in the LMCS register are shown below:



Bits 6 through 15 of the LMCS register represent address bits 11 through 19 (A11-A19) of the upper limit address of the $\overline{LCS}$ line. Bit positions 6 through 13 are a user-programmable portion of this address. Bits 3, 4, and 5 of the LMCS register are hardwired to a value of 1, and have no meaning.

For this example, the upper limit address of the $\overline{LCS}$ line is to be 1FFFF hex. The binary representation of this value is shown below:



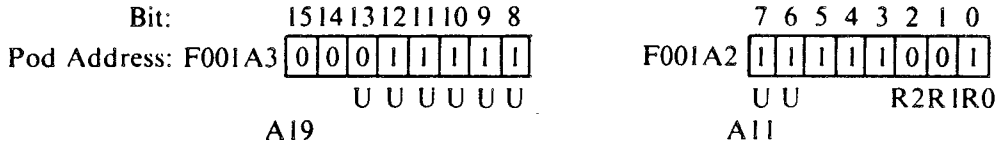
Address bits A17-A19 are set to zero for the hex address 1 FFFF. Thus, the corresponding bits in the LMCS register must be set to zero. The LMCS register bits 14 and 15 (corresponding to address bits A18 and A19, respectively) are already hard-wired to zero. (This restricts the upper limit address of the $\overline{LCS}$ line to 3 FFFF.) Of the bits labeled "U", only bit 13 (representing address bit 17) is to be set to zero.

The bits denoted by R0, R1, and R2 are programmable bits that control the number of Wait states associated with the $\overline{LCS}$ line. Bit R2 controls whether or not the external signals $\overline{ARDY}$ and SRDY are allowed to affect the number of Wait states in bus cycles using the $\overline{LCS}$ line; if R2 is set to zero, ARDY and SRDY are allowed, otherwise they are not. For the purpose of this example, R2 will be set to zero (ARDY and SRDY allowed). Bits R1 and R0 control the number of Wait states that are associated with the $\overline{LCS}$ line, as follows:

BIT VALUES:		NUMBER OF WAIT STATES
R1	R0	
0	0	0
0	1	1
1	0	2
1	1	3

In this example, 1 wait state is required, so R1 will be set to 0 and R0 will be set to 1.

The LMCS register bit assignments look like those shown below:



Hex value: 1FF9

$\overline{\text{LCS}}$ Configuration:
- Upper Address Limit = 1FFFF (hex)
- Lower Address Limit = 0
- Number of Wait States = 1
- ARDY and SRDY allowed

The hex value of the bits in the LMCS register should be 1FF9. To configure the $\overline{\text{LCS}}$ line, use the WRITE function of the Troubleshooter to put the hex data (1FF9) in the LMCS register (at address F001A2 and F001A3) of the Pod, as follows:

WRITE @ F001A2 = F9
WRITE @ F001A3 = 1F

CONFIGURING TIMERS 4B-10.

Introduction 4B-11.

Some UUT's may require that the Pod's timers be programmed a certain way in order for the Pod to be able to communicate with some of the devices on the UUT. The three timers in the Pod's microprocessor can be programmed the same as the UUT's microprocessor normally would be. Once programmed, the timers begin running immediately and remain in operation during normal Pod operation, as well as during the RUN UUT mode. This makes it possible for you to activate all timer-dependent devices.

At power-up, the Pod's timers are set to be non-functional (TMR OUT lines set high, TMR IN lines ignored). In order to use the timers, you will need to write data to several Pod Function addresses that correspond to the UUT microprocessor's Peripheral Control Block registers. The Pod Function Addresses that control the Pod's timers are listed in Table 4B-2.

See Appendix H for information about how to prepare the timer registers to perform a specific function.

NOTE

You will normally set up the timers in the same manner that the UUT's initialization software sets up the timers in the UUT's microprocessor. Use the program listing in Appendix H as a guide to defining the appropriate registers.

NOTE

If the Timer Control registers are changed by the UUT's software during RUN UUT, the register contents will be restored to their previous values when the RUN UUT mode is terminated.

NOTE

The Pod temporarily over-rides the programmed function of the TMR IN and TMR OUT signals during a BUS TEST in order to verify the drivability of the TMR OUT lines. After the BUS TEST, the timer functions are restored to function as they were originally programmed.

Table 4B-2. Peripheral Control Block Special Addresses

ADDRESS	BYTE (H=High L=Low)	DESCRIPTION	POWER-UP DEFAULT VALUE (hex)
Relocation Register Address			
F0 01FF	H	Relocation Register	00
F0 01FE	L		FF
DMA Controller Register Addresses			
F0 01DB	H	DMA Channel 1 Control Word	00
F0 01DA	L		00
F0 01D9	H	DMA Channel 1 Transfer Count	00
F0 01D8	L		00
F0 01D6		DMA Channel 1 Dest. Ptr. (upr 4 bits)	00
F0 01D5		(mdl 8 bits)	00
F0 01D4		(lwr 8 bits)	00
F0 01D2		DMA Channel 1 Src. Ptr. (upr 4 bits)	00
F0 01D1		(mdl 8 bits)	00
F0 01D0		(lwr 8 bits)	00
F0 01CB	H	DMA Channel 0 Control Word	00
F0 01CA	L		00
F0 01C9	H	DMA Channel 0 Transfer Count	00
F0 01C8	L		00
F0 01C6		DMA Channel 0 Dest. Ptr. (upr 4 bits)	00
F0 01C5		(mdl 8 bits)	00
F0 01C4		(lwr 8 bits)	00
F0 01C2		DMA Channel 0 Src. Ptr. (upr 4 bits)	00
F0 01C1		(mdl 8 bits)	00
F0 01C0		(lwr 8 bits)	00
Chip Select Register Addresses			
F0 01A9	H	MPCS Register	A0
F0 01A8	L		FB
F0 01A7	H	MMCS Register	81
F0 01A6	L		FB
F0 01A5	H	PACS Register	40
F0 01A4	L		3B
F0 01A3	H	LMCS Register	3F
F0 01A2	L		FB
F0 01A1	H	UMCS Register	C0
F0 01A0	L		3B
Timer Register Addresses			
F0 0167	H	Timer 2 Mode/Control Word Register	00
F0 0166	L		00
F0 0163	H	Timer 2 Max Count A Register	00
F0 0162	L		00
F0 0161	H	Timer 2 Count Register	00
F0 0160	L		00
F0 015F	H	Timer 1 Mode/Control Word Register	00
F0 015E	L		00
F0 015D	H	Timer 1 Max Count B Register	00
F0 015C	L		00

Table 4B-2. Peripheral Control Block Special Addresses (cont)

ADDRESS	BYTE (H=High L=Low)	DESCRIPTION	POWER-UP DEFAULT VALUE (hex)
F0 015B	H	Timer 1 Max Count A Register	00
F0 015A	L		00
F0 0159	H	Timer 1 Count Register	00
F0 0158	L		00
F0 0157	H	Timer 0 Mode/Control Word Register	00
F0 0156	L		00
F0 0155	H	Timer 0 Max Count B Register	00
F0 0154	L		00
F0 0153	H	Timer 0 Max Count A Register	00
F0 0152	L		00
F0 0151	H	Timer 0 Count Register	00
F0 0150	L		00
Interrupt Controller Register Addresses			
F0 013F	H	INT3 Control Register (master mode)	00
F0 013E	L	Not used in iRMX mode	0F
F0 013D	H	INT2 Control Register (master mode)	00
F0 013C	L	Not used in iRMX mode	0F
F0 013B	H	INT1 Control Register (master mode)	00
F0 013A	L	Timer 2 Control Register (iRMX mode)	0F
F0 0139	H	INT0 Control Register (master mode)	00
F0 0138	L	Timer 1 Control Register (iRMX mode)	0F
F0 0137	H	DMA 1 Control Register (both modes)	00
F0 0136	L		0F
F0 0135	H	DMA 0 Control Register (both modes)	00
F0 0134	L		0F
F0 0133	H	Timer Control Register (master mode)	00
F0 0132	L	Timer 0 Control Register (iRMX mode)	0F
F0 0131	H	Int. Cont. Status Reg. (both modes)	80
F0 0130	L		00
F0 012F	H	Int. Request Register (both modes)	00
F0 012E	L		00
F0 012D	H	In-Service Register (both modes)	00
F0 012C	L		00
F0 012B	H	Priority Mask Register (both modes)	00
F0 012A	L		07
F0 0129	H	Mask Register (both modes)	00
F0 0128	L		FD
F0 0123	H	EOI Register (master mode)	00
F0 0122	L	Specific EOI Register (iRMX mode)	00
F0 0121	H	Int. Vector Register (iRMX mode)	00
F0 0120	L	Not used in master mode	00

CONFIGURING GENERAL POD CHARACTERISTICS 4B-12.

Introduction 4B-13.

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

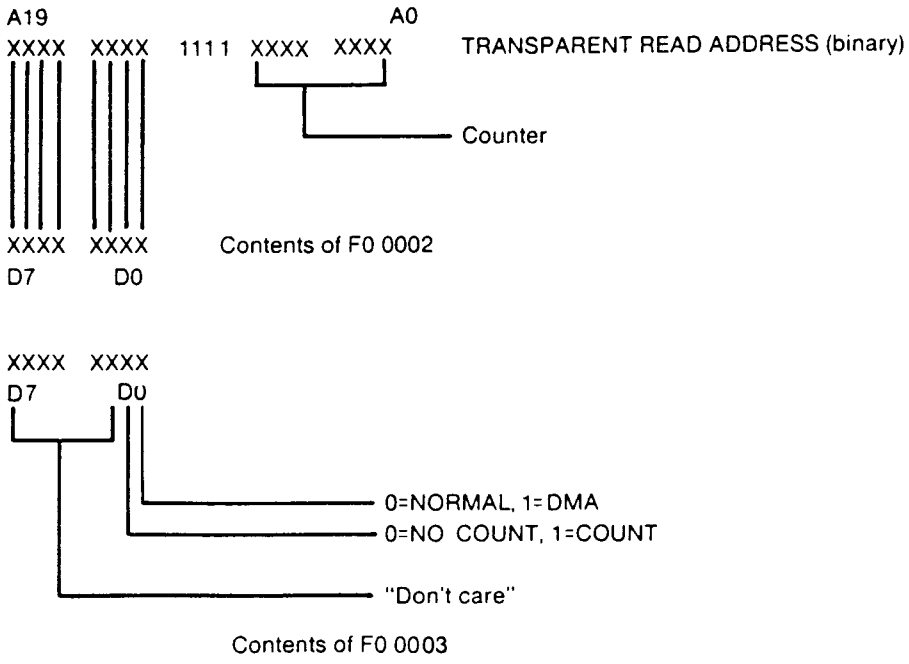
The Pod characteristics are changed by writing to Pod Function addresses. The individual functions are described below.

Changing the Standby Read Address (ADDRESSES F0 0002/F0 0003) 4B-14.

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

If the default value that is provided by the Pod does not access RAM on your UUT, or if that particular address should not be read (because, for example, it might cause an unwanted operation on the UUT), then you may change the address that is used. You may specify the location used for these transparent read operations by writing to Pod Function address F0 0002. (This same address will also be used to perform non-specific read operations that are used in other operations. See Theory of Operation.)

Pod Function addresses F0 0002 and F0 0003 contain the most-significant 8 bits of the address used for Transparent Read operations (A12 - A19) and two control bits. A8 -A11 of the transparent read address are always "1." Bits A0 - A7 of the transparent read address can be configured to count during transparent reads to simulate repetitive DMA operations that would otherwise occur if the CPU were installed in the UUT.



The default value for Pod Function address F0 0002 is FF and F0 0003 is 00 (Normal accesses, no counting, A12-A19=FF). When the counter is disabled, it produces all zeros, so the default transparent read address is F FE00. When the counter is enabled, the address repeats the cycle from F FE00 through F FFFF.

The value of this data is not changed by normal UUT or Pod Reset operations. The value remains constant until power is removed from the Pod, or until the user changes it with a write operation to this address (F0 0002).

The contents of F0 0002 and F0 0003 may be read to view the current specification for the transparent read address.

NOTE

During Transparent Read operations, the Pod will enable any chip-select line that has been programmed to be enabled at the Transparent Read address. Also, Transparent Read cycles will occur with the number of Wait states programmed for that chip select line. For example, if the Pod Function address F0 0002 is set to 2 (by a WRITE @ F0 0002=2), the Transparent Read address will be 02 E00 (hex). If the Lower Chip Select line ($\overline{LCS}$) is defined to be enabled from addresses 0000 through 4000, with one Wait state, then Transparent Read cycles will occur with the $\overline{LCS}$ line enabled (low) with one Wait state.

Note that the Transparent Read address cannot be programmed to enable the PCS1-PSC6 chip select lines.

Note also that Pod operations can be made to execute faster by selecting a Transparent Read address that causes zero Wait states to be inserted into the Transparent Read cycles.

Enable RESET Output During Reset (ADDRESS F0 0004) 4B-15.

Some UUTs may require the RESET output signal to be asserted (high) whenever the Pod is reset by either the Troubleshooter or a power-up. Other UUTs may need this Reset signal to be held low at all times. This Pod Function address gives you the option of specifying whether or not the RESET output signal will be asserted high by the Pod when the Pod is reset by the Troubleshooter. Refer to Appendix E for detailed information about Pod Resets.

Writing non-zero data to this address causes the Pod to assert the RESET output to a high level whenever the Pod is reset. The default value is 00. Thus the Pod initially produces a low level on the RESET output whenever the Pod is reset, except during RUN UUT.

Read contents of F0 0004 to view the current status of the Reset Enable.

CONFIGURING INTERRUPT, DMA, AND RUN UUT FUNCTIONS 4B-16.

Whenever you use the Pod to emulate an 80188 (the RUN UUT mode) some of the Peripheral Control Block registers that regulate these functions might have to be defined first. Since specifying these registers is an integral part of using the Pod in these capacities, details of configuring the Pod for those uses are described in Section 4A. See Testing Interrupt Circuitry, Testing DMA Circuitry, and Using the RUN UUT mode in Section 4A. See Appendix H for detailed information about the Peripheral Control Block.

MASKING ERRORS 4B-17.

Introduction 4B-18.

The are several masks available that you may use to control how the Pod and Troubleshooter report errors. If you wish to suppress the reporting of individual errors, possibly because a particular error is recurring or may be the result of a design quirk in the UUT, you may instruct the Pod and Troubleshooter to NOT stop and display an error message when that error occurs. You can use this capability to avoid getting stuck at certain errors during troubleshooting.

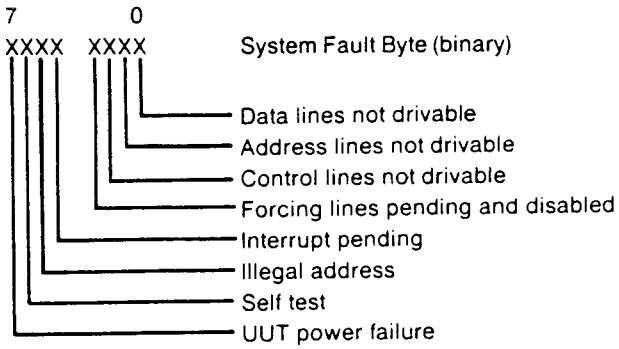
Like the other Pod functions, these masks are used by writing to and reading from Pod Function addresses. For all of these masks, a "0" in a bit position signifies that the error that is assigned to that bit space will not be reported. Details of the individual masks are described below.

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

Error Summary Mask (ADDRESS F0 0060) 4B-19.

After each Pod-UUT operation, the Pod sends a byte of information to the Troubleshooter to indicate possible error conditions or significant changes in status. The Troubleshooter uses this information to generate the correct response to each condition. For example, if the UUT Power Fail bit is set, the Troubleshooter displays a BAD POWER SUPPLY message (the Power Fail bit is set by the Pod if the voltage at one or both of the power supply pins at the UUT's microprocessor socket is not between +4.5v dc and +5.5v dc).

A Write of data YY to address F0 0060 masks the corresponding bits in the error-summary byte from being reported to the Troubleshooter. The data bits denoted by YY represent the Pod's error-summary byte bits. A 0 means that the corresponding bit in the error summary byte is to be masked (ignored during error reporting). The individual bits are assigned as follows:



For example, *WRITE @ F0 0060*=7F indicates that you want the Pod and Troubleshooter to ignore the occurrence of an apparent Bad Power Supply. You might possibly want do this if you are troubleshooting a UUT that intentionally uses a lower

power supply voltage. Without masking off this error, the Pod and Troubleshooter will always stop and report the Bad Power Supply and will not allow you to do other tests.

The power-up default value of the data at this address is FF—all information enabled.

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

Control Drivability Error Mask (ADDRESSES F0 0063/F0 0062) 4B-20.

A Write to addresses F0 0063 and F0 0062 masks individual control line bit errors from being reported to the Troubleshooter. The data corresponds to the control line bit assignments in Table 2-3 as shown below:

```
Control line bit 15 ..... 8 7 ..... 0
      F0 0003  7 ..... 0
      F0 0002          7 ..... 0
```

A data bit value of 0 masks the corresponding control bit. The power-up default for both masks is FF. A Read at these addresses returns the current masks.

Forcing Line Error Mask (ADDRESS F0 0064) 4B-21.

Data written to address F0 0064 individually masks forcing lines from being reported as active. The lines that are masked correspond to any data bits that are specified as 0 in the bit assignments described below. (Note that the bit assignments for the error mask are identical to the status line bit assignments in Table 2-2—except that only the forcing lines are included.)

DATA BIT	FORCING LINE
0	ARDY
1	SRDY
2	HOLD
3	$\overline{\text{RES}}$
4	DRQ0
5	DRQ1

Data bits 6-7 are ignored and normally set to 0. The power-up default mask is 3F (no lines masked).

A Read at this address returns the current mask.

NOTE

If either ARDY or SRDY is masked, reporting will be inhibited on both lines. This is because ARDY and SRDY must both be low on the UUT to cause a forcing line error; if either is masked, the other cannot be reported as an active forcing line by itself, even if it is low on the UUT.

Active Interrupt Error Mask (ADDRESS F0 0066) 4B-22.

A Write at address F0 0066 individually masks interrupts from being reported as active. The interrupt lines correspond to the data bits as shown below. A 0 in the mask indicates that the corresponding interrupt line is not to be reported as active. Bits 5-7 are ignored. The power-up default mask is 1F.

A READ @ F0 0066 returns the current mask.

DATA BIT	INTERRUPT LINE
0	INT0
1	INT1
2	INT2
3	INT3
4	NMI

NOTE

If either interrupt channel is programmed as an Interrupt/Interrupt Acknowledge pair, the corresponding INT2 or INT3 pin is configured as an $\overline{INTA}$ output. Interrupt lines configured in this way are automatically defeated (masked) from being reported as active.

Address Segment Drivability Error Mask (ADDRESS F0 0068) 4B-23.

A Write at address F0 0068 masks the segment (high nibble) of the UUT's address bus. The address bits are assigned to the data in this Pod Function address as follows:

A19 ..... A16
D3 ..... D0

The Default value is 0F.

Low Word Address Drivability Error Mask (ADDRESSES F0 006B/F0 006A) 4B-24.

A Write at addresses F0 006A and F0 006B masks drivability errors on the low word of the UUT's address bus. The address bits are assigned to the data at this Pod Function address as follows:

Address bits	A15 ..... A8	A7 ..... A0
F0 006B	7 0	
F0 006A		7 0

A *WRITE@ F0 006B* = FD and a *WRITE@ F0 006A* = FF (the default), for example, will instruct the Pod and Troubleshooter not to report drivability problems on address line A9 if they occur.

The Default value for both addresses is FF.

Data Drivability Error Mask (ADDRESS F0 006C) 4B-25.

This mask is a map of the UUT's data bus bits. The data bits are assigned to the data in this Pod Function address as follows:

Data bits:	D7 ..... D0
Data mask bits:	D7 ..... D0

The power-up default condition for this mask is FF (errors on all lines reported). A *READ @ F0 006C* returns the current mask.

INTA and TIMER OUT Error Mask (ADDRESS F0 006E) 4B-26.

A Write to address F0 006E individually masks drivability errors on the $\overline{INTA0}$, $\overline{INTA1}$, TMR OUT 0, and TMR OUT 1 lines from being reported as a Control error on the INT ERROR or TMR OUT ERROR pseudo-control lines. A data bit specified as a 0 masks drivability errors on the corresponding line from being reported. The data bit assignments for drivability errors:

DATA BIT	LINE
0	TMR OUT 0
1	TMR OUT 1
2	$\overline{\text{INTA0}}$
3	$\overline{\text{INTA1}}$

Data bits 4-7 are ignored. The power-up default mask is 0F.

A READ @ F0 006E returns the current mask.

Chip Select Error Mask (ADDRESSES F0 0071/F0 0070) 4B-27.

A Write to addresses F0 0071 and F0 0070 individually masks drivability errors on the CHIP SELECT lines from being reported as a Control error on the CHIP SEL ERROR pseudo-control line by the Troubleshooter. Data bits written with a value of 0 inhibit the reporting of drivability errors on the corresponding CHIP SELECT line. The data bits correspond with the chip select lines as follows:

ADDRESS	DATA BIT	LINE
F0 0070	0	$\overline{\text{UCS}}$
	1	$\overline{\text{LCS}}$
	2	$\overline{\text{MCS0}}$
	3	$\overline{\text{MCS1}}$
	4	$\overline{\text{MCS2}}$
	5	$\overline{\text{MCS3}}$
	6	$\overline{\text{PCS0}}$
	7	$\overline{\text{PCS1}}$
F0 0071	0	$\overline{\text{PCS2}}$
	1	$\overline{\text{PCS3}}$
	2	$\overline{\text{PCS4}}$
	3	$\overline{\text{PCS5}}$/A1
	4	$\overline{\text{PCS6}}$/A2

Bits 5-7 in F0 0071 are ignored.

The power-up default mask for F0 0070 is FF and for F0 0071 is 1F.

A Read at address F0 0070 returns the current mask.

DETERMINING ERRORS 4B-28.

Introduction 4B-29.

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

The information at these Pod Function addresses (with the exception of the Last Error Summary at address F0 0040) is not changed until another operation that causes the Pod to execute a UUT-access cycle (a Read or a Write at a UUT address, for example)

is performed. The data at the Last Error Summary address is updated every time an operation that causes the Troubleshooter to communicate with the Pod is performed.

NOTE

Reading the contents of any "last error" Pod Function address does not change the contents of any of the "last error" address, except for the Last Error Summary address at F0 0040.

NOTE

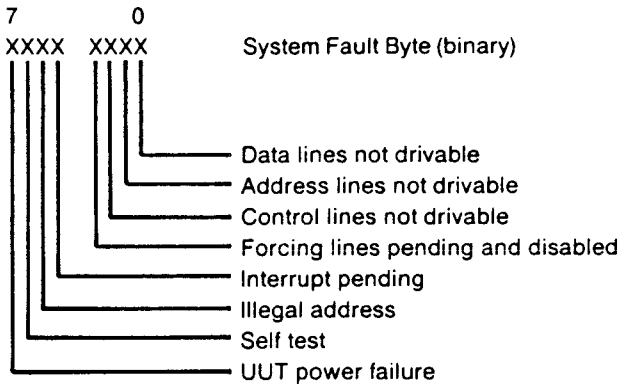
All of the "last error" Pod Function addresses are "Read Only". A Write to any of these addresses will cause the Troubleshooter to display an Illegal Address error message.

The information for the individual Pod Function addresses is described below.

Last Error Summary (ADDRESS F0 0040) 4B-30.

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

A Read at this address (F0 0040) returns the hex value of the Error Summary byte, with the individual bits assigned as follows:



For example, *READ @ F0 0040*=08 indicates that there was an asserted interrupt line during the most recent UUT access.

The power-up default value of the data at this address is 00—no errors.

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

Last Control Errors (ADDRESSES F0 0043/F0 0042) 4B-31.

A Read at addresses F0 0043 and F0 0042 returns the hex representation of the control errors for the most recent Pod/UUT operation. The bits contained in these addresses

Last Chip Select Drivability Errors (ADDRESSES F0 0051/F0 0050) 4B-38.

A Read at addresses F0 0051 and F0 0050 reports drivability errors on the CHIP SELECT lines. A data bit with a value of 1 means that the corresponding line is not drivable. The data corresponds to the lines as follows:

ADDRESS	DATA BIT	LINE
F0 0050	0	$\overline{\text{UCS}}$
	1	$\overline{\text{LCS}}$
	2	$\overline{\text{MCS0}}$
	3	$\overline{\text{MCS1}}$
	4	$\overline{\text{MCS2}}$
	5	$\overline{\text{MCS3}}$
	6	$\overline{\text{PCS0}}$
	7	$\overline{\text{PCS1}}$
F0 0051	0	$\overline{\text{PCS2}}$
	1	$\overline{\text{PCS3}}$
	2	$\overline{\text{PCS4}}$
	3	$\overline{\text{PCS5}}$/A1
	4	$\overline{\text{PCS6}}$/A2

Data bits 5-7 in F0 0051 will always be returned as zeros.

Last Status (ADDRESSES F0 0053/F0 0052) 4B-39.

A Read at addresses F0 0052 and F0 0053 returns the status line values for the most recent Pod operation. The normal status bit assignments as shown in Table 2-3 are used. They correspond to the these two addresses as follows:

Status line bit	15 8	7 0
F0 0053	7 0	
F0 0052		7 0

Section 5
Theory of Operation

INTRODUCTION 5-1.

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

GENERAL POD OPERATION 5-2.

Introduction 5-3.

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

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

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

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

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

Processor Section 5-4.

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

For the 80188 Pod, as well as all other Troubleshooter Pods, the microprocessor inputs from the UUT are referred to as Status Lines, and the outputs to the UUT are referred to as Control Lines. This nomenclature is not always in agreement with the manufacturer's literature (80188 manufacturers, for example, refer to line S3, which is a microprocessor output, as a status line). This convention, however, allows consistency between Pods when implementing the Troubleshooter functions that involve status or control lines, such as READ STATUS or WRITE CONTROL. Refer to the definition of status and control lines in Section 2 for more information about these signals.

All Pod status lines that could adversely affect the Pod operation are either automatically disabled by the Pod or may be disabled by the operator using the Troubleshooter's Setup function. Disabling these status lines allows the Pod to operate in UUT environments where malfunctioning status lines such as HOLD could prevent the Pod from performing any tests. The one microprocessor input that may not be disabled, of course, is the UUT clock. The clock signal must always be present for Pod operation. All the status lines are enabled in the RUN UUT mode.

The Processor Section also contains circuitry for Pod self test. When the Pod ribbon cable plug is inserted into the self test socket, part of the Pod circuitry becomes a simplified pseudo-UUT. During Pod self test, certain tests are performed on this pseudo-UUT, and any failures are reported to the Troubleshooter.

UUT Interface Section 5-5.

The Interface Section, shown in Figure 5-1, consists of buffers and drivers, protection circuits, logic level detection circuits, and a clock-drive oscillator. The buffers and drivers switch the UUT to the microprocessor or to the standby control and address signals, as dictated by the Timing and Control Section.

Each UUT signal is protected from overvoltage or short-circuit conditions that might damage Pod components. Resistors in series with the inputs of the detection circuit latches limit the input current, and resistors in series with the output drive lines limit output current. A pair of clipping diodes connected to ground and power protect the Pod's circuitry against damaging voltages.

The detection circuits consist of latches connected through the protection circuitry to the UUT. The latches are latched during a UUTON cycle, when the signals are expected to be at a known level.

If a signal cannot be driven through the output-current-limiting resistor, it will be detected when the latches are individually read by the Processor Section, and the values are compared with the expected values.

The clock-drive module at the end of the plug cable takes whatever clock signal that may be used on the UUT, either crystal signals on the X1 and X2 pins or a square wave on the X1 pin, and amplifies it to a suitable level to overcome the load of the cable and Pod components.

Timing and Control Section 5-6.

The Timing and Control Section, shown in Figure 5-1, consists of a timer and internal timing and control logic. The Timing and Control Section receives inputs from the Processor Section and the Interface Section. When a UUT access occurs, the internal data bus is disabled and the buffers to the UUT are enabled. During alternate times, the UUT buffers are disabled and the internal data bus is enabled. This bus switch is accomplished by buffer control signals and chip enable signals generated by the Timing and Control Section in response to inputs from the timer, the control register outputs as set by the microprocessor, the status lines from the UUT, and the control lines from the microprocessor.

The length of a normal bus switch equals one microprocessor memory access cycle or I/O cycle. The bus is switched to communicate with the UUT between the last internal operation and the start of the intended UUT operation. The bus is switched back to communicate with the Troubleshooter at the end of the cycle.

If the microprocessor has sent the RUN UUT command to the Pod's control register (or port), the bus switch is started in the normal fashion, but is then held on indefinitely until a Reset signal is received from the Troubleshooter.

During the time that the Pod is not communicating with the UUT, the UUT needs the proper signals so that it can perform the normal dynamic memory refresh operations and other similar tasks. In order to provide these signals to the UUT, the Pod performs what are called transparent reads. A transparent read is a read operation that is performed at a selected address. Transparent reads generate the transparent or fake control signals required to simulate a normal microprocessor read operation. This allows the UUT to maintain non-CPU operations, even when the Pod's microprocessor is not communicating with it.

UUT Power-Sensing Section 5-7.

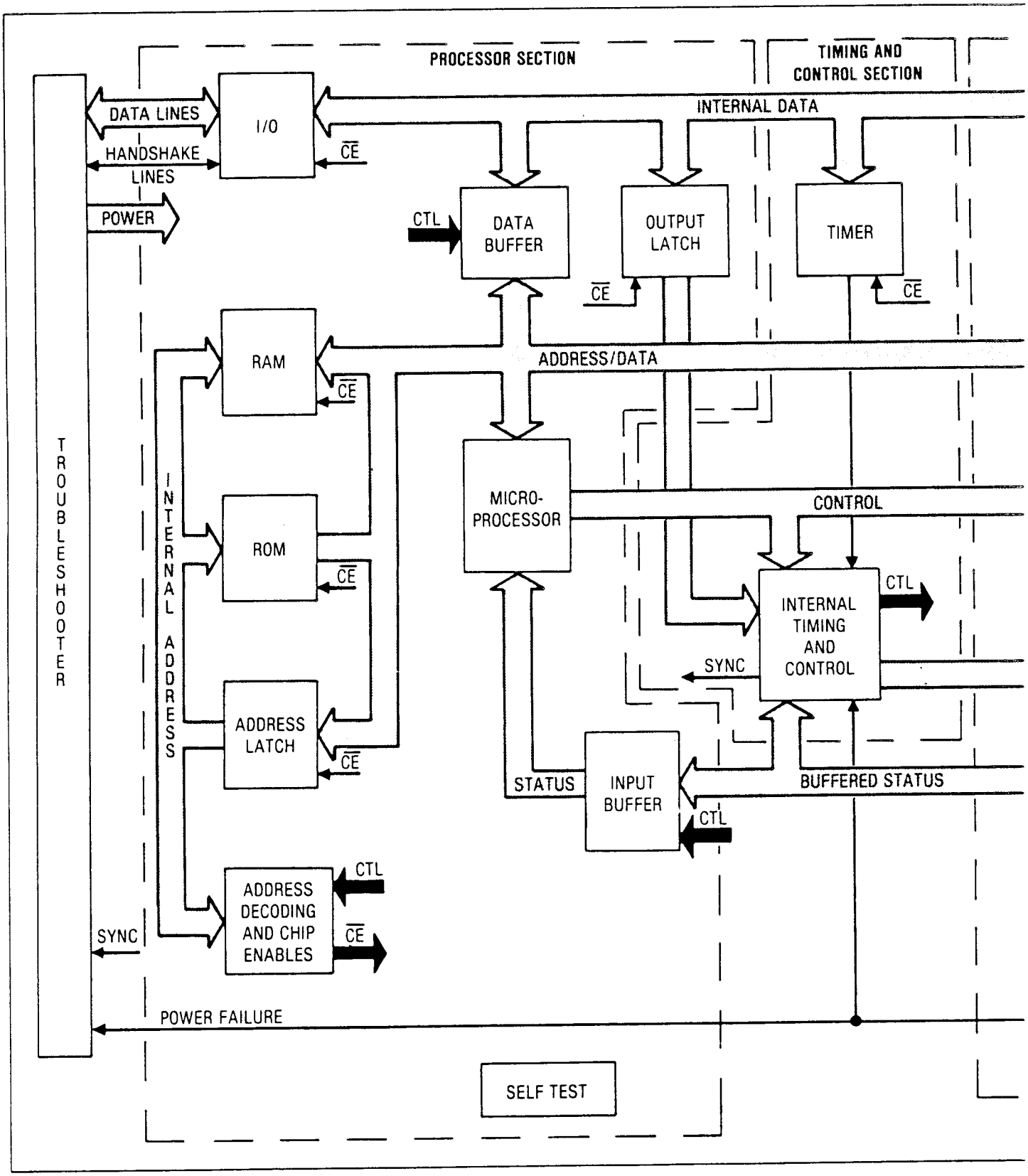
The UUT power-sensing circuit shown in Figure 5-1 constantly monitors the UUT power supply. This circuit produces an output signal to the Troubleshooter in the event that UUT power drops below 4.5V or rises above 5.5V.

Any time that the UUT power supply drops below about 3.5V, all active Pod outputs are changed to their high-impedance states (except the signal on the X2 and CLKOUT pins at the UUT). This feature protects UUT circuits from being damaged by Pod outputs when the UUT power supply drops below safe operating limits. The Troubleshooter will display a UUT power-fail error message. When the proper operating power supplies have been restored to the UUT, the outputs of the Pod will return to normal, and the Troubleshooter will be ready for additional testing.

DETAILED THEORY OF OPERATION 5-8.

Introduction 5-9.

A detailed block diagram of each major Pod section is presented in Figure 5-2. Where possible, the reference designation numbers of specific components are shown in Figure 5-2. The use of the reference designation numbers indicates that most or all of that component is used within the part of the circuit shown. Portions of unlisted components may also be used in the circuit. Each major section is described in the following paragraphs.


PROCESSOR SECTION
TIMING AND CONTROL SECTION
DATA LINES
HANDSHAKE LINES
POWER
I/O
CE
INTERNAL DATA
CTL
DATA BUFFER
OUTPUT LATCH
TIMER
RAM
ADDRESS/DATA
ROM
MICRO-PROCESSOR
CONTROL
INTERNAL TIMING AND CONTROL
SYNC
ADDRESS LATCH
STATUS
INPUT BUFFER
BUFFERED STATUS
ADDRESS DECODING AND CHIP ENABLES
TROUBLESHOOTER
INTERNAL ADDRESS
POWER FAILURE
SELF TEST

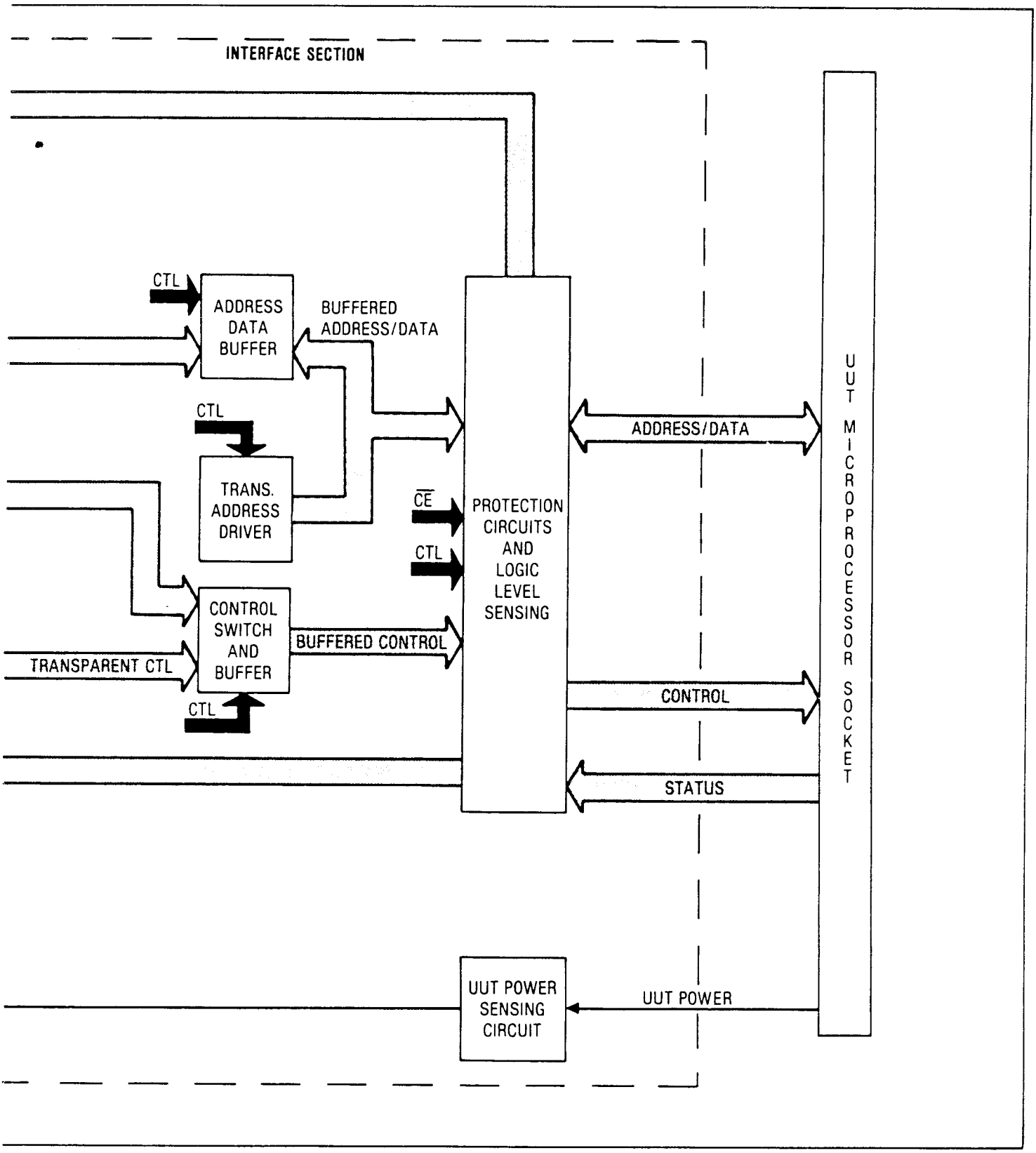
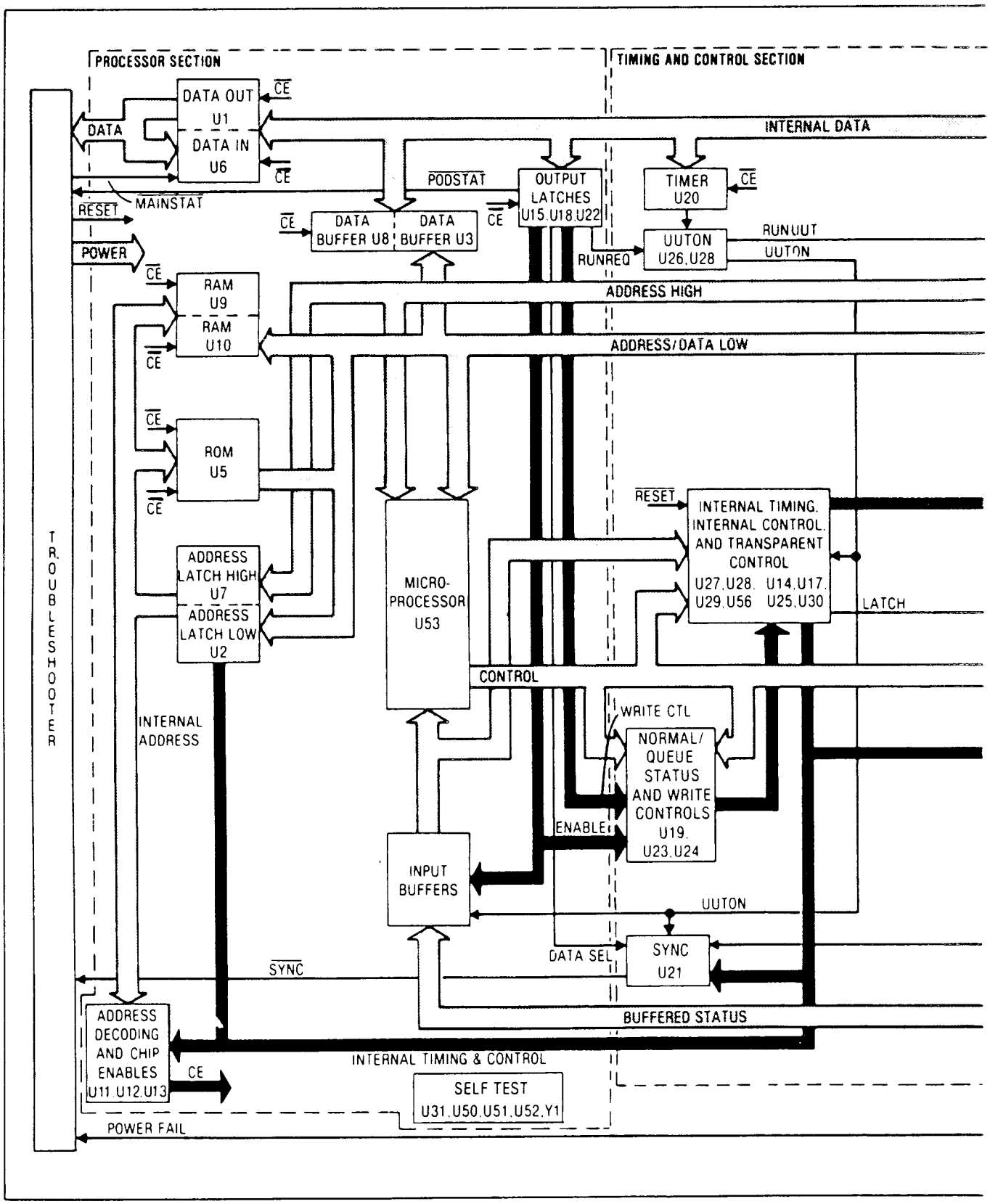


Figure 5-1. 80188 Interface Pod General Block Diagram


PROCESSOR SECTION
TIMING AND CONTROL SECTION
DATA OUT U1
DATA IN U6
DATA
CE
INTERNAL DATA
RESET
MAINSTAT
PODSTAT
OUTPUT LATCHES U15.U18.U22
TIMER U20
DATA BUFFER U8
DATA BUFFER U3
RUNREQ
UUTON U26.U28
RUNUUT
UUTON
POWER
RAM U9
RAM U10
ADDRESS HIGH
ADDRESS/DATA LOW
ROM U5
TROUBLESHOOTER
ADDRESS LATCH HIGH U7
ADDRESS LATCH LOW U2
MICRO-PROCESSOR U53
INTERNAL TIMING, INTERNAL CONTROL, AND TRANSPARENT CONTROL U27,U28, U14,U17, U29,U56 U25,U30
LATCH
CONTROL
INTERNAL ADDRESS
WRITE CTL
NORMAL/ QUEUE STATUS AND WRITE CONTROLS U19, U23,U24
ENABLE
INPUT BUFFERS
UUTON
DATA SEL
SYNC U21
SYNC
BUFFERED STATUS
ADDRESS DECODING AND CHIP ENABLES U11,U12,U13
INTERNAL TIMING & CONTROL
SELF TEST U31,U50,U51,U52,Y1
POWER FAIL

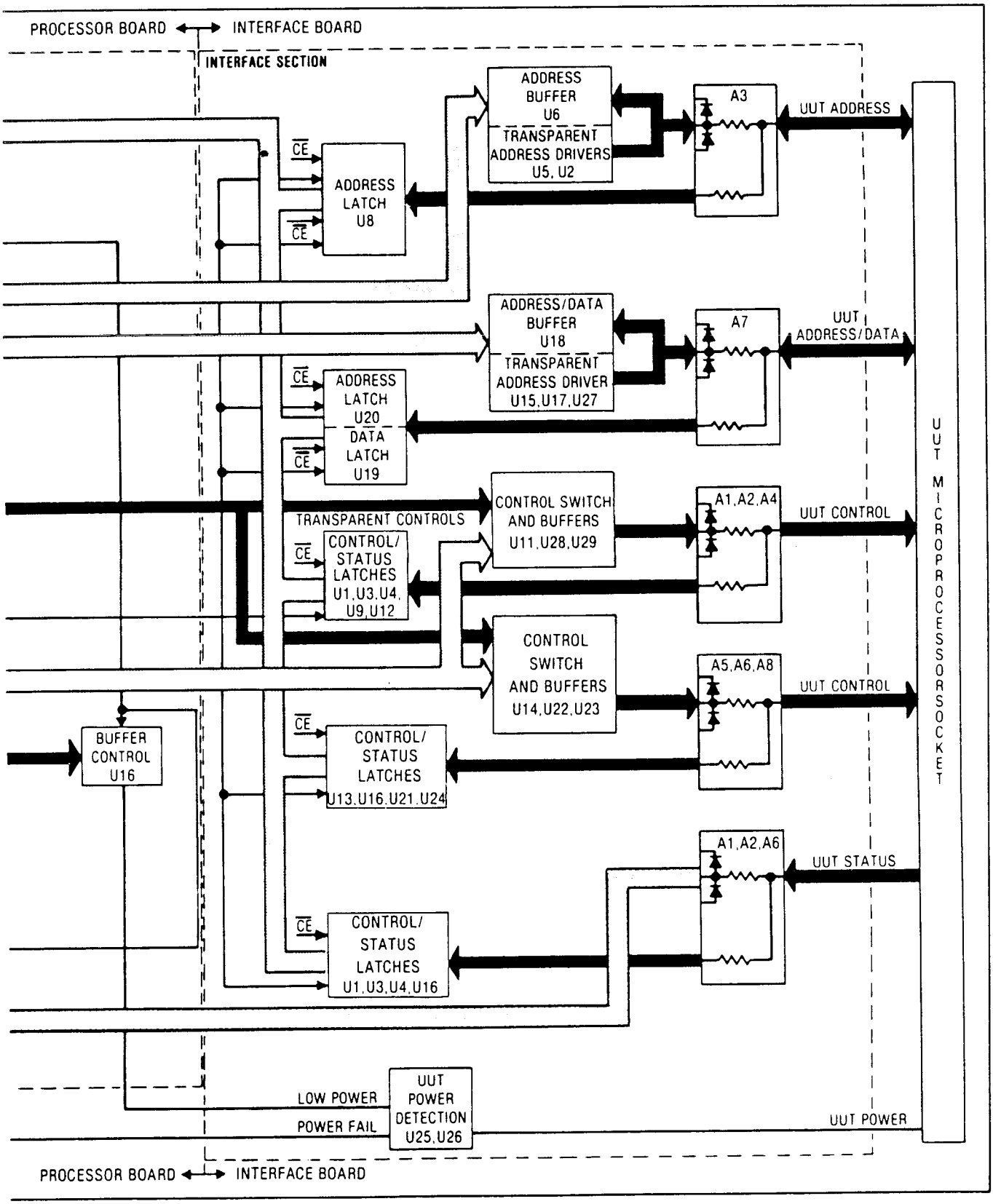


Figure 5-2. 80188 Interface Pod Block Diagram

Processor Section 5-10.

The microprocessor (U53), RAMs (U9, U10), and ROM (U5) comprise a small microprocessor system that is the heart of the Pod. Because the address/data bus is time-multiplexed, the address latches (U2, U7) are required to supply valid addresses to the RAM and ROM during the entire memory cycle. The data side of the RAMs and ROMs is tied directly to the address/data bus because the RAMs and ROMs cannot tolerate the extra delay of the data buffers. All other data is input through data buffers (U3, U8) to reduce loading on the bus. The Processor Section communicates with the troubleshooter through additional buffers (U1, U6).

All communication between the Pod and the Troubleshooter is fully handshaked according to the protocol shown in Figure 5-3. The two handshake lines are $\overline{\text{MAINSTAT}}$ and $\overline{\text{PODSTAT}}$. $\overline{\text{MAINSTAT}}$ is produced by the Troubleshooter and monitored by the the Pod. $\overline{\text{MAINSTAT}}$ initiates all data transactions. $\overline{\text{PODSTAT}}$ is produced by the Pod to indicate the Pod's response.

The address decoding and chip enable circuitry (U11, U12, U13, and associated logic) select which components or buffers are enabled on the data bus. This circuitry also controls the direction of the data buffers, and disables all internal components during UUTON. (The chip select lines [$\overline{\text{UCS}}$, $\overline{\text{LCS}}$, etc.] are not used internally by the Pod for address decoding.)

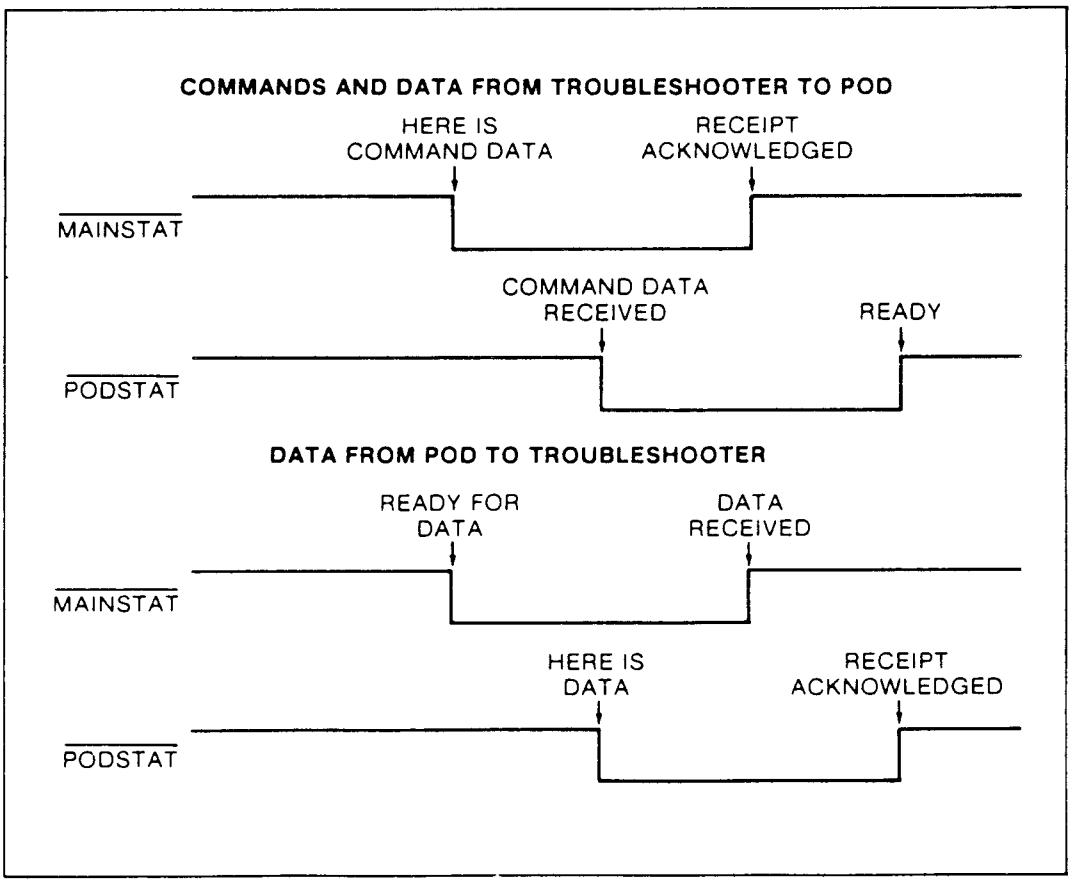


Figure 5-3. Handshake Signals

The output latches (U15, U18, U22) are used by the microprocessor to control the function of the Pod. The output latches control which inputs are enabled, the value of lines during a Write Control operation, which sync pulse is returned to the troubleshooter, and the Run UUT mode. $\overline{\text{PODSTAT}}$ is also one of the outputs of the latches.

The various protection networks help protect the microprocessor from potentially damaging signal levels at the UUT (see the description of the Interface section below). Intervening logic between the protection circuits and the microprocessor allows or disallows inputs, depending on which lines are enabled. This logic also allows all inputs if the Run UUT mode is selected.

The self test circuitry consists of a clock generator (Y1, C2, C3) and some latches that buffer the address (U50, U51, U52). The outputs of these latches are placed on the data bus during a read operation. All status lines are driven by common signals that cause the status lines to be active and ensure that the Pod can operate in the potentially hostile UUT environment. Vcc is returned to check the power level. All output lines are cross-connected to input lines so that continuity of the cable and functioning of the interface board can be checked.

Timing and Control Section 5-11.

The components that compose the Timing and Control Section are located on the processor board. The Timing and Control Section is divided into functional subsections for the purposes of discussion, although some of the subsections are intricately interrelated.

The timing relationships of several important Pod signals are shown in Figure 5-4. The signals include UUTON, the latch controls, and sync pulse. The UUTON circuit (U26, U28, and associated logic gates) controls the bus switch timing and enable signals. The UUTON signal is initiated by an output of the timer (U20), and is normally ended by the signal $\overline{\text{LODEN}}$, which is generated by the internal timing circuits. $\overline{\text{LODEN}}$ is an extended or long data enable pulse. If U26, pin 5 is high, UUTON stays high until a reset is received from the Troubleshooter.

The sync circuitry (U21 and associated gates) sends a sync signal to the troubleshooter during the address or data portion of UUTON. (Address or data sync is selected by the Processor Section.) Sync is inhibited by the signal RUNUUT to prevent damage to the probe pulser.

The buffer control circuitry (U16) controls the bus switch by controlling the buffer direction and enable signals of the buffers in the Interface Section. Component U16 is a PAL* (Programmable Array of Logic); a schematic diagram of U16 is included in Section 8 of this manual.

The internal timing, internal control, and transparent control circuitry generate a stable set of usable control signals for both the Normal and the Queue Status mode of 80188 operation. Most of the signals generated have names that are similar to the names of the microprocessor control lines, with an I or a T added to the name to indicate whether the signal is used internally or is transparent. The internal signals are not sent to the UUT, but are used solely for internal Pod operations. The transparent signals are sent to the UUT during a transparent read. (Transparent reads are described in the previous section titled Timing and Control Section.) Some of the transparent signals are also used internally. The flip-flops U25, U27, U28, and U29 generate most of the transparent signals and switch between the Normal and Queue Status modes.

*PAL™ is a trademark of Monolithic Memories, Inc.

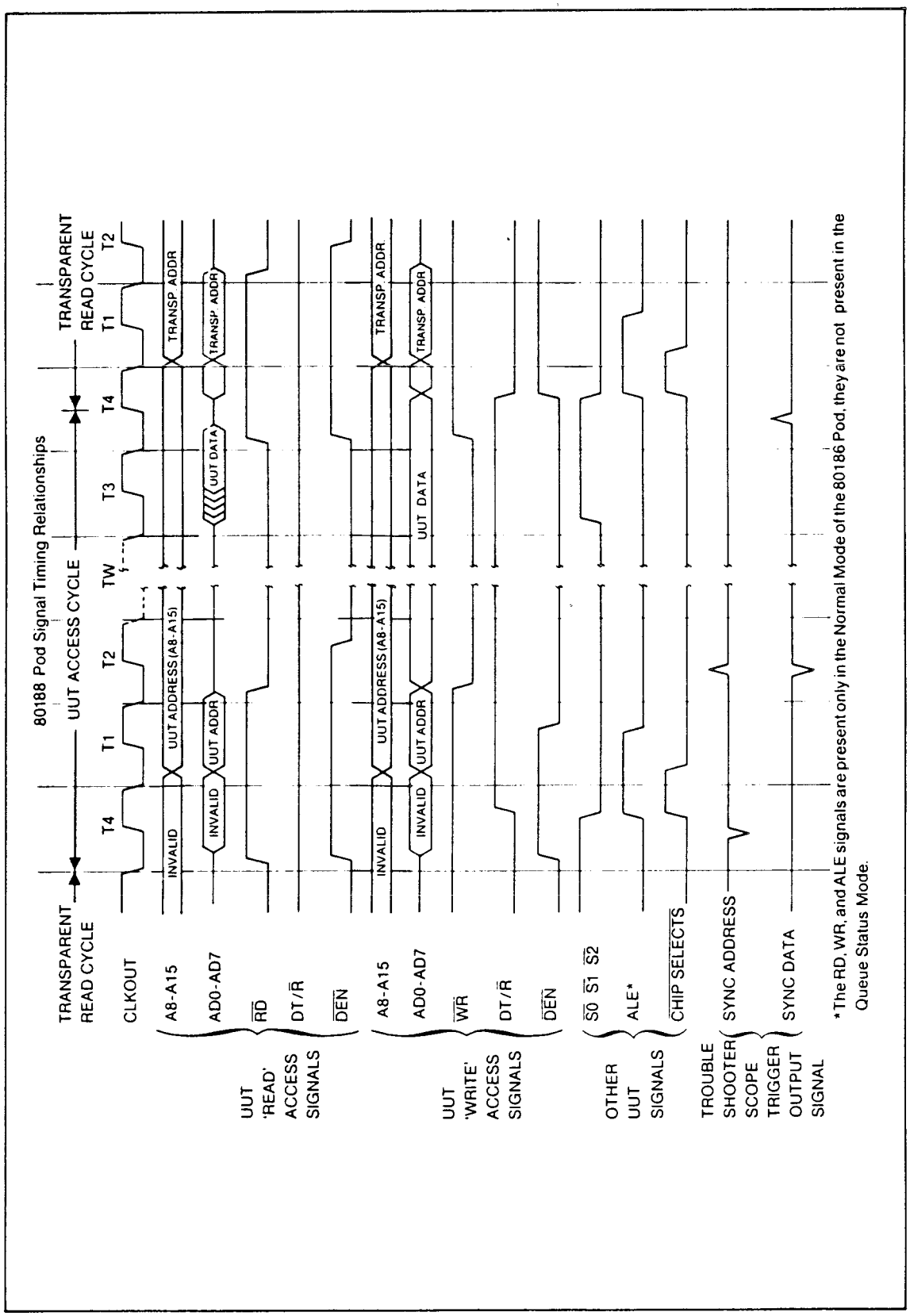


Figure 5-4. 80188 Pod Signal Timing Relationships

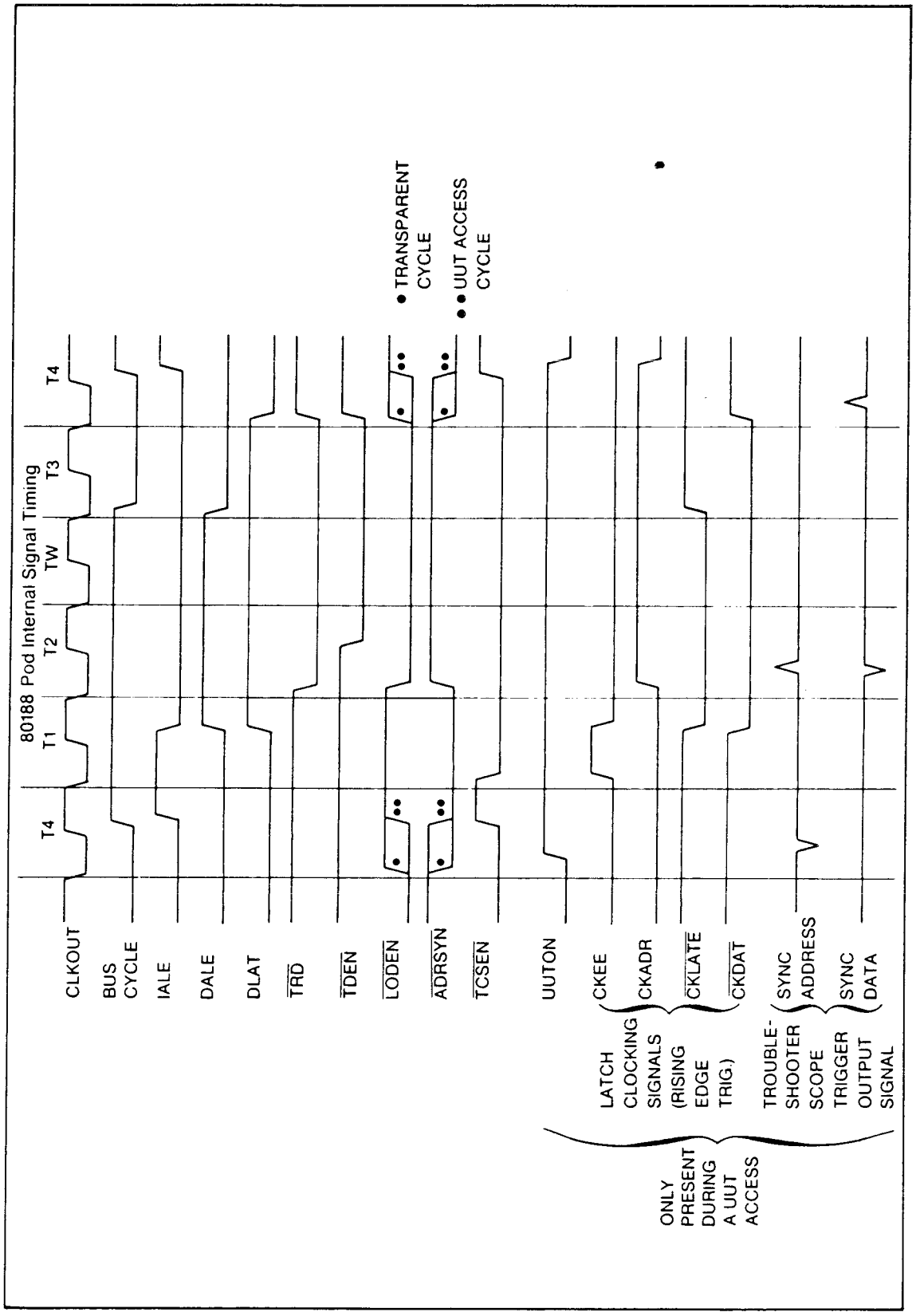


Figure 5-5. 80188 Pod Internal Signal Timing

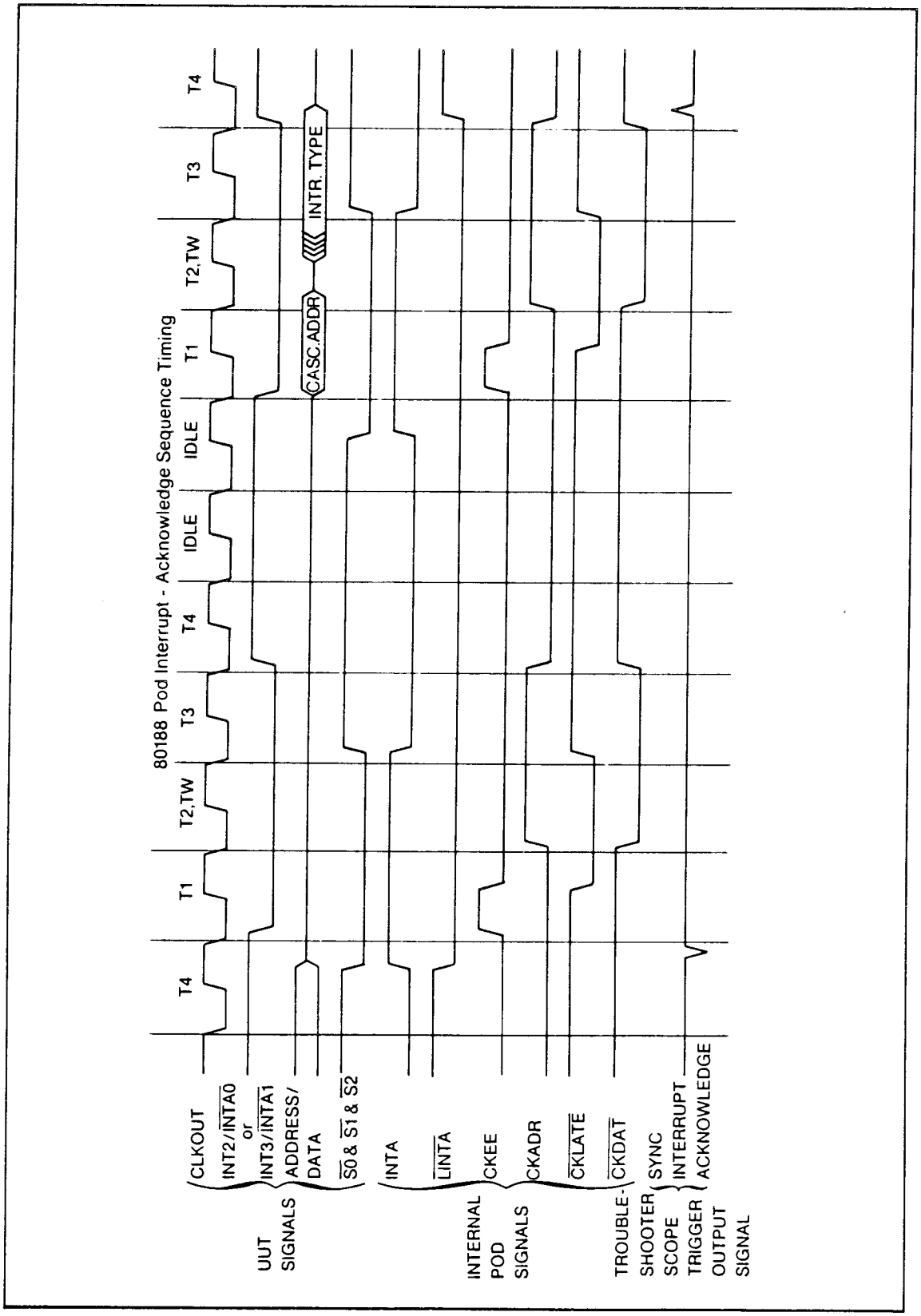


Figure 5-6. 80188 Pod Interrupt - Acknowledge Sequence Timing

The flip-flops U25, U26, U27, U28, U29, U30, and U56, and the associated logic generate internal signals to allow the internal devices of the Pod to communicate with each other (and to generate transparent reads) regardless of whether the Pod has been placed in the Normal mode or the Queue Status mode by the UUT.

The PAL U23 and part of flip-flip U25 switch the Pod between the Normal mode and the Queue Status mode whenever the Pod is reset, depending upon how the $\overline{RD}/\overline{QSMD}$ signal is created by the UUT.

Interface Section 5-12.

Components A1 through A8 are Fluke-designed hybrid ICs containing current-limiting resistors and clipping diodes to protect Pod circuits from overvoltage conditions. A1 through A8 also contain (not shown) 200-kilohm pull-up resistors.

The transparent-read address is produced during transparent reads by the buffer U27, the counter U5, and part of the buffer U15.

The upper eight bits (A12-A19) of the transparent-read address are written to the output latch U17 by the processor section. These transparent-read address bits are then sent through buffer U27. The outputs of U27 are enabled and disabled by the Timing and Control Section. The upper eight transparent-read address bits are selectable by the user (see Configuring General Pod Characteristics in Section 4A).

Transparent-read address bits A8-A11 are produced by the buffer U15. Only half of U15 is used to produce these transparent-read address bits; the other half buffers the address bits A16-A19 during UUT-access cycles. The outputs of U15 are enabled and disabled by the Timing and Control Section.

Transparent-read address bits A0-A7 are produced by the eight-bit counter U5. Normally, this counter is configured by the Processor Section to produce each of A0-A7 as a logic zero, but the counter can be configured to cycle repeatedly through a count from 0 to FF. This allows the Pod to simulate DMA operations during transparent read operations.

When U5 is configured to count, the value of the transparent-read address cycles from XXF00 to XXFFF. This is because the counter outputs are connected to A0-A7. (The X's represent the selectable transparent-read address bits A12-A19.) The value of the counter outputs is incremented by one after each transparent-read cycle.

U14 is a multiplexer which produces the signals S3-S6, which are time-multiplexed with address signals A16-A19. The outputs of U14 are enabled and disabled by the Timing and Control Section.

The inputs to all the latches (U1, U3, U4, U7, U8, U12, U13, U16, U19, U20, U24) are connected to the UUT lines through the 700-ohm resistors of the components A1 through A8. The latches are under control of the Timing and Control Section during a UUT access or during an interrupt acknowledge cycle. The latches may receive multiple latch pulses during a UUT cycle; the last data latched during the UUT cycle is the data read by the Processor Section. The Processor Section compares the latched data with what the it expected to see to detect drivability errors, forcing lines pending, or interrupts pending.

The UUT power-detection circuit constantly monitors the UUT power supply. If the UUT power supply drops below 4.5V or rises above 5.5V, the power-detection circuit produces an output to the Troubleshooter which causes the Troubleshooter to display a UUT power fail error message.

Also, any time the UUT power supply drops below about 3.5V, all active Pod outputs are disabled. This feature has been incorporated to protect UUT circuits from being damaged by Pod outputs when the UUT power supply drops below safe operating limits. If this happens, the Troubleshooter displays a UUT power fail error message. When the proper operating power supplies have been restored to the UUT, the outputs of the Pod return to normal and the Troubleshooter is ready for additional testing.

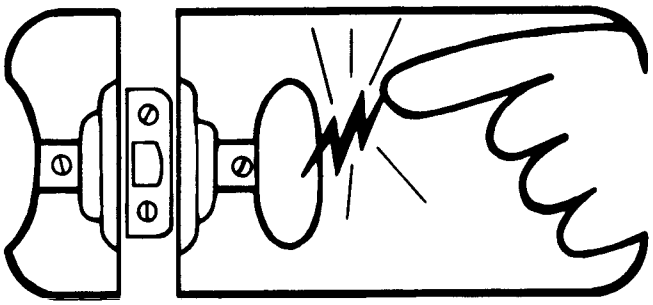
During UUT access cycles, the Chip Select signals ($\overline{\text{UCS}}$, $\overline{\text{LCS}}$, $\overline{\text{MCS0-3}}$, and $\overline{\text{PCS0-6}}$) of the Pod's microprocessor are sent to the UUT through the buffers U10 and U22. These buffers are disabled during transparent reads. Fake chip select signals are produced for transparent reads by the output latches U11 and U23, and the multiplexers U28 and U29. U28 and U29 (along with the flip-flop U58 of the processor section) are used to ensure that the selected (low) transparent Chip Select line only goes low at the proper time.

The buffered Chip Select outputs are latched by U12 and U24. The internal Chip Select lines are latched (on the internal side of buffers U10 and U22) by the latches U9 and U21. The processor section reads these latches, and compares the results of the internal latches (U9 and U21) to the results of the latches U10 and U22 to determine if there are drivability errors on the Chip Select outputs. The Pod must use the internal latches to read the expected Chip Select line conditions because the Pod software doesn't automatically know how the Chip Select lines will be produced by the Pod during a given UUT access cycle.

static awareness

A Message From

John Fluke Mfg. Co., Inc.

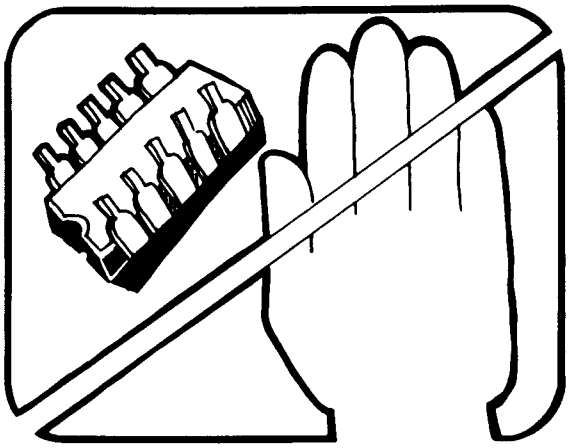


Some semiconductors and custom IC's can be damaged by electrostatic discharge during handling. This notice explains how you can minimize the chances of destroying such devices by:

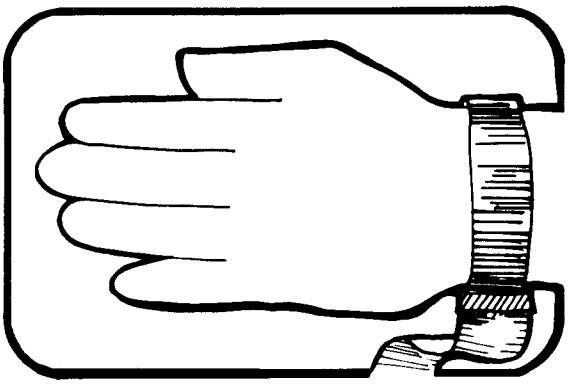
1. Knowing that there is a problem.
2. Learning the guidelines for handling them.
3. Using the procedures, and packaging and bench techniques that are recommended.

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

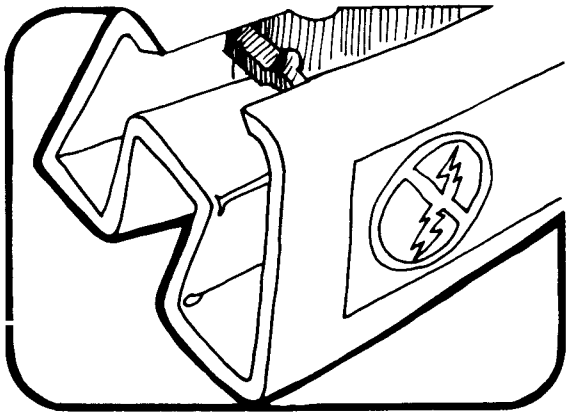
The following practices should be followed to minimize damage to S.S. devices.



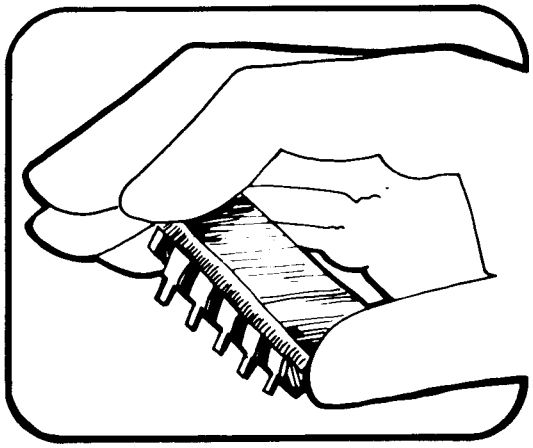
1. MINIMIZE HANDLING



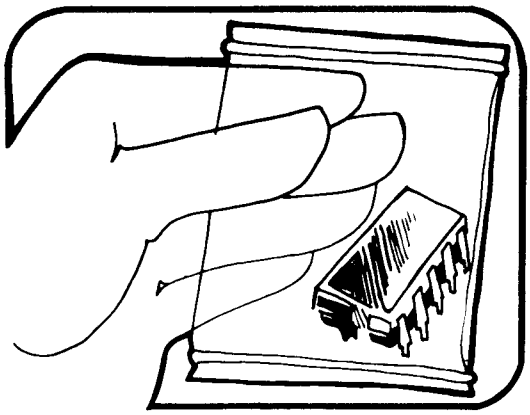
3. DISCHARGE PERSONAL STATIC BEFORE HANDLING DEVICES



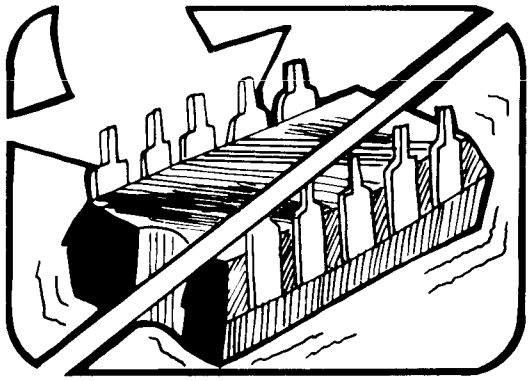
2. KEEP PARTS IN ORIGINAL CONTAINERS UNTIL READY FOR USE.



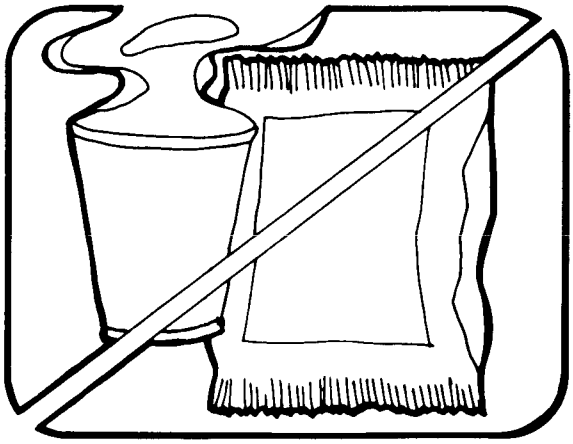
4. HANDLE S.S. DEVICES BY THE BODY



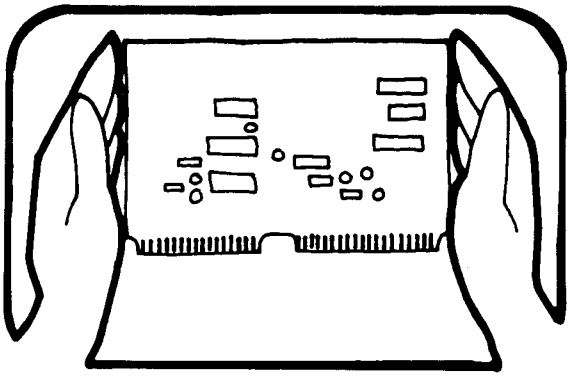
5. USE ANTI-STATIC CONTAINERS FOR HANDLING AND TRANSPORT



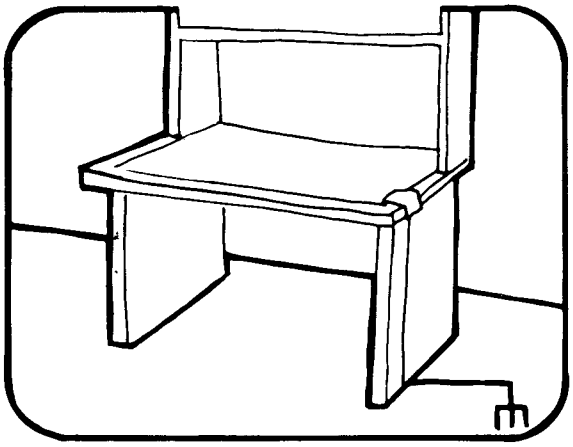
6. DO NOT SLIDE S.S. DEVICES OVER ANY SURFACE



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 USUALLY PROVIDES COMPLETE PROTECTION TO 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.

Anti-static bags, for storing S.S. devices or pcbs with these devices on them, can be ordered from the John Fluke Mfg. Co., Inc.. See section 5 in any Fluke technical manual for ordering instructions. Use the following part numbers when ordering these special bags.

John Fluke Part No.	Description
680892	5" x 8" Bag
680934	8" x 10" Bag
680942	8" x 12" Bag
680983	12" x 16" Bag
681023	18" x 18" Bag
Pink Poly Sheet	Wrist Strap
30" x 60" x 60 Mil	P/N TL6-60
P/N RC-AS-1200	$7.00
$20.00	

® Dow Chemical

J0089C-07U8412/SE EN Litho in U.S.A.

Section 6
Troubleshooting

INTRODUCTION 6-1.

This section provides troubleshooting information for the Pod, and includes repair precautions and disassembly procedures.

The built-in Pod Self Test (described in Section 2 of this manual) will detect most Pod malfunctions. Whenever the Troubleshooter displays a message indicating a Self Test error, or whenever the Pod appears to be defective or inoperative, make a note of the message or symptoms. If the Pod is still covered under the Warranty, or if you want to have the Pod repaired by Fluke, send the Pod to a Fluke Technical Service Center for repair as described below. If you are going to troubleshoot and repair the Pod yourself, continue to paragraph 6-5, Getting Started.

CAUTION

The Pod's Processor PCB uses numerous Surface-Mounted parts. Replacement of these parts requires a reflow soldering technology. Attempting to replace Surface-Mounted components using a soldering iron or other conventional means will damage the PCB.

WARRANTY AND FACTORY SERVICE 6-2.

Troubleshooting and repair during the one-year Warranty period should be done by a Fluke Technical Service Center. (See the Warranty statement at the front of this manual for details of the Warranty.) If the Pod is still covered under the Warranty, send the Pod, along with the description of the symptoms, to a Fluke Technical Service Center. The Troubleshooter Operator Manual or Service Manual contains a list of Fluke Technical Service Centers.

After the Warranty period, if you do not want to service the Pod yourself, or if attempted troubleshooting fails to reveal the Pod fault, you may still ship the Pod to a Fluke Technical Service Center for repair at a reasonable cost. If requested, a free cost estimate will be provided before any repair work is performed.

INSPECTING A SHIPMENT 6-3.

If you are receiving an original delivery of this Pod, inspect the Pod thoroughly immediately upon arrival for damage and missing items.

- If the Pod is damaged in any way, file a claim with the carrier. You are responsible for negotiations and final claims with the carrier. If you want to have shipping

damage repaired by Fluke, contact the nearest Fluke Technical Center for a quotation.

- Check all material in the container against the enclosed packing list. Fluke will not be responsible for shortages unless you notify us immediately.

SHIPPING THE POD TO FLUKE FOR REPAIR OR ADJUSTMENT 6-4.

Ship Pod to Fluke prepaid via United Parcel Service or "Best Way" (Air Freight from overseas). Ship the Pod in its original packing carton, or, if that is not available, a suitable container that is rigid and of adequate size. If you use a substitute container, wrap the Pod in paper and surround it with at least four inches of excelsior or other shock-absorbing material.

GETTING STARTED 6-5.

Troubleshooting the Pod is similar to troubleshooting any other microprocessor-based UUT, and requires the equipment listed in Table 6-1. You should use the Theory of Operation in Section 5 and the schematic diagrams in Section 8 to augment the troubleshooting procedures that are outlined below.

Table 6-1. Required Test Equipment for Pod Troubleshooting

EQUIPMENT TYPE	REQUIRED TYPE
Micro-System Troubleshooter	Fluke 9000 Series
Interface Pod	Fluke 9000A-80188
Digital Multimeter	Fluke 77
Oscilloscope	Tektronix 485 or equivalent

NOTE

All references to data and addresses in the following sections are in hexadecimal notation.

CAUTION

Static discharge can damage MOS components contained in the Pod. To prevent this possibility, take the following precautions when troubleshooting and/or repairing the unit.

- **Never remove, install, or otherwise connect or disconnect PCB (printed circuit board) assemblies without disconnecting the Pod from the Troubleshooter.**
- **Perform all repairs at a static-free work station.**
- **Do not handle ICs or PCB assemblies by their connectors.**
- **Wear a static ground strap when performing repair work.**
- **Use conductive foam to store replacement or removed ICs.**

- **Remove all plastic from the work area (including vinyl and expanded foam, such as Styrofoam*).**
- **Use a grounded soldering iron.**
- **Always place the Pod in a static-free plastic bag for shipping.**

DETERMINING WHETHER THE POD IS DEFECTIVE OR INOPERATIVE 6-6.

The first task of troubleshooting the Pod is to determine whether the Pod is defective or inoperative. This determination is based on the results of the Pod self test described in Section 2. If you have not performed the self test, refer to Section 2 and perform the self test before proceeding with the troubleshooting.

The results of the Pod self test and the Pod behavior when connected to a known good UUT will categorize the problem into one of the three following groups:

Defective Pod:	The Pod fails the Pod self test and the Troubleshooter displays a self-test failure code. Refer to Troubleshooting a Defective Pod in this section.
Inoperative Pod:	The Pod is unable to complete the Pod self test and the Troubleshooter displays an *ATTEMPTING RESET* message. Refer to Troubleshooting an Inoperative Pod in this section.
Suspected Defective Pod:	The Pod passes the Pod self test but exhibits abnormal behavior when connected to a known good UUT. Refer to Extended Troubleshooting Procedures in this section.

NOTE

The 80188 Interface Pod is only designed to be used with a Troubleshooter that has been updated with improved delay lines and probes. Earlier models used a slow TTL part as a delay line, which may provide unstable probe readings at the high clock frequencies (possibly greater than 6 MHz) used with the 80188 CPU. If your Troubleshooter's Serial Number is lower than 3194000 and is demonstrating such symptoms, contact a Fluke Technical Service Center for advice.

TROUBLESHOOTING A DEFECTIVE POD 6-7.

Introduction 6-8.

This section describes what to do if the Troubleshooter displays the *POD SELF TEST 80188 FAIL xx* (where xx represents a self test failure code) message when the Pod self test is performed. If instead, the Troubleshooter displays an *ATTEMPTING RESET* message, refer to Troubleshooting an Inoperative Pod.

NOTE

Make sure that the message is POD SELF TEST 80188 FAIL xx. The messages FLUKE 90XXA POWER-UP FAIL or FLUKE 90XXA RESTARTED FAIL indicate failures within the Troubleshooter, not within the Pod. Refer to the Troubleshooter Operator Manual for help if you see either of these messages.

*Styrofoam is a registered trademark of Dow Chemical Co., Inc.

The procedures for troubleshooting a defective Pod are based on the information reported by the self test failure codes. These self test failure codes provide information that can enable the operator to locate the cause of the Pod failure.

NOTE

The fact that the self test was completed (but still reported an error) is a good indication that the problem is probably located in the UUT Interface Section of the Pod. If the self test was completed, the Processor Section and the Timing Section are probably functioning normally. Those sections are essential for accepting the self test commands and communicating the results to the Troubleshooter.

Interpreting the Self Test Failure Codes 6-9.

INTRODUCTION 6-10.

Information from two self test sequences is available: the standard self test that is controlled by the Troubleshooter and performed during the self-test procedure, and the enhanced self test that is built into the Pod and performed during power-up and reset sequences. The enhanced self test provides more thorough evaluation of the Pod than is provided by the standard self test alone. The results of both self tests are available after doing a normal self test operation on the Pod (see Section 2 for instructions).

NOTE

The error codes described below are used with the 9005A and 9010A Troubleshooters. Different error codes are used when using the Pod with a 9020A. See Appendix D in the 9020A Operator Manual for a description of the 9020A error codes.

Whenever the Troubleshooter displays the message *POD SELF TEST 80188 FAIL 00,* then the Pod may have either failed standard self test or the enhanced self test. Refer to Using the Enhanced Self Test below to find instructions for determining the source of the failure. Any other failure code (01, 02, 03) denotes a failure of the standard self test. Standard self test failure codes are described in detail in Table 6-2.

Table 6-2. Standard Self Test Failure Codes

FAILURE CODE	DESCRIPTION
00*	UUT read access failed or the enhanced self test failed
01	UUT write access failed
02	Control line(s) cannot be driven
03	Enableable status line(s) failed

*When the failure code 00 is received, the Pod may have failed an internal enhanced self test. To determine whether the Pod failed the enhanced self test or not, perform a read at F0 0000. If the data returned is 00FF, the Pod did not fail the enhanced self test. If the message returned is anything other than 00FF OK, the enhanced self test failed; refer to the paragraphs titled Recreating the Enhanced Self Test Routines.

USING THE STANDARD SELF TEST 6-11.

Whenever the Troubleshooter displays the message *POD SELF TEST 80188 FAIL xx* where xx equals 01, 02, or 03, the Pod has failed the standard self test. Record this information and continue to Preparation to Troubleshoot a Defective Pod.

USING THE ENHANCED SELF TEST 6-12.

The Enhanced Self Test performs an improved functional test (compared to the Troubleshooter's built-in Pod test) and a nearly complete cable test as well. It consists of a series of Read, Write, and Write Control operations using the Self-Test socket. If an error is detected, all relevant information is saved, and an error code is inserted into the fault byte at the end of the reset string before it is sent to the Troubleshooter. This code produces the *POD SELF TEST 80188 FAIL 00* message.

Pod Function Address F0 0000 in the Pod contains the result code from the last Enhanced Self Test. A READ @ F0 0000 will return a code indicating which Enhanced Self Test routine failed. Table 6-3 describes the possible error codes produced by the Enhanced Self Test. As with the regular self test data, record the information from F0 0000 and continue to Preparation for Troubleshooting a Defective Pod.

A WRITE @ F0 0000 = 00 will disable the reporting of the Self-Test bit in the fault byte on all operations. This allows the Self-Test socket to be used as a UUT for diagnostics. Reenable the Self-Test bit by writing FF to F0 0000 (WRITE @ F0 0000 = FF), or by turning the Troubleshooter power off and then on again.

Preparation for Troubleshooting a Defective Pod 6-13.

CAUTION

Any adjustment, maintenance, or repair of the opened Pod under voltage should be avoided as far as possible and, if done, must be carried out only by a skilled person who is aware of the hazards involved.

Prepare to troubleshoot your defective Pod as follows:

1. Disassemble the Pod, referring to the later section titled Disassembly. It is not necessary to separate the two PCB assemblies at this point. The two PCB assemblies should remain securely fastened together with screws to avoid possible problems with electrical connections between the two PCB assemblies.

2. Look for any obvious problems such as burned components or ICs that are loose in their sockets. Replace components if necessary.

3. Connect the Pod to the Troubleshooter, and insert the ribbon cable plug into the self test socket as shown in Figure 6-1.

4. Press the Bus Test key on the Troubleshooter to initiate the Self Test, then press the Stop Key. Perform a WRITE @ F0 0000 = 00 to disable the Pod Self Test. (The Pod Self Test may be re-enabled by cycling Pod power off and then on, or by a WRITE @ F0 0000 = FF.)

Table 6-3. Enhanced Self Test Failure Codes

EXECUTION/ FAILURE CODE	OPERATION	EXPECTED RESULT (TROUBLESHOOTING CLUES)
01	READ @ AA55 = AA	*DATA RETURNED = AA (Check A8-A15: Odd bits high, even low)
02	READ @ 55AA = 55	*DATA RETURNED = 55 (Check A8-A15: Even bits high, odd low)
03	READ @ 55 = 55	*DATA RETURNED = 55 (Check A0-A7: Even bits high, odd low)
04	READ @ AA = AA	*Data returned = AA (Check A0-A7: Odd bits high, even low)
05	WRITE @ AA = 55	No Address, Data errors (Check D0-D7: Even bits high, odd low)
06	WRITE @ 55 = AA	No Address, Data errors (Check D0-D7: Odd bits high, even low)
07	READ @ A0000 = AB	*DATA RETURNED = AB (Checks A17, A19 = 1; A16, A18 = 0; $\overline{S0}$, $\overline{S2}$ = 1; $\overline{S1}$ = 0; S7 = 1.)
08	READ @ 50000 = 5B	*DATA RETURNED = 5B (Checks A17, A19 = 0; A16, A18 = 1; $\overline{S0}$, $\overline{S2}$ = 1; $\overline{S1}$ = 0; S7 = 1.)
09	READ @ 200000 = 03	*DATA RETURNED = 03 (Checks $\overline{S0}$, A16-A19 = 0; $\overline{S1}$, $\overline{S2}$ = 0.)
0A	READ @ 40080 = 1F	*DATA RETURNED = 1F (Checks $\overline{PCS1}$ = 0; all other $\overline{PCS}$ lines = 1.)
0B	READ @ 40100 = 2F	*DATA RETURNED = 2F (Checks $\overline{PCS2}$ = 0; all other $\overline{PCS}$ lines = 1.)
0C	READ @ 40180 = 37	*DATA RETURNED = 37 (Checks $\overline{PCS3}$ = 0; all other $\overline{PCS}$ lines = 1.)
0D	READ @ 40200 = 3B	*DATA RETURNED = 3B (Checks $\overline{PCS4}$ = 0; all other $\overline{PCS}$ lines = 1.)
0E	READ @ 40280 = 3D	*DATA RETURNED = 3D (Checks $\overline{PCS5}$ = 0; all other $\overline{PCS}$ lines = 1.)
0F	READ @ 40300 = 3E	*DATA RETURNED = 3E (Checks $\overline{PCS6}$ = 0; all other $\overline{PCS}$ lines = 1.)
10	WRITE CTL = 001	**HOLD = 1; ARDY, SRDY = 0
11	WRITE CTL = 010	**HOLD, ARDY = 0; SRDY = 1
12	WRITE CTL = 100	**HOLD, SRDY = 0; ARDY = 1
13	WRITE @ 0 = 0	**$\overline{RES}$ = 0; $\overline{TEST}$ = 1 (Checks $\overline{WR}$ = 0; $\overline{RD}$ = 1.)

Table 6-3. Enhanced Self Test Failure Codes (cont)

EXECUTION/ FAILURE CODE	OPERATION	EXPECTED RESULT (TROUBLESHOOTING CLUES)
14	READ @ 0 = XX	**$\overline{RES}$ = 1; $\overline{TEST}$ = 0 (Checks $\overline{WR}$ = 1; $\overline{RD}$ = 0.)
15	READ @ C0000 = XX	**INT0 = 0; INT1-3 = 1; NMI, DRQ0-1 = 1 (Checks $\overline{UCS}$ low; $\overline{LCS}$, $\overline{MCS0}$-3, $\overline{PCS0}$ high)
16	READ @ 0 = XX	**INT1 = 0; INT0,2,3 = 1; NMI, DRQ0-1 = 1 (Checks $\overline{LCS}$ low; $\overline{UCS}$, $\overline{MCS0}$-3, $\overline{PCS0}$ high)
17	READ @ 80000 = XX	**INT2 = 0; INT0,1,3 = 1; NMI, DRQ0-1 = 1 (Checks $\overline{MCS0}$ low; $\overline{LCS}$, $\overline{UCS}$, $\overline{MCS1}$-3, $\overline{PCS0}$ high)
18	READ @ 90000 = XX	**INT3 = 0; INT0,1,2 = 1; NMI, DRQ0-1 = 1 (Checks $\overline{MCS1}$ low, $\overline{LCS}$, $\overline{UCS}$, $\overline{MCS0}$,2,3, $\overline{PCS0}$ high)
19	READ @ A0000 = XX	**NMI = 0; INT0-3 = 1; DRQ0-1 = 1 (Checks $\overline{MCS2}$ low; $\overline{LCS}$, $\overline{UCS}$, $\overline{MCS0}$,1,3, $\overline{PCS0}$ high)
1A	READ @ B0000 = XX	**DRQ0 = 0; INT0-3 = 1; NMI, DRQ1 = 1 (Checks $\overline{MCS3}$ low; $\overline{LCS}$, $\overline{MCS0}$-2, $\overline{PCS0}$ high)
1B	READ @ 40000 = XX	**DRQ1 = 0; INT0-3 = 1; NMI, DRQ0 = 1 (Checks $\overline{PCS0}$ low; $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$-3 high)
1C	TRANSPARENT READ	**TMR IN 0 = 0; TMR IN 1 = 1 (Checks TMR OUT 0 low; TMR OUT 1 high)
1D	TRANSPARENT READ	**TMR IN 0 = 1; TMR IN 1 = 0 (Checks TMR OUT 0 high; TMR OUT 1 low)

*A Data Error will be created by the Pod if the data read is not the expected data.

**A Control Error will be created by the Pod if the latched result is not the expected result.

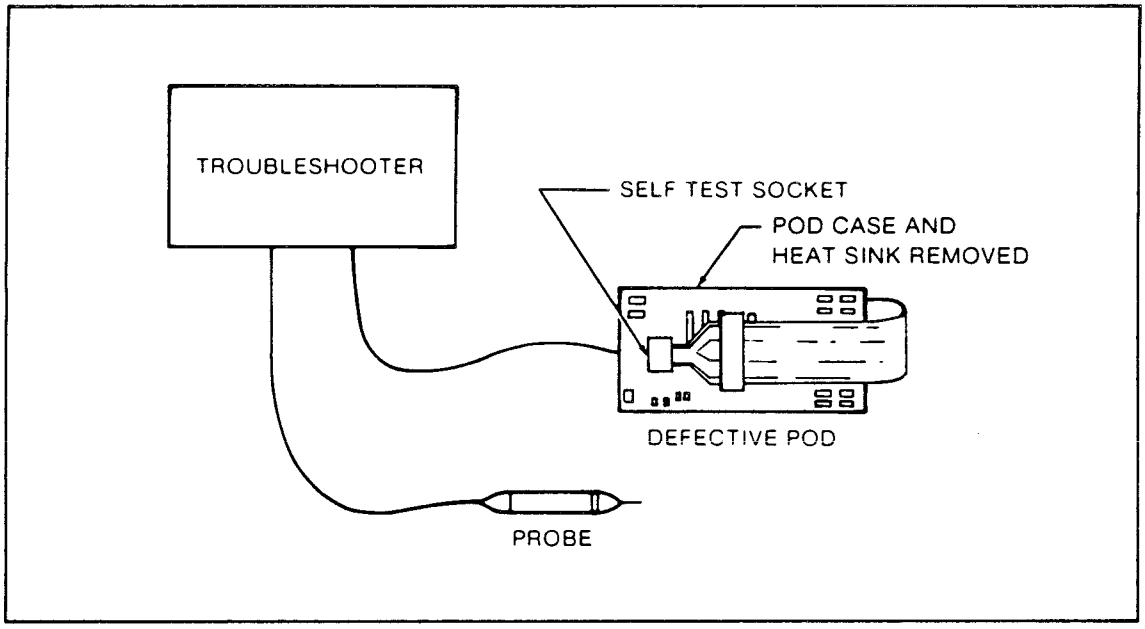


Figure 6-1. Troubleshooting a Defective Pod

5. Press the Setup key on the Troubleshooter and set the following conditions:

SET-TRAP BAD POWER SUPPLY? YES
SET-TRAP ILLEGAL ADDRESS? YES
SET-TRAP ACTIVE INTERRUPT? NO
SET-TRAP ACTIVE FORCE LINE? NO
SET-TRAP CTL ERR? YES
SET-TRAP ADDR ERR? YES
SET-TRAP DATA ERR? YES
SET-ENABLE EXTRDY? NO
SET-ENABLE HOLD? NO

You are now ready to explore the possible causes of Pod failure that are indicated by the Pod self test failure codes.

Troubleshooting a Defective Pod 6-14.

INTRODUCTION 6-15.

When the Pod and the Troubleshooter are connected in this configuration (and with Pod Function Address F0 0000 = 00 to disable the self test), the tests and troubleshooting functions of the Troubleshooter can be applied to the Pod, much like any other UUT. For example, you can perform read or write operations on the UUT (which is actually the self test socket). The Troubleshooter no longer knows that the Pod is plugged into its self-test socket.

NOTE

The default values for all of the Peripheral Control Block registers are suitable when you use the Pod as a UUT. No changes have to be made to the Pod's configuration.

Use the failure code data that was reported as the starting point for subsequent troubleshooting. Use standard troubleshooting techniques to investigate possible sources of the problem.

While investigating the problem, you may find it useful to recreate the standard self test routine that detected the failure. Procedures for recreating the tests are listed below.

RECREATING THE STANDARD SELF TEST 6-16.

Follow the steps listed in Table 6-4 to recreate the individual tests done in the normal self test.

The same basic techniques may be used for the additional failure codes that may be obtained when using the 9020A Micro-System Troubleshooter in the remote mode.

Table 6-4. Recreating the Standard Self Test Routines

TEST ROUTINE/ FAILURE CODE	POD OPERATION	OPERATOR ACTIONS TO RECREATE TEST
00	Reset Pod READ @ 0FF0=0F	WRITE @ F00000=0 READ @ 0FF0 If a power fail error message occurs, check the power-detection circuits.
01	WRITE @ 0FF0=0F	WRITE @ 0FF0=0F
02	Test Control Line	BUS TEST
03	Send command to enable all Pod Enableable lines and verify that a Pod timeout occurs. This timeout is transparent to the operator.	Enable EXTRDY and HOLD

RECREATING THE ENHANCED SELF TEST ROUTINES 6-17.

The enhanced self test consists of several test routines that are performed on the Pod's circuitry. The test routines are listed and described in Table 6-3.

Whenever the Pod fails the enhanced self test, the Pod causes the Troubleshooter to display the message *POD SELF TEST 80188 FAIL 00.* To confirm that the enhanced self test failed, rerun the individual test routine failed, and display information about the failure, use the following procedure:

NOTE

Quick-Looping can be done on the self-test routines.

1. Do a WRITE @ F0 0000 = 0 operation to cancel the self test mode.

2. Perform a read operation at address F0 0000. If there was a previous failure of an enhanced self test routine, this read operation will cause that failed routine to re-execute. (In many cases an error message will be displayed.)

3. Press the MORE key to find out more information about the error that was detected. (Sometimes more than one line of information is available.)

4. Then press the CONT key to find out which enhanced self test routine failed.

You may rerun the failing routine at any time by either repeating the read operation of F0 0000 (to rerun the last failing routine) or by writing a test code to address F0 0000. The procedure for using a Write operation to recreate a specific self test routine is as follows:

1. Do a WRITE @ F0 0000 = 0 operation to cancel the self test mode.

2. A WRITE @ F0 0000=02, for example, will cause the Pod to re-execute enhanced self test routine number 2.

NOTE

If you write bad data to this address, (i.e., try to execute an enhanced selftest routine that doesn't exist), the Pod will report a data drivability error on the non-zero bits of the data.

Assume that the Pod self test is performed and the Troubleshooter displays the message *POD SELF TEST 80188 FAIL 00.* The following sequence of keystrokes provides information about the Pod failure detected:

PRESS	DISPLAY	COMMENT
WRITE @ F00000=0	*WRITE @ F00000=0*	Disable the Self Test.
READ F00000 ENTER	*DATA ERR @ F00000-LOOP?*	Enhanced self test detected data error.
MORE	*DATA BITS 80-LOOP?*	Unexpected data on data bit 7 (hex 0080 corresponds to bit 7).
CONT	*READ @ F0 0000* = 02 FAIL	The failure occurred during test routine 2.

NOTE

If you press the LOOP key or the YES key in response to the LOOP? prompts that accompany the preceding messages, the Troubleshooter loops on the READ @ F0 0000; the Troubleshooter then loops on the test routine in which the failure occurred.

The previous sequence of messages indicates that data bit 7 was found to be in error while the Pod was performing test routine 2. Note that although a data error implies a drivability error, when encountered with the enhanced self test a data error may also indicate that incorrect data was received during a read operation.

TROUBLESHOOTING AN INOPERATIVE POD 6-18.

Introduction 6-19.

This section describes what to do if the Troubleshooter displays any of the three *ATTEMPTING RESET* messages when the Pod self test is performed. The *ATTEMPTING RESET* messages indicate that the Pod is not operating and is not responding to the Troubleshooter.

If you correct a problem while using the procedures provided in this section, try the Pod self test again. If the Troubleshooter again displays an *ATTEMPTING RESET* message, continue with the procedures in this section. However, if the Troubleshooter displays the message *POD SELF-TEST 80188 FAIL xx,* refer to the previous section titled Troubleshooting a Defective Pod. The reason for referring to the other section is that when the Pod is again communicating with the Troubleshooter, you may use the Pod to help troubleshoot itself.

Preparation for Troubleshooting an Inoperative Pod 6-20.

An inoperative Pod is like any other microprocessor-based UUT that is not operating properly; the easiest way to fix an inoperative Pod is by using a Troubleshooter and a good Pod. Preparation instructions also apply to troubleshooting without a good Pod, but note that the detailed troubleshooting steps that follow only apply to using the second Troubleshooter and Pod. Prepare to troubleshoot the inoperative Pod by performing the following steps:

1. Disassemble the Pod, referring to the later section titled Disassembly, but do not separate the two PCB assemblies.

2. Look for any obvious problems, such as burned components or ICs that are loose in their sockets. Replace components if necessary. If such obvious defects are found, it might be prudent to try the self test again at this point.

3. Remove the Pod microprocessor from its socket.

4. If a second Troubleshooter is available, connect the Pod cable plug from the inoperative Pod to the second Troubleshooter to supply the inoperative Pod with power. If a second Troubleshooter is not available, connect +5V (2 amp) power supply to the Pod as shown in Figure 6-2. An easy place to make the power connections is at the connector that normally connects the cable to the Troubleshooter.

CAUTION

Do not operate the Pod with the voltage supply exceeding 5.25 volts, or damage to the Pod could result.

5. Provide a clock signal for the inoperative Pod by inserting the ribbon cable of the inoperative Pod into its own self test socket. (Make sure the clock is working properly.) An alternative source for a clock signal is a known good UUT or frequency generator.

6. Connect the Troubleshooter to the good Pod as shown in Figure 6-2. Apply power to the Troubleshooter, then install the ribbon cable plug of the good Pod into the microprocessor socket of the inoperative Pod.

CAUTION

Do not apply or remove power to the good Pod with the ribbon cable connected between the good Pod and the inoperative Pod.

CAUTION

Do not separate the PCB assemblies of the inoperative Pod with power applied to the inoperative Pod. Failure to comply with this can damage CMOS components in the Pod. The PCB assemblies should be securely fastened together with the proper screws before applying power.

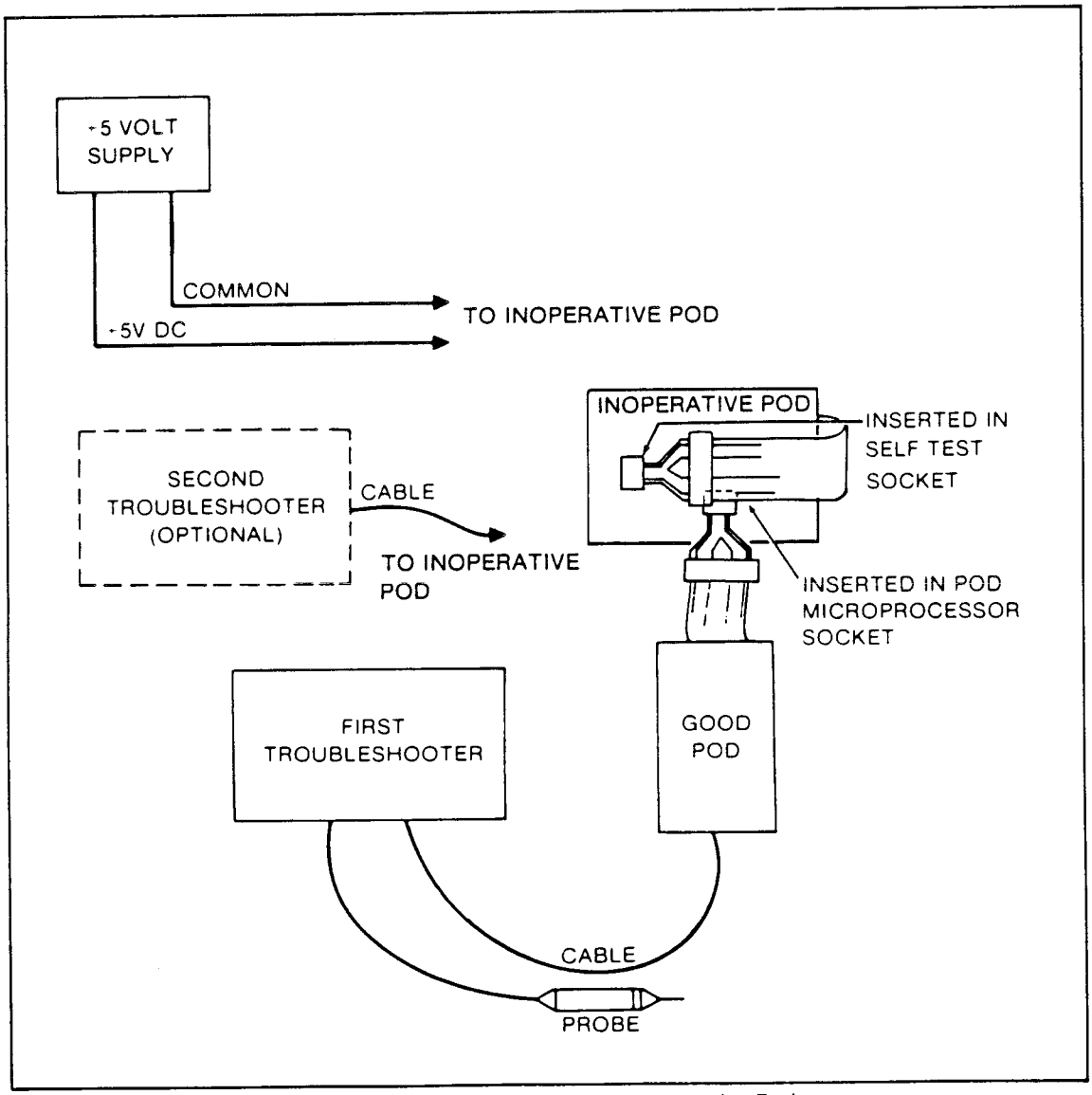


Figure 6-2. Troubleshooting an Inoperative Pod

Procedure for Troubleshooting an Inoperative Pod 6-21.

Use the following steps as a guide for troubleshooting an inoperative Pod using a good Pod. The circuits and components mentioned in these steps appear in the schematic diagrams in Section 8, and the circuits are described in the Theory of Operation in Section 5.

NOTE

The following procedures are intended only to direct the technician towards the source of a problem. In each case, once an anomaly is detected, it is up to the technician to pursue the problem further. It is suggested that standard troubleshooting procedures be used, once a problem is identified, to locate the source of the trouble. This involves such things as checking and verifying activity of enabling and timing signals, input and output signals, and logic.

NOTE

When performing looping read or write operations with a synchronized oscilloscope connected to the Troubleshooter, use the Quick-Looping Read or Write feature described in Section 4 to obtain a brighter signal trace on the scope.

1. Prepare the Pod as outlined in the previous section (Preparation for Troubleshooting an Inoperative Pod). Position the inoperative Pod so that the processor board (the one with the microprocessor socket) is upwards. Apply power to the inoperative Pod. If a second Troubleshooter is used to supply power to the Pod to be tested, press the STOP key on the second Troubleshooter to prevent repetitive Pod resets.
2. Check that the microprocessor's clock signal is present at U53, pin 59. If it is not, trace the clock signal to the fault.
3. Check that $\overline{\text{RES}}$ is not being asserted (low) at U53 pin 24. If $\overline{\text{RES}}$ is asserted (low), tie pin 23 ($\overline{\text{RESET}}$) of the Troubleshooter cable connector to +5V.
4. If the Pod is plugged into a second Troubleshooter, press the BUS TEST key and then press the STOP key. The Bus Test will reset the inoperative Pod. Pressing the STOP key will prevent the Troubleshooter from continuing to reset the Pod. If the second Troubleshooter is not available, use the first Troubleshooter's probe to pulse the inoperative Pod Reset line low.
5. After the inoperative Pod is initialized, try a BUS TEST with the first Troubleshooter. If you get a timeout message, use the Setup functions to disable the enableable lines.
6. If the Bus Test works now, check the input buffers.
7. If a timeout still occurs, check the clock circuit.
8. Do a ROM TEST and a RAM TEST. Refer to Table 6-5 for the ROM and RAM addresses, the expected ROM signature that should be obtained when the ROM Test is performed, and the expected ROM checksum that should be obtained when the Quick ROM Test is performed.

   The Interrupt vector is at FFFFF
   The Signature is at FFFFC and FFFFD (9000 ROM TEST)
   The Checksum is at FFFFA and FFFFB (Quick TEST)

   If the ROM Test and RAM Test are successful, the next thing to check is the data communication with the Troubleshooter.

Table 6-5. Pod Device Addresses

DEVICE	LOWER ADDRESS	UPPER ADDRESS (IF ONE EXISTS)
RAM	0000	0FFF
ROM**	F 8000	F FFFF
Peripheral Control* Block	FF00 2000	FFFE 20FE

*The address range of the internal peripheral control block will be set to either FF00-FFFE in IO space or 2000-20FE (segment = 0) in memory space. It will be moved back and forth between these two address spaces, depending upon where the next UUT access occurs.

**The last three words of the Pod's ROM contain:

FFFFF	Interrupt Information (FF - high byte)
FFFFD	Signature (Troubleshooter ROM TEST - high byte)
FFFFC	Signature (Troubleshooter ROM TEST - low byte)
FFFFB	Checksum (Pod's Quick ROM Test - high byte)
FFFFA	Checksum (Pod's Quick ROM Test - low byte)

When performing a ROM Test, test from F 8000 to F FFFB, then compare the resulting signature to the contents of FFFFC and FFFFD. Perform a Quick ROM test from F8000 to FFFF9, then compare the resulting checksum at 300 000C and 300 000D with the contents of FFFFB and FFFFA.

9. Check the path of incoming data. If a second Troubleshooter is being used and has timed out, the data AB is probably being placed on the second Troubleshooter's data lines. The first Troubleshooter can read the data lines of the second Troubleshooter by performing a read operation at 1E00 (1E00 is the address of incoming data on the Processor PCB). The data should be AB. Perform a read at 1E01—the most-significant bit (bit 7) of the data should be the value of $\overline{\text{MAINSTAT}}$. The second-most-significant bit (bit 6) of the data should be LOWPR (it should be low). The rest of the bits in the data are indeterminate.

   If a second Troubleshooter is not being used, you may check the incoming data path by performing a looping read operation at 1E00. Set the probe stimulus to toggle high and low pulses; then apply the stimulus pulses to the data lines and check whether all bits are readable high and low.

10. If you are using a second Troubleshooter, checking the path of outgoing data is not as straightforward as previously described for incoming data. Do the following procedure to check the path of outgoing data:

    a. Disconnect the inoperative Pod and connect a good Pod to the second Troubleshooter.

    b. Do a complete operation that does not time out, then press the STOP key. This will leave the Troubleshooter data lines in a tristate condition.

c. Without disconnecting power, disconnect the good Pod from the second Troubleshooter and reconnect the inoperative Pod. Do not press any keys on the second Troubleshooter.

d. Using the first Troubleshooter probe in the probe stimulus mode, pulse the Reset line on the inoperative Pod to initialize the inoperative Pod.

e. On the first Troubleshooter, perform the operation WRITE @ 1D01 to set Bit 6 high; this should turn on the data output latch.

NOTE

Table 6-6 contains a description of this and other latches, including addressing and timing. Table 6-7 contains bit definitions of selected Pod addresses.

f. Write a value to 1F00 and check the data lines with the probe to see if the expected value is present.

g. Check the output port to make sure all signals are writable.

h. Make sure a Pod Reset resets all output port lines low.

i. Try writing to the timer. Use the probe in the free-run mode. If RUNREQ is low, the UUTON should have a short pulse. If RUNREQ is high, RUNUUT should become active after the timer fires, and stay on until the Pod is reset. This is the RUN UUT mode. While in RUN UUT mode, you should not be able to read the ROM or RAM of the inoperative Pod, but instead you should be communicating with the inoperative Pod UUT or the self test socket.

j. Check that the signal $\overline{\text{INTNMI}}$ is not asserted (check that it is high). If $\overline{\text{INTNMI}}$ is asserted (low), the Pod's microprocessor is disabled from writing to devices in the Pod. $\overline{\text{INTNMI}}$ is an output of U30, pin 8. This flip-flop is cleared by power-up and Troubleshooter resets.

If all these tests are good, you should no longer have an inoperative Pod, but you may still have a defective Pod. If the Pod is still defective, refer to the previous section titled Troubleshooting a Defective Pod.

EXTENDED TROUBLESHOOTING PROCEDURES 6-22.

Introduction 6-23.

The troubleshooting procedures provided in this section supplement the circuit checks performed on the Pod during the Pod self test. These procedures are appropriate for use with a Pod that passes the Pod self test but does not appear to function normally when used with a Trubleshooter and a good UUT. If a Pod fails self test, it would be better to begin troubleshooting with the procedure provided in the section titled Troubleshooting a Defective Pod.

Potential problems that may exist in a Pod that passes the Pod self test may be divided into three categories:

- Misconfigured Pod
- Partially checked circuits
- Timing and noise problems

Misconfigured Pod 6-24.

The Pod may exhibit abnormal behavior if it is not configured properly for the UUT being tested. Recheck the procedures listed in Section 2 to make sure that the Pod has its Chip Selects, Interrupt Control registers, DMA Control registers, Timer Control registers, RESET output signal, and Transparent Read address set correctly. Check to see that the Pod is not accidently being placed in the Queue Status mode. A Pod that has inadvertently been changed to the Queue Status mode will show symptoms such as unexplained drivability errors on the address lines.

Partially Checked Circuits 6-25.

The most obvious partially checked circuits, and the easiest to check, are possible open lines on the UUT cable. The circuits are usually checked through the hybrid protection circuits and back through the latches, but the cable lines themselves are not checked. Refer to Table 6-8 for a list of the partially checked lines. These lines can be checked with an ohmmeter, but if they are intermittent, a better way is to check them with the Troubleshooter probe or an oscilloscope while the Pod is exhibiting its symptoms. Check at the Pod end for the status lines (inputs), and check at the UUT end for control lines (outputs).

Table 6-6. Pod Latch Addresses and Timing

LATCH	ADDRESS	TIMING*	LATCH	ADDRESS	TIMING*
DATAOUT (low byte)	1F00	N/A	TRANSADR (high byte)	1601	Data
DATAIN (low byte)	1E00	N/A	CTLOUTB	1500	N/A
DATAIN (high byte)	1E01	N/A	LLATCHC (low byte)	1400	Late
CTLOUTA (low byte)	1D00	N/A	LLATCHC (high byte)	1401	Late
CTLOUTA (high byte)	1D01	N/A	LLATCHB (low byte)	1300	Late
TIMER (low byte)	1C00	N/A	LLATCHB (high byte)	1301	Late
DATLATCH (low byte)	1900	Data	LLATCHA (low byte)	1200	Late
**DATLATCH (high byte)	1901	Data	LLATCHA (high byte)	1201	Late
ADRLATCH (low byte)	1800	Address	EELATCH (low byte)	1100	Extra Early
ADRLATCH (high byte)	1801	Address	ELLATCH (low byte)	1000	Address
TRANSCS (low byte)	1700	Late	ELLATCH (high byte)	1001	Late
TRANSCS (high byte)	1701	Late			
TRANSADR (low byte)	1600	Data			

*Definition of Latch Timing:
Address = Beginning of T2
Data = Beginning of T4
Late = Middle of T3
Extra Early = Beginning of T1

**The latch at 1901 contains A8-A15, but it is latched during the data period.

Table 6-7. Bit Definitions of Selected Pod Addresses

ADDRESS*	DATA BITS 7	6	5	4	3	2	1	0
1F00	Pod7	Pod6	Pod5	Pod4	Pod3	Pod2	Pod1	Pod0
1E01	$\overline{\text{MAINSTAT}}$	LOPWR	—	—	—	—	—	—
1E00	Pod7	Pod6	Pod5	Pod4	Pod3	Pod2	Pod1	Pod0
1D01	PODSTAT	9KOUT	SYNC1	SYNC0	SLFSEL1	SLFSEL0	ENHOLD	ENERDY
1D00	ENTCS	WTCTL	MINT	FQS1	FQS0	FHLDA	FRESET	FLOCK
1500	—	—	SPECRU	FTMRIN	FINT	COUNTON	FS6	ENRESET
1C00	Run Request	—	—	—	← Timer Data →			
1901	D7 ← Data →							D0
1900	A15 ← Address →							A7
1801	A15 ← Address →							A7
1800	A8 ← Address →							A0
1701	—	—	—	$\overline{\text{PCS6}}$/A2	$\overline{\text{PCS5}}$/A1	$\overline{\text{PCS4}}$	$\overline{\text{PCS3}}$	$\overline{\text{PCS2}}$
1700	$\overline{\text{PCS1}}$	$\overline{\text{PCS0}}$	$\overline{\text{MCS3}}$	$\overline{\text{MCS2}}$	$\overline{\text{MCS1}}$	$\overline{\text{MCS0}}$	$\overline{\text{LCS}}$	$\overline{\text{UCS}}$
1601	A15 ← Address →							A7
1600	A8 ← Address →							A0
1401	$\overline{\text{TEST}}$	S7	S6	NMI	INIT3/$\overline{\text{INTA1}}$	INT2/$\overline{\text{INTA0}}$	INT1	INT0
1400	DT/$\overline{\text{R}}$	$\overline{\text{DEN}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$/QS1	ALE/QS0	HOLD	SRDY	ARDY
1301	TMR OUT 0	TMR OUT 1	—	$\overline{\text{PCS6}}$/A2	$\overline{\text{PCS5}}$/A1	$\overline{\text{PCS4}}$	$\overline{\text{PCS3}}$	$\overline{\text{PCS2}}$
1300	$\overline{\text{PCS1}}$	$\overline{\text{PCS0}}$	$\overline{\text{MCS3}}$	$\overline{\text{MCS2}}$	$\overline{\text{MCS1}}$	$\overline{\text{MCS0}}$	$\overline{\text{LCS}}$	$\overline{\text{UCS}}$
1201	—	—	—	$\overline{\text{PCS6}}$/A2	$\overline{\text{PCS5}}$/A1	$\overline{\text{PCS4}}$	$\overline{\text{PCS3}}$	$\overline{\text{PCS2}}$
1200	$\overline{\text{PCS1}}$	$\overline{\text{PCS0}}$	$\overline{\text{MCS3}}$	$\overline{\text{MCS2}}$	$\overline{\text{MCS1}}$	$\overline{\text{MCS0}}$	$\overline{\text{LCS}}$	$\overline{\text{UCS}}$
1100	$\overline{\text{SELF TEST}}$	$\overline{\text{DEN}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$/QS1	$\overline{\text{ALE}}$/QS0	$\overline{\text{LENGND}}$	$\overline{\text{POWER FAIL2}}$	$\overline{\text{POWER FAIL1}}$
1001	A19	A18	A17	A16	—	$\overline{\text{S2}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$
1000	TMR IN 1	TMR IN 0	DRQ1	DRQ0	$\overline{\text{RES}}$	HLDA	RESET	$\overline{\text{LOCK}}$

Table 6-8. Pod Ribbon Cable Lines Partially Checked in the Pod Self Test

SIGNAL	UNTESTED CONDITIONS
CLKOUT	Drivability Signal Presence
$\overline{S1}$, DT/$\overline{R}$, $\overline{DEN}$	Open in the Pod end of the cable and a short to ground at the UUT end of the cable
$\overline{S2}$	Low condition
Gnd (pin 60)	Open
Vcc	Shorts to the Pod's +5V supply

During the Pod self test, the Pod self test socket forces the Pod into the normal mode, so some of the Queue Status mode circuits are only partially checked. These include any signal that is different in the Queue Status mode from the Normal mode.

The circuits that generate the transparent reads to the UUT when the Pod is not in a UUTON cycle are also partially checked. These circuits could be malfunctioning between cycles and causing problems with the UUT.

Another circuit that is partially checked is the tristate logic. If the UUT uses DMA (HOLD/HOLDAcknowledge) cycles, the circuitry of the Pod that handles these functions could be the problem.

Timing and Noise Problems 6-26.

Timing or noise problems are usually caused by components that are still functioning, but are not functioning within the component specifications. The best way to check this problem is to look at suspected signals using an oscilloscope synchronized to either address or data. Look for slow rise times or signals driven to marginal levels. If the part is too slow, it might fail on the UUT, but pass the Pod self test because of narrower design margins on the UUT.

If a part has marginal drive capabilities, the added noise of a UUT environment might cause it to fail. Be sure to note that inputs can malfunction (they may leak perhaps) and put too much load on an output causing either low levels, slow rise times, or both.

DISASSEMBLY 6-27.

To gain access to the two PCB assemblies in the Pod, perform the following steps:

1. Remove the Pod ribbon cable plug from the self test socket.

2. Turn the Pod over on its top (with the large Pod decal facing up). Remove the four phillips screws that hold the case halves together and remove the top and bottom case halves. Place the PCB assemblies so that the self test socket (on the processor PCB assembly) is facing up.

3. Remove the four phillips screws that connect the heat sink to the processor PCB assembly and remove the heat sink.

NOTE

The heat sink is not needed for heat dissipation unless the Pod is fully assembled. If the two PCB assemblies are removed from the top and bottom cases, the heat sink may be removed during Pod operation and troubleshooting.

4. Remove the single phillips screw that retains the shield surrounding the PCB assemblies. Remove the shield.

NOTE

When the shield and the heat sink are removed, all the components are exposed. It may not be necessary to separate the two PCB assemblies while troubleshooting except to replace components.

5. To separate the two PCB assemblies, turn the PCB assemblies over so that the self test socket is facing down. Remove the four phillips screws at the corners of the PCB assemblies and carefully pull the boards apart at the two connectors along the sides.

Section 7
List of Replaceable Parts

INTRODUCTION 7-1.

This section contains an illustrated parts list for the instrument. Components are listed alphanumerically by assembly.

Parts lists include the following information:

1. Reference Designation.
2. Static Warning marker.

CAUTION

Devices indicated by an asterisk (*) in the listing are subject to damage by static discharge.

Devices indicated by an asterisk (*) in the listing are subject to damage by static discharge.

3. Description of Each Part.
4. Fluke Stock Number.
5. Federal Supply Code for Manufacturers (see the 9000 Series Troubleshooter Service Manual for Code-to-Name list).
6. Manufacturer's Part Number.
7. Total Quantity of Components Per Assembly.
8. Recommended spares quantity. This entry indicates the recommended number of spare parts necessary to support one to five instruments for a period of two years. This list presumes an availability of common electronic parts at the maintenance site. For maintenance for one year or more at an isolated site, it is recommended that at least one of each assembly in the instrument be stocked.

HOW TO OBTAIN PARTS 7-2.

Components may be ordered directly from the manufacturer by using the manufacturer's part number, or from the John Fluke Mfg. Co., Inc. or an authorized representative by using the Fluke Stock Number.

In the event the part ordered has been replaced by a new or improved part, the replacement will be accompanied by an explanatory note, and installation instructions if necessary.

To ensure prompt and efficient handling of your order, include the following information.

1. Quantity.
2. Fluke Stock Number.
3. Description.
4. Reference Designation.
5. Printed Circuit Board Part Number and Revision Letter.
6. Instrument Model and Serial Number.

A Recommended Spare Parts Kit for your basic instrument is available from the factory. This kit contains those items listed in the RSQ column of the parts lists in the quantities recommended.

Parts price information is available from the John Fluke Mfg. Co., Inc. or its representative. Prices are also available in a Fluke Replacement Parts Catalog, which is available upon request.

MANUAL CHANGE AND BACKDATING INFORMATION 7-3.

Table 7-4 contains information necessary to backdate the manual to conform with earlier PCB configurations. To identify the configuration of the PCB's used in your instrument, refer to the revision letter on the component side of each PCB assembly.

As changes and improvements are made to the instrument, they are identified by incrementing the revision letter marked on the affected PCB assembly. These changes are documented on a supplemental change/errata sheet which, when applicable, is inserted at the front of the manual.

To backdate this manual to conform with an earlier assembly revision level, perform the changes indicated in Table 7-4. There are no backdating changes at this printing. All PCB assemblies are documented at their original revision level.

TABLE 7-1. 9000A-80188 FINAL ASSEMBLY
(SEE FIGURE 7-1.)

REFERENCE DESIGNATOR A->NUMERICS--->	S	----------DESCRIPTION----------	FLUKE STOCK --NO--	MFRS SPLY CODE-	MANUFACTURERS PART NUMBER --OR GENERIC TYPE--	TOT QTY	R S -Q	NOT -E
A 41	*	INTERFACE PCB ASSEMBLY	715979	89536	715979	1		2
A 42	*	PROCESSOR PCB ASSEMBLY	716001	89536	716001	1		1
H 1		SCREW,MACH,PHP SEMS,STL,4-40X1/4	185918	89536	185918	3		
H 2		SCREW,MACH,PHP,STL,4-40X3/4	115063	89536	115063	4		
H 3		SCREW,MACH,PHP,S.STL,4-40X3/8	256164	89536	256164	4		
H 4		SCREW,MACH,PHP,STL,4-40X5/8	145813	89536	145813	1		
H 5		WASHER,LOCK,INTRNL,STEEL,#4	110403	89536	110403	1		
MP 1		SHELL TOP	744797	89536	744797	1		
MP 2		SHELL, BOTTOM	648881	89536	648881	1		
MP 3		LABEL, STATIC CAUTION	605808	89536	605808	1		
MP 4		DECAL, POD	737973	89536	737973	1		
MP 5		DECAL, SPEC	737981	89536	737981	1		
MP 6		WARNING DECAL 8086	659805	89536	659805	1		
MP 7		SPACER,HEX,ALUM,4-40X0.375	187575	89536	187575	1		
MP 8		SHIELD, ALUM MYLAR	749978	89536	749978	1		
MP 9		HEAT DIS.CHIP CARRIER LID,ALUM	745851	89536	745851	1		
MP 10		HEATSINK	744813	89536	744813	1	1	
TM 1		INSTRUCTION MANUAL	738005	89536	738005	1		
TM 2		REFERENCE CARD	758458	89536	758458	1		
TM 3		GETTING STARTED	776005	89536	776005	1		
U 2	*	IC,LSTTL,8-BIT BINARY CNTR W/REG-OUT	741173	89536	741173	1		
U 5	*	PROGRAMED 27256 V1.0	745380	89536	745380	1	1	
U 9, 10	*	IC, 2K X 8 STAT RAM	647222	51157	HM6116P-3	2	1	
U 16	*	PROGRAMED 16L8 V1.0	745356	89536	745356	1	1	
U 23	*	PROGRAMED 16L8 V1.0	745364	89536	745364	1	1	
U 53	*	IC,NMOS,16-BIT MICROPROCESSOR	741348	89536	741348	1		
W 1	*	ASSY, UUT CABLE	744763	89536	744763	1	1	
W 2		CABLE, POD 8086/88	607184	89536	607184	1		
XU 5		CLIP, HEATSINK, 24 PIN	607655	89536	607655	1		
XU 16, 23		CLIP, HEATSINK, 20 PIN	607671	89536	607671	2		
XU 53		SOCKET,IC,CHIP CARRIER,68 PIN,.155	758235	89536	758235	1		
Z 6		HEADER, PROGRAMMABLE	659839	89536	659839	1		

NOTE 1 = PART OF PROCESSOR ASSEMBLY
NOTE 2 = PART OF INTERFACE ASSEMBLY

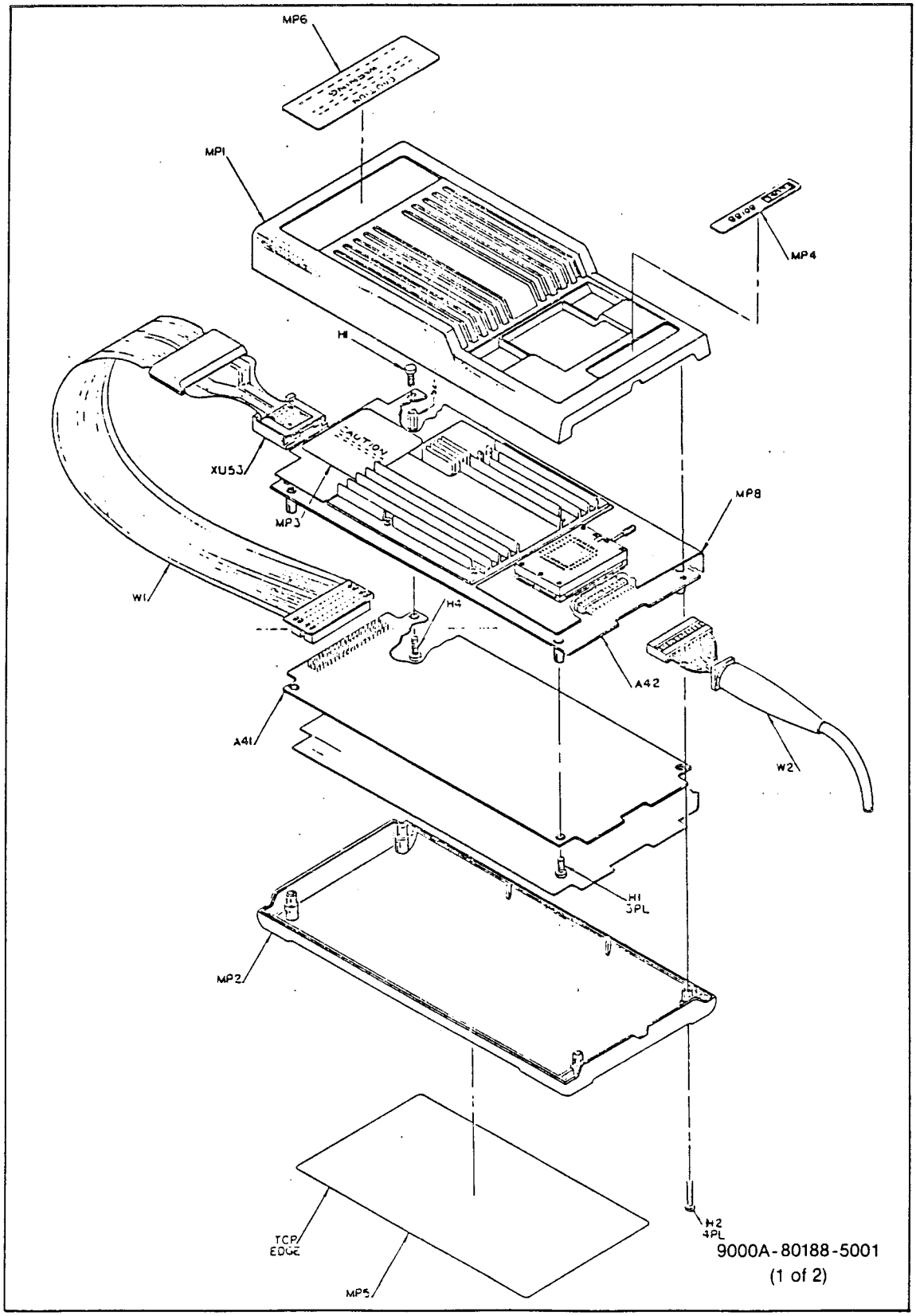


Figure 7-1. 9000A-80188 Final Assembly

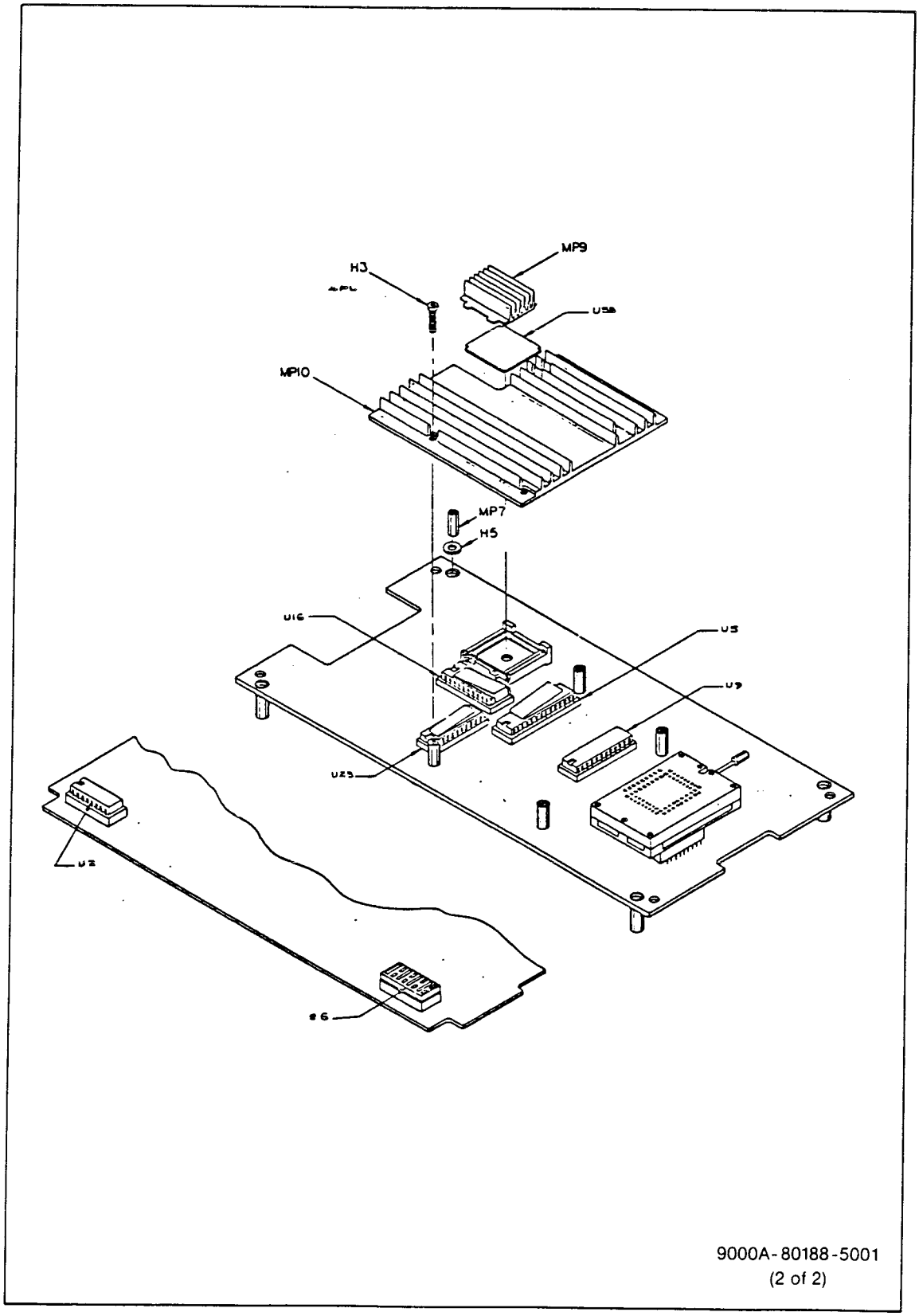


Figure 7-1. 9000A-80188 Final Assembly (cont)

TABLE 7-2. A40 PROCESSOR PCB ASSEMBLY
(SEE FIGURE 7-2.)

REFERENCE DESIGNATOR A->NUMERICS--->	S	----------DESCRIPTION----------	FLUKE STOCK --NO--	MFRS SPLY CODE-	MANUFACTURERS PART NUMBER --OR GENERIC TYPE--	TOT QTY	R S -Q	N O T -E
C 1		CAP,TA,10UF,+-20%,15V	740472	89536	740472	1	1	
C 2, 3		CAP,CER,22PF,+-10%,50V,COG	740563	89536	740563	2		
C 4		CAP,CER,100PF,+-10%,50V,COG	740571	89536	740571	1		
C 5- 10, 12-		CAP,CER,0.22UF,+80-20%,50V,Y5V,1206	740597	89536	740597	17		
C 19, 21- 23			740597					
CR 1, 2	*	DIODE,SI,BV= 75.0V,IO=150MA,500 MW	203323	07910	1N4448	2	1	
J 1		HEADER,2 ROW,0.100CTR,RT ANG,26 PIN	512590	89536	512590	1		
J 2		BLOCK, SPACER 68 POS	773242	89536	773242	1		
MP 1		SPACER,SWAGED,RND,BRASS,4-40X0.340	380329	89536	380329	4		
MP 2		SPACER,SWAGED,RND,BRASS,6-32X0.875	266486	89536	266486	1		
MP 3		SPACER,SWAGED,RND,BRASS,4-40X0.437	442913	89536	442913	4		
P 1, 2		PIN,SINGLE,PWB,0.025 SQ	267500	00779	87022-1	120		
R 1, 3, 14		RES,CHIP,CERM,27K,+-5%,0.125W	740530	89536	740530	3		
R 2, 5, 9,		RES,CHIP,CERM,4.7K,+-5%,0.125W	740522	89536	740522	10		
R 10, 11, 16-			740522					
R 20			740522					
R 4		RES,CHIP,CERM,100K,+-5%,0.125W	740548	89536	740548	1		
R 7, 12		RES,CHIP,CERM,82,+-5%,0.125W	740480	89536	740480	2		
R 15, 21, 22,		RES, CHIP, CERMET, 220,+-5%, 0.125W	746347	89536	746347	4		
R 23			746347					
TP 1- 4		TERM,FASTON,TAB,SOLDR,0.110 WIDE	512889	02660	62395	4		
U 1	*	IC,ALSTTL,OCTAL D F/F,+EDG TRG	740910	89536	740910	1	1	
U 2, 7, 50,	*	IC,LSTTL,OCTAL D TRANSPARENT LATCHES	504514	01295	SN74LS373N	5	1	
U 51, 52	*		504514					
U 3, 8	*	IC, CMOS, OCTAL BUS TRANSCEIVER	535906	36665	MD74C245AC	2	1	
U 6, 31	*	IC,LSTTL,OCTAL BUFFER/LINE DRIVER	634105	04713	SN74LS541N	2	1	
U 11, 13	*	IC,LSTTL,3-8 LINE DCDR W/ENABLE,SOIC	740969	89536	740969	2	1	
U 12	*	IC,LSTTL,2-4 LINE DEMUX,SOIC	740951	89536	740951	1	1	
U 14, 17	*	IC,ALSTTL,QUAD 2-INPUT MULTIPLEXER	740902	89536	740902	2	1	
U 15	*	IC,LSTTL,HEX D F/F,+EDG TRG,SOIC	740944	89536	740944	1	1	
U 18, 22	*	IC,LSTTL,OCTAL D F/F,+EDG TRG,SOIC	740928	89536	740928	2	1	
U 19, 24	*	IC,ASTTL,DUAL 4-INPUT MUX W/3 STATE	740894	89536	740894	2	1	
U 20	*	IC,LSTTL, DIVIDE BY 16 BIN CNTR,SOIC	740936	89536	740936	1	1	
U 21	*	IC,FTTL,8 LINE MUX W/SELECT	697763	89536	697763	1	1	
U 25, 26, 30,	*	IC,LSTTL,DUAL D F/F,+EDG TRG,SOIC	740985	89536	740985	4	1	
U 54	*		740985					
U 27, 28, 29,	*	IC,FTTL,DUAL D F/F,+EDG TRG,W/CL&SET	659508	07263	74F74PC	4	1	
U 58	*		659508					
U 32	*	IC,CMOS,QUAD 2-INPUT NAND GATE	741280	89536	741280	1	1	
U 33, 35, 39,	*	IC,LSTTL,QUAD 2 INPUT OR GATE,SOIC	740878	89536	740878	5	1	
U 43, 57	*		740878					
U 34, 46, 55,	*	IC,LSTTL,QUAD 2 INPUT NAND GATE,SOIC	741033	89536	741033	4		
U 56	*		741033					
U 36, 38, 40	*	IC,LSTTL,QUAD 2 INPUT AND GATE,SOIC	740860	89536	740860	3	1	
U 37	*	IC,LSTTL,QUAD 2 INPUT NOR GATE,SOIC	741025	89536	741025	1	1	
U 41, 47	*	IC,LSTTL,HEX INVERTER,SOIC	741017	89536	741017	2	1	
U 42	*	IC,CMOS,QUAD 2INPUT XOR GATE	740845	89536	740845	1	1	
U 44	*	IC,ALSTTL,TRIPLE 3 INPUT NAND GATE	740886	89536	740886	1		
U 45	*	IC,LSTTL,TRIPLE 3 INPUT NOR GATE,SOIC	740993	89536	740993	1	1	
U 48	*	IC,LSTTL,TRIPLE 3 INPUT AND GATE,SOIC	741009	89536	741009	1	1	
U 49	*	IC,LSTTL,QUAD BUS,SOIC	740977	89536	740977	1	1	
XJ 2		SOCKET RECEPTICAL ASSEMBLY	775981	89536	775981	1	1	
XU 5		SOCKET,IC,28 PIN	448217	91506	328-AG39D	1		
XU 9, 10		SOCKET,IC,24 PIN	376236	91506	324-AG39D	2		
XU 16, 23		SOCKET,IC,20 PIN	454421	09922	DILB20P-108	2		
XU 53		SOCKET,IC,CHIP CARRIER,68 PIN,.100	720888	89536	720888	1		
Y 1	*	CRYSTAL,16MHZ,+-0.005%,HC-18U	721639	89536	721639	1	1	

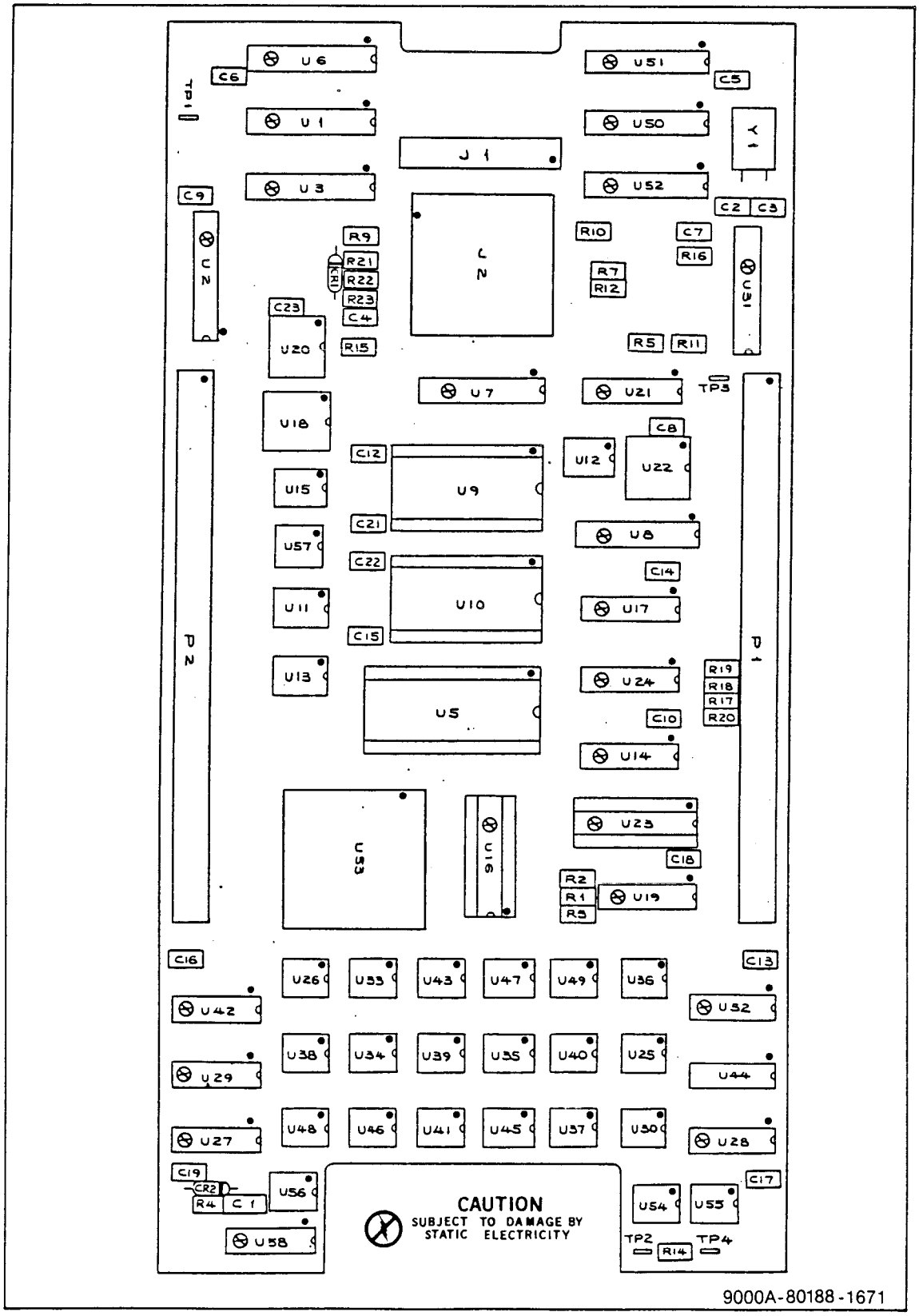


Figure 7-2. A42 Processor PCB Assembly

TABLE 7-3. A41 INTERFACE PCB ASSEMBLY
(SEE FIGURE 7-3.)

REFERENCE DESIGNATOR A-)NUMERICS---)	S	----------DESCRIPTION----------	FLUKE STOCK --NO--	MFRS SPLY CODE-	MANUFACTURERS PART NUMBER --OR GENERIC TYPE--	TOT QTY	R S -Q	NOT -E
A 1- 8	*	HYBRID, 700(, TESTED	582189	89536	582189	8		
C 1- 17		CAP,CER,0.22UF,+-20%,50V,Z5U	309849	71590	CW3C0C224K	17		
#J 1, 2		SOCKET,2 ROW,PWB,0.150CTR,60 POS	602813	00779	86396-6	2		
J 3, 4		HEADER,2 ROW,0.100 CTR,40 PIN	603670	89536	603670	2		
R 1, 11		RES,MF,12.1K,+-1%,0.125W,100PPM	234997	91637	CMF551212F	2		
R 2, 10		RES,MF,22.6K,+-1%,0.125W,100PPM	288431	91637	CMF552262F	2		
R 3		RES,MF,3.01K,+-1%,0.125W,100PPM	312645	91637	CMF553011F	1		
R 4		RES,MF,340,+-1%,0.125W,50PPM	229401	89536	229401	1		
R 5- 7, 9,		RES,CF,4.7M,+-5%,0.25W	543355	80031	CR251-4-5P4M7	5		
R 12			543355					
R 8		RES,CF,1M,+-5%,0.25W	348987	80031	CR251-4-5P1M	1		
R 13		RES,CF,10K,+-5%,0.25W	348839	80031	CR251-4-5P10K	1		
TP 1, 2		TERM,FASTON,TAB,SOLDR,0.110 WIDE	512889	02660	62395	2		
U 1, 3, 4,	*	IC,CMOS,OCTAL D F/F,+EDG TRG,3-STATE	707695	89536	707695	13	1	
U 7, 8, 9,	*		707695					
U 12, 13, 16,	*		707695					
U 19, 20, 21,	*		707695					
U 24	*		707695					
U 6, 18	*	IC,ALSTTL,OCTAL BUS XCVR W/3-STATE	647214	01295	SN74ALS245N	2	1	
U 10	*	IC,LSTTL,OCTAL BUFFER/LINE DRIVER	634105	04713	SN74LS541N	1	3	
U 11, 17, 23	*	IC,ALSTTL,OCTAL D F/F,+EDG TRG	740910	89536	740910	3	1	
U 14, 28, 29	*	IC,ALSTTL,QUAD 2-INPUT MULTIPLEXER	740902	89536	740902	3	1	
U 15, 27	*	IC,ALSTTL,OCTL LINE DRVR W/3-STATE	741165	89536	741165	2		
U 22	*	IC,LSTTL,HEX BUFFER W/3-STATE OUTPUT	536458	01295	SN74LS365N	1	1	
U 25	*	IC,COMPARATOR,QUAD,14 PIN DIP	387233	12040	LM339N	1	1	
U 26	*	IC,COMPARATOR,DUAL,LO-PWR,8 PIN DIP	478354	12040	LM393N	1	1	
VR 1	*	IC, 1.22V,100 PPM T.C.,BANDGAP REF	452771	89536	452771	1	1	
XU 1, 3, 4,		SOCKET,IC,20 PIN	454421	09922	DILB20P-108	13		
XU 7- 9, 12,			454421					
XU 13, 16, 19-			454421					
XU 21, 24			454421					
XU 2, 5		SOCKET,IC,16 PIN	276535	91506	316-AG39D	3		
XU 6			276535					
XVR 1		SPACER,MOUNT,NYLON,	175125	89536	175125	1		
Z 1, 2		RES,NET,SIP,10 PIN,9 RES,4.7K,+-2%	484063	80031	95081002CL	2		
Z 3	*	RES NET THICK FILM ASSY,TESTED-9000	583476	89536	583476	1	1	
Z 4		RES,NET,SIP,10 PIN,5 RES,10K,+-2%	529990	89536	529990	1		
Z 5		RES,NET,SIP,6 PIN,5 RES,1.5K,+-2%	414011	89536	414011	1		

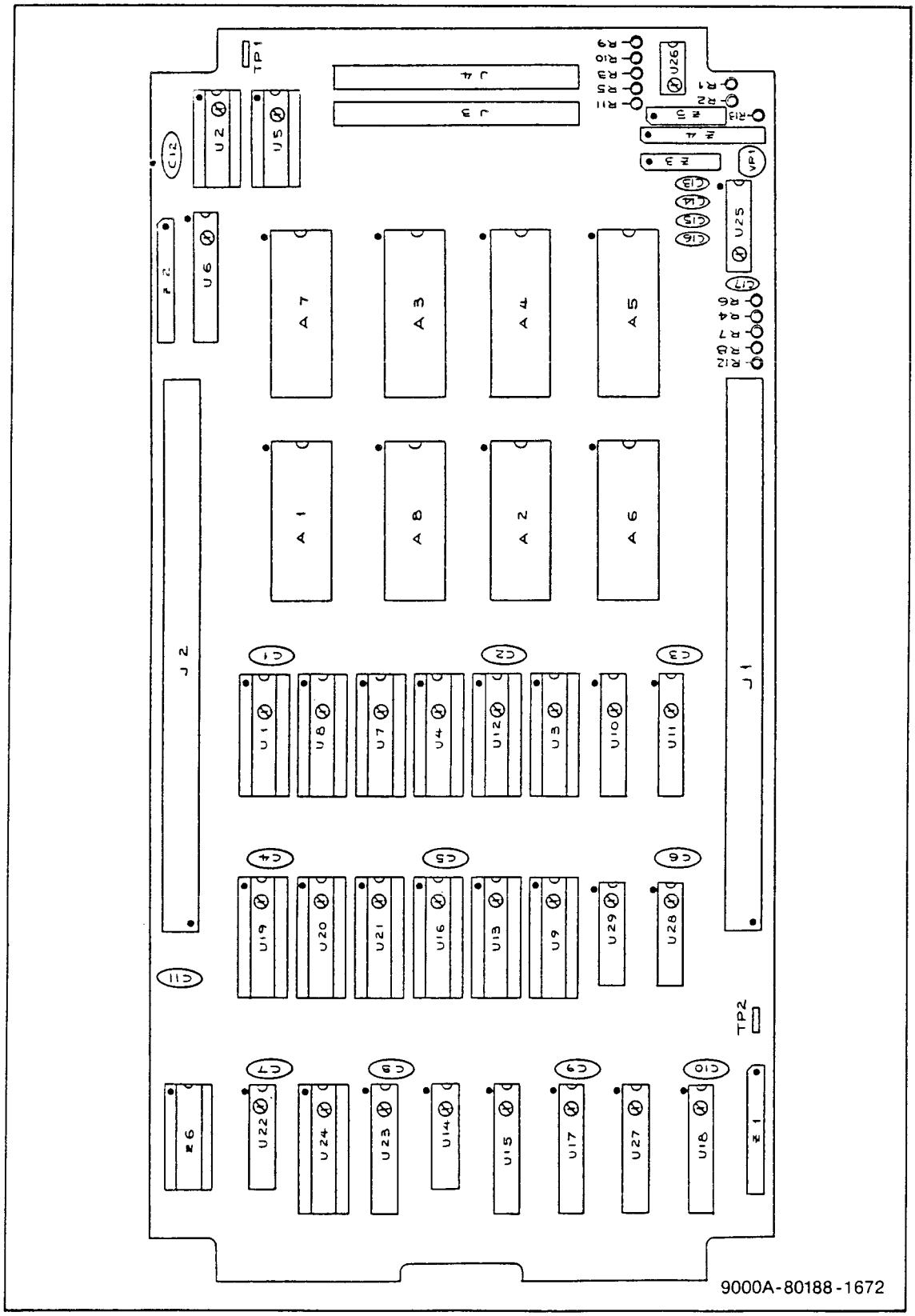


Figure 7-3. A41 Interface PCB Assembly

Table 7-4. Manual Status and Backdating Information

REF OR OPTION NO.	ASSEMBLY NAME	FLUKE PART NO.	*To adapt manual to earlier rev configurations perform changes in decending order (by no.), ending with change under desired rev letter																						
			—	A	B	C	D	E	F	G	H	J	K	L	M	N	P								
A41	Interface PCB	715979		X																					
A42	Processor PCB	716001		●	X																				

* X = The PCB revision levels documented in this manual.
● = These revision letters were never used in the instrument.
— = No revision letter on the PCB.
· = Change did not affect manual.

Appendix A
Compiled Programs For the 80188 Pod

USING THE 9010A LANGUAGE COMPILER PROGRAM A-1.

Introduction A-2.

The 9010A Language Compiler is a microcomputer program that creates test programs for the Troubleshooter. It creates these test programs from source files that are created and edited on the microcomputer. It is available for several common microcomputers, including the Fluke 1720A and 1722A Instrument Controllers, computers with the CP/M operating system, and the IBM Personal Computer. Contact a Fluke Sale Office for information about the 9010A Language Compiler.

Before using the compiler to create 9010A programs for use with the 80188 Pod, you will need to create a new Pod data file. The simple procedure to create the Pod data file is listed below.

Creating a New Pod Data File A-3.

A Pod data file is a simple ASCII file that you create using the text editor on your host computer system. The procedure is a follows:

1. Using the editor, create a new file named 80188.POD.
2. Copy the following lines into the file.

```
!
! 80188 Pod data file
!
FORCELN extrdy = 1
FORCELN hold = 1
busadr = 0000
uutadr = FFF0
```

3. Save this new file as file 80188.POD on the disk.

Using the Pod Data File A-4.

The new Pod data file can now be used with the compiler as described in the 9010A Language Compiler manual.

Verifying the Pod Data File A-5.

The VERIFY program (which is supplied on the 9LC disk with the 9010A Language Compiler) verifies the integrity of files on the disk. It is used to detect files which have

been corrupted, which it does by calculating a checksum for each file and comparing that checksum to the one contained in the VERIFY.DAT file (also on the 9LC disk).

If you would like to add your new Pod data file to the list of files that are checked by the VERIFY program, do the following steps:

1. Edit file VERIFY.DAT (supplied on the 9LC disk) and add the following line to the end of the file:

   80188.POD DDDD

   80188.POD is the name of the new Pod data file and DDDD is a dummy checksum for the file. (You'll replace the dummy checksum with a real one later.)

2. Save the modified VERIFY.DAT file on the disk.

3. Run the VERIFY program. The last two messages that it reports should be:

   File 80188.POD error - signature is CCCC, should be DDDD

   zz files tested - 1 bad signatures, 0 missing files

   80188.POD is the name of the new Pod data file, CCCC is the correct checksum for the Pod data file, and zz is the number of files tested.

4. Write down the correct checksum for the Pod data file (CCCC).

5. Re-edit the file VERIFY.DAT and replace the dummy checksum that you entered before (DDDD) with the correct checksum (CCCC).

6. Run the VERIFY program again to confirm that all changes have been made satisfactorily. The last two messages that it reports should now be:

   File 80188.POD verified

   zz files tested — no errors

A USEFUL QUICK TEST PROGRAM A-6.

The compiler source program listed in Figure A-1 will make the use of the 80188 Pod's Quick Functions appear to operate like the tests that are built into the Troubleshooter. This program is provided on side B of the PCB P: covery tape provided with the 80188 Pod. When using this program, you can use the Quick Functions by entering parameters in response to display prompts, just like the normal built-in tests, rather than by writing information to several special addresses.

NOTE

The program is also shown in standard form in Figure A-2. That program may be used on any Troubleshooter without the compiler. It is entered line-by-line as shown, then saved on magnetic tape. Refer to the 9010A Programmers Manual if you need help.

Once the program is available on tape, regardless of which form it originated from, it needs to be loaded into the Troubleshooter and read into memory. Consult the 9010A Operator's Manual for information about using stored programs. Once the program is in memory and is executed, it will begin looping through a display sequence which prompts the operator to select from the available Quick Functions, then prompts for address information to use with the selected test.

```
include "80188.POD"
declarations
        assign rega to key
        assign regb to incr
        assign reg8 to highaddr
        assign reg9 to lowaddr

setup information
        pod = 80188

program Main
        dpy QUICK 80188 OPERATIONS              ! scroll through tests
        execute delay
ramtest:
        dpy QUICK RAM TEST (Y-N)?key
        if key = 0 goto romtest
        execute quickram                        ! do the ram test
romtest:
        dpy QUICK ROM TEST (Y-N)?key
        if key = 0 goto fill
        execute quickrom                        ! do the rom test
fill:
        dpy QUICK FILL OR VERIFY (Y-N)?key
        if key = 0 goto ramtest
        execute filltest                        ! do the fill or verify

program quickram                                ! Quick RAM test
        dpy ADDRESS INCREMENT? /key
        incr = key shl 4
        incr = incr or 1                        ! only basic RAM test
        dpy BEGINNING ADDRESS? /key
        lowaddr = key or 2000000                ! mask special 2XXXXXX address
        dpy ENDING ADDRESS? /key                ! for Quick Ram test
        highaddr = key or 2000000
        write @ lowaddr = 0                     ! Start Ramtest at beginning
        write @ highaddr = incr                 ! write at ending address

stat_lp:
        read @ ADR                              ! READ @ ENTER
        if DAT and F0 = F0 goto ram_err         ! if an error occurrs
        if DAT and FF = C0 goto ram_ok          ! if ramtest passes
        if DAT and F0 = A0 goto abort           ! if test is aborted
        dpy BUSY, STATUS $e                     ! else display status
        execute delay
        goto stat_lp
abort:
        dpy TEST ABORTED
        goto ram_end
ram_err:                                        ! get address of error
        read @ 200000B                          ! high byte of address
        reg1 = DAT shl 24                       ! move to msb
        read @ 200000A                          ! get next byte
        reg1 = reg1 or DAT shl 16               ! move to nmsb
        read @ 2000009                          ! get next byte
        reg1 = reg1 or DAT shl 8                ! move to nlsb
        read @ 2000008                          ! low byte of address
        reg1 = reg1 or DAT
        read @  2000010                         ! get error code
        if DAT = F0 goto rdwr_err
        if DAT = F1 goto dcd_err
        goto stat_lp
rdwr_err:
        dpy FAILED RD/WR ERROR @ $1
        goto ram_end
dcd_err:
        dpy FAILED, DECODING ERROR @ $1
        goto ram_end
ram_ok:
        dpy RAM OK
ram_end:
        stop


program quickrom                                ! Quick ROM test
        dpy ADDRESS INCREMENT? /key             ! get address increment
        incr =  key shl 4
        incr = incr or 1
        dpy BEGINNING ADDRESS? /key             ! get start address
        lowaddr = key or 3000000                ! mask in 3XXXXXX address for
        dpy ENDING ADDRESS? /key                ! 80186 quick ROM test
        highaddr = key or 3000000
        write @ lowaddr = 0                     ! starting address
```

Figure A-1. Compiler Source Program

```
        write @ highaddr = incr                    ! ending address and increment
stat_lp:
        read @ ADR                                 ! READ @ ENTER
        if DAT and FO = CO goto rom_done
        if DAT and FO = AO goto abort
        dpy BUSY, STATUS $e                        ! display status
        execute delay                              ! delay speeds test by not
        goto stat_lp                               ! interrupting the pod during
rom_done:                                          ! the quick test
        read @ ADR
        if DAT and FF = C1 goto rom_err            ! if code C1, then an error
        if DAT and FF = CO goto chksum             ! occurred
abort:
        dpy TEST ABORTED
        goto rom_end

rom_err:
        read @ 300000E
        dpy INACTIVE BITS DETECTED $e
        goto rom_end
chksum:
        read @ 300000C                              ! display Checksum
        reg9 = DAT
        read @ 300000D
        DAT = DAT shl 8 or reg9
        dpy CHECKSUM = $e
        goto rom_end
rom_end:
        stop

program filltest                                   ! Quick fill and verify
        dpy BEGINNING ADDRESS? /lowaddr            ! get address info
        dpy ENDING ADDRESS? /highaddr
        dpy FILL MEMORY (Y-N)?key                  ! get type of test
        if key = 1 goto fill
        dpy VERIFY MEMORY (Y-N)?key
        if key = 1 goto verify
        dpy FILL AND VERIFY (Y-N)?key
        if key = 1 goto fil_ver
fill:
        reg1 = 1                                   ! code 1 is fill only
        dpy FILL DATA?  /key                       ! get data and write to
        write @ lowaddr = key                      ! first location
        goto dotest
verify:
        reg1 = 2                                   ! code 2 is verify only
        goto dotest
fil_ver:
        reg1 = 3                                   ! 3 is fill & verify
        dpy FILL DATA? /key                        ! get data and write to
        write @ lowaddr = key                      ! first location
        goto dotest
dotest:
        dpy Address Increment? /key                ! get "Z", the address
        incr = key shl 4                           ! increment
        incr = incr or reg1
        lowaddr = lowaddr or 4000000               ! mask in 4XXXXXX, the
        highaddr = highaddr or 4000000             ! special address for
        write @ lowaddr = 0                        ! block tests.
        write @ highaddr = incr                    ! Start test
stat_lp:
        read @ ADR                                 ! READ @ ENTER for info
        if DAT and FO = FO goto ver_err            ! code FX means an error
        if DAT and FO = CO goto ver_com            ! occurred
        if DAT and FO = AO goto abort
        dpy BUSY, STATUS $e
        execute delay                              ! delay speeds test by
        goto stat_lp                               ! not interrupting the
ver_err:                                           ! pod durring quick test
        read @ 400000B                             ! get high word of error Addr.
        reg 1 = DAT shl 24
        read @ 400000A
        reg 1 = reg1 or DAT shl 16
        read @ 4000009
        reg 1 = reg1 or DAT shl 8
        read @ 4000008                             ! get low word error Addr.
        reg 1 = reg 1 or DAT
        dpy FAILED VERIFY @ $1
        goto fil_end
ver_com:
        dpy TEST COMPLETE, NO ERRORS
        goto fil_end
abort:
        dpy TEST ABORTED
        goto fil_end

fil_end:
        stop

program delay
        declarations
        assign reg1 to counter
        counter = 30
dloop:
        counter = counter dec
        if counter ) 0 goto dloop
```

Figure A-1. Compiler Source Program (cont)

```
include "80188.POD"
*
*  80188 pod data file - Version 1.0
*
*
*
*       FORCELN extrdy = 0
*       FORCELN hold = 1
*       busadr = 0000
*       uutadr = FFFF0
declarations
 assign rega to key
 assign regb to incr
 assign reg8 to highaddr
 assign reg9 to lowaddr

setup information
 pod - 80188

program 0
 dpy QUICK 80188 OPERATIONS  ! scroll through tests
 execute 4
0:
 dpy QUICK RAM TEST (Y-N)?A
 if REGA = 0 goto 1
 execute 1                          ! do the ram test
1:
 dpy QUICK ROM TEST (Y-N)?A
 if REGA = 0 goto 2
 execute 2                          ! do the rom test
2:
 dpy QUICK FILL OR VERIFY (Y-N)?A
 if REGA = 0 goto 0
 execute 3                          ! do the fill or verify

program 1                                   ! Quick RAM test
 dpy ADDRESS INCREMENT? /A
 REGB = REGA shl 4
 REGB = REGB or 1                           ! only basic RAM test
 dpy BEGINNING ADDRESS? /A
 REG9 = REGA or 2000000             ! mask special 2XXXXXX address
 dpy ENDING ADDRESS? /A  ! for Quick Ram test
 REG8 = REGA or 2000000
 write @ REG9 = 0                           ! Start Ramtest at beginning
 write @ REG8 = REGB                        ! write at ending address

0:
 read @ REGF                                ! READ @ ENTER
 if REGE and F0 = F0 goto 1                 ! if an error occurrs
 if REGE and FF = C0 goto 2                 ! if ramtest passes
 if REGE and F0 = A0 goto 3                 ! if test is aborted
 dpy BUSY, STATUS $e    ! else display status
 execute 4
 goto 0
3:
 dpy TEST ABORTED
 goto 4
1:                                          ! get address of error
 read @ 200000B                             ! high byte of address
 reg1 = REGE shl 24                         ! move to msb
 read @ 200000A                             ! get next byte
 reg1 = reg1 or REGE shl 16                 ! move to nmsb
 read @ 2000009                             ! get next byte
 reg1 = reg1 or REGE shl 8                  ! move to nlsb
 read @ 2000008                             ! low byte of address
 reg1 = reg1 or REGE
 read @  2000010                            ! get error code
 if REGE = F0 goto 5
 if REGE = F1 goto 6
 goto 0
5:
 dpy FAILED RD/WR ERROR @ $1
 goto 4
6:
 dpy FAILED, DECODING ERROR @ $1
 goto 4
2:
 dpy RAM OK
4:
 stop


program 2                                   ! Quick ROM test
 dpy ADDRESS INCREMENT? /A  ! get address increment
 REGB =  REGA shl 4
 REGB = REGB or 1
 dpy BEGINNING ADDRESS? /A  ! get start address
 REG9 = REGA or 3000000             ! mask in 3XXXXXX address for
 dpy ENDING ADDRESS? /A  ! 80186 quick ROM test
 REG8 = REGA or 3000000
 write @ REG9 = 0                           ! starting address
 write @ REG8 = REGB                        ! ending address and increment
0:
 read @ REGF                                ! READ @ ENTER
 if REGE and F0 = C0 goto 1
```

Figure A-2. Troubleshooter Program

```
 87   if REGE and F0 = A0 goto 2
 88   dpy BUSY, STATUS $e   ! display status
 89   execute 4                                ! delay speeds test by not
 90   goto 0                              ! interrupting the pod during
 91  1:                                        ! the quick test
 92   read @ REGF
 93   if REGE and FF = C1 goto 3               ! if code C1, then an error
 94   if REGE and FF = C0 goto 4               ! occurred
 95  2:
 96   dpy TEST ABORTED
 97   goto 5
 98
 99  3:
100   read @ 300000E
101   dpy INACTIVE BITS DETECTED $e
102   goto 5
103  4:
104   read @ 300000C                            ! display Checksum
105   reg9 = REGE
106   read @ 300000D
107   REGE = REGE shl 8 or reg9
108   dpy CHECKSUM = $e
109   goto 5
110  5:
111   stop
112
113  program 3                                 ! Quick fill and verify
114   dpy BEGINNING ADDRESS? /9  ! get address info
115   dpy ENDING ADDRESS? /8
116   dpy FILL MEMORY (Y-N)?A   ! get type of test
117   if REGA = 1 goto 0
118   dpy VERIFY MEMORY (Y-N)?A
119   if REGA = 1 goto 1
120   dpy FILL AND VERIFY (Y-N)?A
121   if REGA = 1 goto 2
122  0:
123   reg1 = 1                                  ! code 1 is fill only
124   dpy FILL DATA?  /A   ! get data and write to
125   write @ REG9 = REGA                       ! first location
126   goto 3
127  1:
128   reg1 = 2                                  ! code 2 is verify only
129   goto 3
130  2:
131   reg1 = 3                                  ! 3 is fill & verify
132   dpy FILL DATA? /A   ! get data and write to
133   write @ REG9 = REGA                       ! first location
134   goto 3
135  3:
136   dpy Address Increment? /A  ! get "Z", the address
137   REGB = REGA shl 4                         ! increment
138   REGB = REGB or reg1
139   REG9 = REG9 or 4000000           ! mask in 4XXXXXX, the
140   REG8 = REG8 or 4000000           ! special address for
141   write @ REG9 = 0                          ! block tests.
142   write @ REG8 = REGB                       ! Start test
143  4:
144   read @ REGF                               ! READ @ ENTER for info
145   if REGE and F0 = F0 goto 5                ! code FX means an error
146   if REGE and F0 = C0 goto 6                ! occurred
147   if REGE and F0 = A0 goto 7
148   dpy BUSY, STATUS $e
149   execute 4                                 ! delay speeds test by
150   goto 4                           ! not interrupting the
151  5:                                         ! pod durring quick test
152   read @ 400000B                            ! get high word of error Addr.
153   reg 1 = REGE shl 24
154   read @ 400000A
155   reg 1 = reg1 or REGE shl 16
156   read @ 4000009
157   reg 1 = reg1 or REGE shl 8
158   read @ 4000008                            ! get low word error Addr.
159   reg 1 = reg 1 or REGE
160   dpy FAILED VERIFY @ $1
161   goto 8
162  6:
163   dpy TEST COMPLETE, NO ERRORS
164   goto 8
165  7:
166   dpy TEST ABORTED
167   goto 8
168
169  8:
170   stop
171
172
173  program 4
174   declarations
175   assign reg1 to counter
176   REG1 = 30
177  0:
178   REG1 = REG1 dec
179   if REG1 ) 0 goto 0
180
181
```

Figure A-2. Troubleshooter Program (cont)

Appendix B
Using the Pod with a Remote 9020A

INTRODUCTION B-1.

The 80188 Pod may be used with a 9020A Micro-Systemroubleshooter in its remote IEEE-488- or RS-232-controlled mode. The only Pod-specific information that you need to know to use the 80188 Pod with a remotely operated 9020A are the commands for controlling the Enableable Status Lines. (Using Status Lines in the Local Mode is described in Section 2 of this manual.)

REMOTE SETUP OF ENABLE LINES B-2.

The two Enableable Status lines (Enable Lines) of the 80188 Pod may be changed (as in the Local Mode Setup Command) by sending setup commands via the remote bus. Paragraph 6-31 in the 9020A Operator Manual gives instructions for sending these commands. The commands for the 80188 Pod are shown in Table B-1 below, which is an addenda to Table 6-9, Enable Line Setup Commands, in the 9020A Operator Manual.

Table B-1. Enable Line Setup Commands

INTERFACE POD	S0,8	S0,9	S0,10	S0,11	S0,12	S0,13	S0,14	S0,15
80188 (both Normal and Queue Status Modes)	EXTRDY	HOLD	*	*	*	*	*	*

*If this command is sent to the 9020A, it will cause a command error (status response 95).

Appendix C
Power-Up Defaults

INTRODUCTION C-1.

Whenever power is applied to the Pod, it sends a string of Reset information to the Troubleshooter, and applies default values to many of the addresses and registers. These default values are available, in case you do not want to select them explicitly every time you use certain functions.

Many of the default values are described in detail elsewhere in this manual, but they are listed here for completeness. In some cases, there are methods available for changing default values. When you change default values, those values are usually available until power is removed from and reapplied to the Pod. The methods for changing values, when provided, are also described in other parts of this manual under the appropriate topic.

MODE C-2.

When the Pod is plugged into a UUT, the Pod mode is determined by the state of the UUT's $\overline{RD}/\overline{QSMD}$ line. If the the line is tied low when the Pod is reset (either by a power-up reset or a Troubleshooter reset), the Pod will go into the Queue Status Mode; otherwise, it will go into the Normal mode.

ADDRESSES C-3.

Introduction C-4.

The following paragraphs describe the default address parameters that are provided for the Learn operation and the Bus Test. Other default addresses not mentioned in this manual are described in the Troubleshooter Operator manual and apply to all Pods.

Address Increment C-5.

The default address increment value is 1—specifying byte accesses.

Segment Registers C-6.

The segment registers for RUN UUT (Pod Function addresses F0 0020 - F0 0026) contain the following default values:

ES Register = 0000
SS Register = 0000
CS Register = FFFF
DS Register = 0000

LEARN Operation Default Address C-7.

If you select the LEARN operation and do not specify the starting and ending addresses for the operation, the Pod specifies the default address spaces of 0000 0000 -000F FFFF for the basic word address space and 0020 0000 - 0020 FFFF for the I/O word space. The LEARN operation is then performed over these address spaces.

Default Address for Data Line Testing for BUS TEST C-8.

No address is explicitly required when you select the BUS TEST. However, as part of BUS TEST, the drivability of the data lines is tested with Write operations to a particular address supplied by the Pod. For the 80188 pod, the data line testing occurs at address 0000 0000. You may change this address with the Troubleshooter Setup function by entering the desired address for the Setup message *SET-BUS TEST @ 00000000-CHANGE?*

RUN UUT C-9.

Like BUS TEST, no address is explicitly required when you select the RUN UUT. RUN UUT is set by default to occur at address 000F FFF0. You may change this address with the Troubleshooter Setup function by entering the desired address for the Setup message *SET-RUN UUT @ FFFF0- CHANGE?*

Peripheral Control Block C-10.

INTRODUCTION C-11.

Table 4B-2 shows the specific default contents of the Peripheral Control Block registers, including those to control the chip selects, timers, and interrupts.

CHIP SELECTS C-12.

Chip Selects are defined by default to provide to widest possible range of address coverage. Table C-1 shows the default chip select specifications.

TIMERS C-13.

The timers will all be idle and unprogrammed after power-up. All of the timer output lines will be high.

INTERRUPTS C-14.

All of the interrupt lines will be configured as direct mode inputs. All interrupt lines will be defeated. Refer to Testing Interrupt Circuitry in Section 4A.

Masks C-15.

All of the error reporting control masks will be configured so that no errors will be masked from being reported by the Troubleshooter. See the individual descriptions under Masking Errors.

Enableable Status Lines C-16.

All enableable status lines are enabled by default.

Table C-1. Power-up Chip Select Default Summary

LINE	LOWER LIMIT ADDRESS	UPPER LIMIT ADDRESS	WAIT STATES
$\overline{LCS}$	0	3FFFF	3
#$\overline{PCS0}$	40000	4007F	3
#$\overline{PCS1}$	40080	400FF	3
#$\overline{PCS2}$	40100	4017F	3
#$\overline{PCS3}$	40180	401FF	3
#$\overline{PCS4}$	40200	4027F	3
*#$\overline{PCS5}$	40280	402FF	3
*#$\overline{PCS6}$	40300	4037F	3
$\overline{MCS0}$	80000	8FFFF	3
$\overline{MCS1}$	90000	9FFFF	3
$\overline{MCS2}$	A0000	AFFFF	3
$\overline{MCS3}$	B0000	BFFFF	3
$\overline{UCS}$	C0000	FFFFF	3

*$\overline{PCS5}$/A1 and $\overline{PCS6}$/A2 are configured to produce $\overline{PCS5}$ and $\overline{PCS6}$ (not A1 and A2) upon power up.

#$\overline{PCS0}$ through $\overline{PCS6}$ are configured to operate in memory space upon power up.

NOTE

Externally generated wait states will be allowed (unless ARDY and SRDY are disabled at the Troubleshooter).

Appendix D
Segment Registers

INTRODUCTION D-1.

The 80188 microprocessor uses two 16-bit registers to form addresses for memory accesses. The Pod uses the segment registers somewhat differently than the 80188 microprocessor. This Appendix describes how the Segment Registers work in both the 80188 microprocessor and the Pod. During troubleshooting operations, the Pod uses the segment registers in a way that is transparent to the user. The user simply enters a 5-digit address on the Troubleshooter keyboard, and the Pod knows how to form the address for a UUT access.

SEGMENT REGISTERS IN THE 80188 D-2.

The 80188 has four segment registers that form the effective memory address. The registers are the Extra Data register (ES), Stack register (SS), Code register (CS), and Data register (DS). The CS register points to the current code segment from which instructions are fetched. The DS register points to the current data segment, which generally contains program variables. The SS register points to the current stack segment; stack operations are performed on locations in this segment. The ES register points to the current extra data segment, which is typically used for data storage.

The segment registers normally contain a base address, which identifies the start of that memory segment. The contents of the register are shifted four bits to the left and added to the 16-bit "offset" address to make the completed 20-bit address.

SEGMENT REGISTERS IN POD OPERATION D-3.

The Pod only uses the lower 16 bits of the address internally and does not use the segment information. For external addressing, the lower 12 bits of the segments are kept at 0, and the 20-bit addresses are formed as usual:

S 0 0 0	Segment Register
+ XXXX	Offset Register
SXXXX	Complete Address

NOTE

Segment registers may be predefined before doing a RUN UUT operation. See RUN UUT in Section 4A for more information.

Appendix E
Pod Resets

There are several conditions that reset the Pod to its initial state. When the Pod is reset, it returns to its predefined start-up configuration and it also performs some internal set-up procedures, such as checking the $\overline{\text{QSMD}}$ line and entering the correct operating mode (see Testing in the Queue Status Mode).

The contents of all user-defined variables are not changed by a Troubleshooter-generated reset. The contents of the Chip Select Control Registers, for example, are also saved by the Pod in its RAM memory. After the Pod has been reset by the Troubleshooter, the Pod automatically reprograms the Chip Select control registers by reading their values from RAM, and replacing them in the appropriate registers in the Pod's microprocessor. Only removing and restoring power to the Pod will cause these values to change. See the descriptions of the specific items in Sections 4A and 4B.

Reset signals may come from either the Troubleshooter, the UUT, or from within the Pod itself. The conditions which cause the Pod to be reset are described below:

NOTE

The $\overline{\text{RES}}$ input signal from the UUT is not allowed to reset the Pod's microprocessor, except during the RUN UUT mode.

Troubleshooter-generated Pod resets:

- When you perform a BUS TEST.
- When you change the sync mode.
- When you enable or disable a user-enableable status line (EXTRDY or HOLD).
- When the Pod exits the RUN UUT mode.
- When the Pod recovers from a Pod timeout or a UUT power fail condition.

UUT-generated Pod resets:

- When the UUT asserts a low $\overline{\text{RES}}$ signal at the microprocessor socket while the Pod is in the RUN UUT mode.

Pod-generated Pod resets:

- When you apply power to the Troubleshooter (and the Pod).
- When the UUT experiences a low-power condition (both Vcc pins fall below 3.5 volts) while the Pod is in the RUN UUT mode.

Appendix F
Problems Due to a Marginal UUT

INTRODUCTION F-1.

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

UUT OPERATING SPEED AND MEMORY ACCESS F-2.

Some UUTs operate at speeds which approach the time limits for memory access. The Pod contributes a slight time delay which causes memory access problems to become apparent.

UUT NOISE LEVELS F-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 will actually be made worse (and easier to troubleshoot) through use of the Pod and Troubleshooter.

BUS LOADING F-4.

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

CLOCK LOADING F-5.

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

Appendix G
Operating the Pod in the Queue Status Mode

The Pod will operate in either of the 80188's two modes: the Normal mode or the Queue Status mode. UUTs are normally wired permanently so that they initialize in one mode or the other. If your UUT is supposed to run in the Queue Status mode, then the $\overline{RD}/\overline{QSMD}$ line should be tied low at the UUT's microprocessor socket. If the $\overline{RD}/\overline{QSMD}$ line is not tied low, the Pod will enter the Normal mode.

The Pod will change modes (depending upon the state of the $\overline{QSMD}$ line) whenever the Pod is reset. See Pod Resets for a list of reset conditions.

When the Pod is in the Queue Status mode, the $\overline{QSMD}$ status bit (bit 10) will be set to 0. When the Pod is in the Normal mode, the $\overline{QSMD}$ bit is set to 1. To read the status bits, perform a READ @ STS operation on the Troubleshooter.

NOTE

If you are testing a UUT that does not operate in the Queue Status mode, and the $\overline{RD}$ output line is accidently tied low, the Pod will go into the Queue Status mode. If this happens, the Pod will report address, data, or control drivability errors that do not actually exist on the UUT.

NOTE

The Pod will not report drivability errors on $\overline{RD}$ when the Pod is in the Queue Status mode.

Appendix H
Peripheral Control Block

INTRODUCTION

The information in this Appendix is reprinted from the Intel Corporation's Data Sheet for the iAPX 186* High Integration 16-Bit Microprocessor. It is provided here as an aid to configuring the individual bits of the Pod's Peripheral Control Block registers, which you may need to do to set up the Pod to work with your UUT or which you may want to change while troubleshooting individual components on the UUT.

Refer to Appendix I for information about using the PCB Recover Tape to extract set-up data for the Peripheral Control Block from a known-good UUT.

NOTE

The 80188 microprocessor addresses the PCB (Peripheral Control Block) in word accesses. The Pod, on the other hand, reads and writes to these locations using byte accesses. The control word for each function is therefore composed of a high and low byte—the low byte being at an even address, the high byte at the next odd address. For example: the UMCS register is at offset A0, with a default value of C03B; this would be read by the Troubleshooter and Pod as READ @ F0 01A0 = 3B and READ @ F0 01A1 = C0.

INTERNAL PERIPHERAL INTERFACE**

All the 80188 integrated peripherals are controlled via 16-bit registers contained within an internal 256-byte control block. This control block may be mapped into either memory or I/O space. Internal logic will recognize the address and respond to the bus cycle. During bus cycles to internal registers, the bus controller will signal the operation externally (i.e., the $\overline{RD}$, $\overline{WR}$, status, address, data, etc., lines will be driven as in a normal bus cycle), but D7-0, SRDY, and ARDY will be ignored. The base address of the control block must be on an even 256-byte boundary (i.e., the lower 8 bits of the base address are all zeros). All of the defined registers within this control block may be read or written by the 80188 CPU at any time. The location of any register contained within the 256-byte control block is determined by the current base address of the control block.

The control block base address is programmed via a 16-bit relocation register contained within the control block at F0 01FE from the base address of the control block (see Figure H-1). It provides the upper 12 bits of the base address of the control block. Note that mapping the control register block into an address range

* iAPX 166 is a trademark of the Intel Corporation

corresponding to a chip-select range is not recommended (the chip select circuitry is discussed later in this data sheet). In addition, bit 12 of this register determines whether the control block will be mapped into I/O or memory space. If this bit is 1, the control block will be located in memory space, whereas if the bit is 0, the control block will be located in I/O space. If the control register block is mapped into I/O space, the upper 4 bits of the base address must be programmed as 0 (since I/O addresses are only 16 bits wide).

In addition to providing relocation information for the control block, the relocation register contains bits which place the interrupt controller into iRMX mode, and cause the CPU to interrupt upon encountering ESC instructions. At RESET, the relocation register is set to 20FFH. This causes the control block to start at FF00H in I/O space. An offset map of the 256-byte control register block is shown in Figure H-2.

The integrated 80188 peripherals operate semi-autonomously from the CPU. Access to them for the most part is via software read/write of the control and data locations in the control block. Most of these registers can be both read and written. A few dedicated lines, such as interrupts and DMA request provide real-time communication between the CPU and peripherals as in a more conventional system utilizing discrete peripheral blocks. The overall interaction and function of the peripheral blocks has not substantially changed.

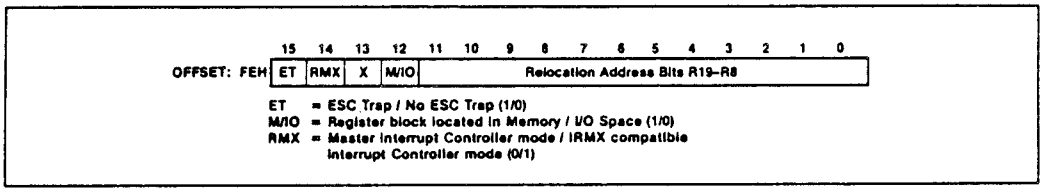


Figure H-1. Relocation Register

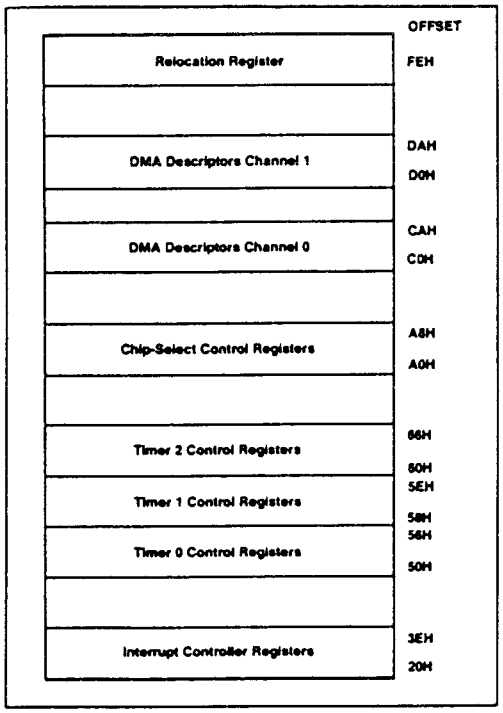


Figure H-2. Internal Register Map

CHIP-SELECT/READY GENERATION LOGIC

The 80188 contains logic which provides programmable chip-select generation for both memories and peripherals. In addition, it can be programmed to provide READY (or WAIT state) generation. It can also provide latched address bits A1 and A2. The chip-select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

Memory Chip Selects

The 80188 provides 6 memory chip select outputs for 3 address areas: upper memory, lower memory, and midrange memory. One each is provided for upper memory and lower memory, while four are provided for midrange memory.

The range for each chip select is user-programmable and can be set to 2K, 4K, 8K, 16K, 32K, 64K, 128K (plus 1K and 256K for upper and lower chip selects). In addition, the beginning or base address of the midrange memory chip select may also be selected. Only one chip select may be programmed to be active for any memory location at a time. All chip select sizes are in bytes.

Upper Memory $\overline{CS}$

The 80188 provides a chip select, called $\overline{UCS}$, for the top of memory. The top of memory is usually used as the system memory because after reset the 80188 begins executing at memory location FFFF0H.

The upper limit of memory defined by this chip select is always FFFFFH, while the lower limit is programmable. By programming the lower limit, the size of the select block is also defined. Table H-1 shows the relationship between the base address selected and the size of the memory block obtained.

The lower limit of this memory block is defined in the UMCS register (see Figure H-3). This register is at F0 01A0 in the internal control block. The legal values for bits 6-13 and the resulting starting address and memory block sizes are given in Table H-1. Any combination of bits 6-13 not shown in Table H-1 will result in undefined operation. After reset, the UMCS register is programmed for a 1K area. It must be reprogrammed if a larger upper memory area is desired.

Table H-1. UMCS Programming Values

Starting Address (Base Address)	Memory Block Size	UMCS Value (Assuming R0=R1=R2=0)
FFC00	1K	FFF8H
FF800	2K	FFB8H
FF000	4K	FF38H
FE000	8K	FE38H
FC000	16K	FC38H
F8000	32K	F838H
F0000	64K	F038H
E0000	128K	E038H
C0000	256K	C038H

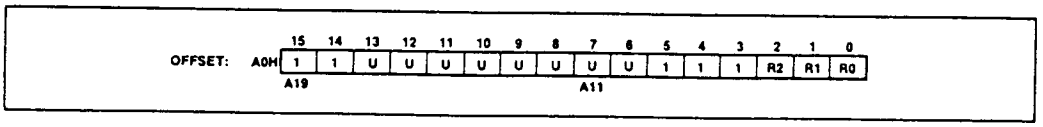


Figure H-3. UMCS Register

Any internally generated 20-bit address whose upper 16 bits are greater than or equal to UMCS (with bits 0-5 "0") will cause $\overline{USC}$ to be activated. UMCS bits R2-R0 are used to specify READY mode for the area of memory defined by this chip-select register, as explained below.

Lower Memory $\overline{CS}$

The 80188 provides a chip select for low memory called $\overline{LCS}$. The bottom of memory contains the interrupt vector table, starting at location 00000H.

The lower limit of memory defined by this chip select is always 0H, while the upper limit is programmable. By programming the upper limit, the size of the memory block is also defined. Table H-2 shows the relationship between the upper address selected and the size of the memory block obtained.

The upper limit of this memory block is defined in the LMCS register (see Figure H-4). This register is at F0 01A2 in the internal control block. The legal values for bits 6-15 and the resulting upper address and memory block sizes are given in Table H-2. Any combination of bits 6-15 not shown in Table H-2 will result in undefined operation. After reset, the LMCS register value is undefined. However, the $\overline{LCS}$ chip-select line will not become active until the LMCS register is accessed.

Any internally generated 20-bit address whose upper 16 bits are less than or equal to LMCS (with bits 0-5 "1") will cause $\overline{LCS}$ to be active. LMCS register bits R2-R0 are used to specify the READY mode for the area of memory defined by this chip-select register.

Mid-Range Memory CS

The 80188 provides four MCS lines which are active within a user-locatable memory block. This block can be located anywhere within the 80188 1 M byte memory address space exclusive of the areas defined by UCS and LCS. Both the base address and size of this memory block are programmable.

The size of the memory block defined by the mid-range select lines, as shown in Table H-3, is determined by bits 8-14 of the MPCS register (see Figure H-5). This register is at location A8H in the internal control block. One and only one of bits 8-14 must be set at a time. Unpredictable operation of the $\overline{MCS}$ lines will otherwise occur. Each of the four chip-select lines is active for one of the four equal contiguous divisions of the mid-range block. Thus, if the total block size is 32K, each chip select is active for 8K of memory with $\overline{MCS0}$ being active for the first range and $\overline{MCS3}$ being active for the last range.

Table H-2. LMCS Programming Values

Upper Address	Memory Block Size	LMCS Value (Assuming R0=R1=R2=0)
003FFH	1K	0038H
007FFH	2K	0078H
00FFFH	4K	00F8H
01FFFH	8K	01F8H
03FFFH	16K	03F8H
07FFFH	32K	07F8H
0FFFFH	64K	0FF8H
1FFFFH	128K	1FF8H
3FFFFH	256K	3FF8H

Table H-3. MPCS Programming Values

Total Block Size	Individual Select Size	MPCS Bits 14-8
8K	2K	0000001B
16K	4K	0000010B
32K	8K	0000100B
64K	16K	0001000B
128K	32K	0010000B
256K	64K	0100000B
512K	128K	1000000B

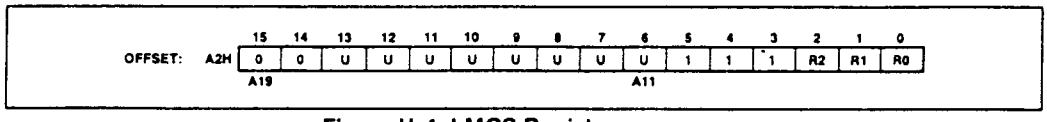


Figure H-4. LMCS Register

The EX and MS bits in MPCS relate to peripheral functionality as described in a later section.

The base address of the mid-range memory block is defined by bits 15-9 of the MMCS register (see Figure H-6). This register is at F0 01A6 in the internal control block. These bits correspond to bits A19-A13 of the 20-bit memory address. Bits A12-A0 of the base address are always 0. The base address may be set at any integer multiple of the size of the total memory block selected. For example, if the mid-range block size is 32K (or the size of the block for which each $\overline{MCS}$ line is active is 8K), the block could be located at 10000H or 18000H, but not at 14000H, since the first few integer multiples of a 32K memory block are 0H, 8000H, 10000H, 18000H, etc. After reset, the contents of both of these registers is undefined. However, none of the $\overline{MCS}$ lines will be active until both the MMCS and MPCS registers are accessed. (Note: the Pod provides default values for all CS lines.)

MMCS bits R2-R0 specify READY mode of operation for all mid-range chip selects. All devices in mid-range memory must use the same number of WAIT states.

The 512K block size for the mid-range memory chip selects is a special case. When using 512K, the base address would have to be either locations 00000H or 80000H. If it were to be programmed at 00000H when the $\overline{LCS}$ line was programmed, there would be an internal conflict between the $\overline{LCS}$ ready generation logic and the $\overline{MCS}$ ready generation logic. Likewise, if the base address were programmed at 80000H, there would be a conflict with the $\overline{UCS}$ ready generation logic. Since the $\overline{LCS}$ chip-select line does not become active until programmed, while the $\overline{UCS}$ line is active at reset, the memory base can be set only at 00000H. If this base address is selected, however, the $\overline{LCS}$ range must not be programmed.

Peripheral Chip Selects

The 80188 can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. This base address may be located in either memory or I/O space.

Seven $\overline{CS}$ lines called $\overline{PCS0}$-6 are generated by the 80188. The base address is user programmable; however it can only be a multiple of 1K bytes, i.e., the least significant 10 bits of the starting address are always 0.

$\overline{PCS5}$ and $\overline{PCS6}$ can also be programmed to provide latched address bits A1, A2. If so programmed, they cannot be used as peripheral selects. These outputs can be connected directly to the A0, A1 pins used for selecting internal registers of 8-bit peripheral chips. The scheme simplifies the hardware interface because the 8-bit registers of peripherals are simply treated as 16-bit registers located on even boundaries in I/O space or memory space where only the lower 8-bits of the register are significant: the upper 8-bits are "don't cares".

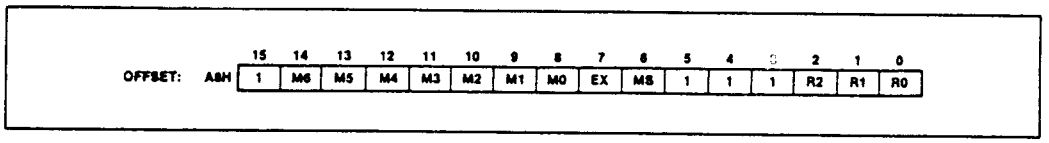


Figure H-5. MPCS Register

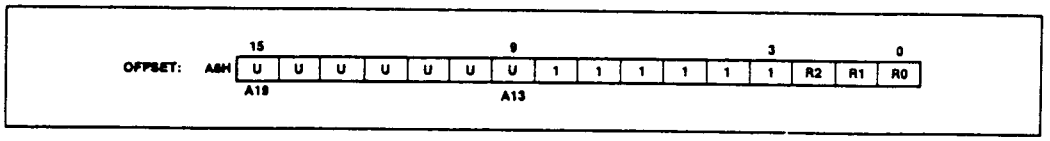


Figure H-6. MMCS Register

The starting address of the peripheral chip-select block is defined by the PACS register (see Figure H-7). This register is located at F0 01A4 in the internal control block. Bits 15-6 of this register correspond to bits 19-10 of the 20-bit Programmable Base Address (PBA) of the peripheral chip-select block. Bits 9-0 of the PBA of the peripheral chip-select block are all zeros. If the chip-select block is located in I/O space, bits 12-15 must be programmed zero, since the I/O address is only 16 bits wide. Table H-4 shows the address range of each peripheral chip select with respect to the PBA contained in PACS register.

The user should program bits 15-6 to correspond to the desired peripheral base location. PACS bits 0-2 are used to specify READY mode for $\overline{PCS0}$-$\overline{PCS3}$.

The mode of operation of the peripheral chip selects is defined by the MPCS register (which is also used to set the size of the mid-range memory chip-select block, see Figure H-8). This register is located at F0 01A8 in the internal control block. Bit 7 is used to select the function of $\overline{PCS5}$ and $\overline{PCS6}$, while bit 6 is used to select whether the peripheral chip selects are mapped into memory or I/O space. Table H-5 describes the programming of these bits. After reset, the contents of both the MPCS and the PACS registers are undefined, however none of the $\overline{PCS}$ lines will be active until both of the MPCS and PACS registers are accessed.

MPCS bits 0-2 are used to specify READY mode for $\overline{PCS4}$-$\overline{PCS6}$ as outlined below.

READY Generation Logic

The 80188 can generate a "READY" signal internally for each of the memory or peripheral $\overline{CS}$ lines. The number of WAIT states to be inserted for each peripheral or memory is programmable to provide 0-3 wait states for all accesses to the area for which the chip select is active. In addition, the 80188 may be programmed to either ignore external READY for each chip-select range individually or to factor external READY with the integrated ready generator.

READY control consists of 3 bits for each $\overline{CS}$ line or group of lines generated by the 80188. The interpretation of the ready bits is shown in Table H-6.

Table H-4. PCS Address Ranges

PCS Line	Active between Locations
PCS0	PBA — PBA+127
PCS1	PBA+128 — PBA+255
PCS2	PBA+256 — PBA+383
PCS3	PBA+384 — PBA+511
PCS4	PBA+512 — PBA+639
PCS5	PBA+640 — PBA+767
PCS6	PBA+768 — PBA+895

Table H-5. MS, EX Programming Values

Bit	Description
MS	1 = Peripherals mapped into memory space. 0 = Peripherals mapped into I/O space.
EX	0 = 5 $\overline{PCS}$ lines. A1, A2 provided. 1 = 7 $\overline{PCS}$ lines. A1, A2 are not provided.

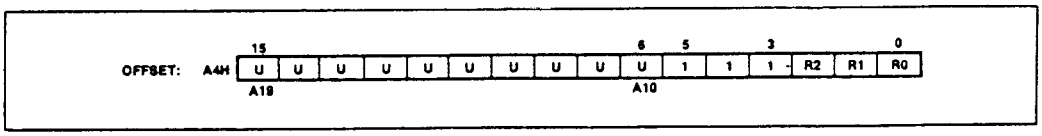


Figure H-7. PACS Register

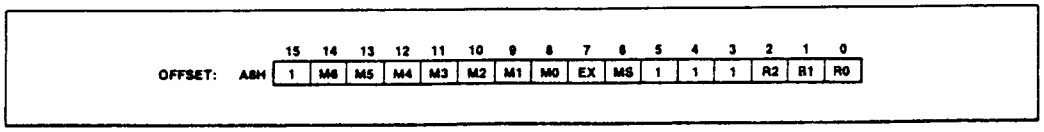


Figure H-8. MPCS Register

The internal ready generator operates in parallel with external READY, not in series if the external READY is used (R2 = 0). This means, for example, if the internal generator is set to insert two wait states, but activity on the external READY lines will insert four wait states, the processor will only insert four wait states, not six. This is because the two wait states generated by the internal generator overlapped the first two wait states generated by the external ready signal. Note that the external ARDY and SRDY lines are always ignored during cycles accessing internal peripherals.

R2-R0 of each control word specifies the READY mode for the corresponding block, with the exception of the peripheral chip selects: R2-R0 of PACS set the $\overline{PCS0}$-3 READY mode, R2-R0 of MPCS set the $\overline{PCS4}$-6 READY mode.

Chip Select/Ready Logic and Reset

Upon reset, the Chip-Select/Ready Logic will perform the following actions:

- All chip-select outputs will be driven HIGH.
- Upon leaving RESET, the $\overline{UCS}$ line will be programmed to provide chip selects to a 1K block with the accompanying READY control bits set at 011 to allow the maximum number of internal wait states in conjunction with external Ready consideration (i.e., UMCS resets to FFFBH).
- No other chip select or READY control registers have any predefined values after RESET. They will not become active until the CPU accesses their control registers. Both the PACS and MPCS registers must be accessed before the $\overline{PCS}$ lines will become active. Note: this is how the $\overline{CS}$ lines behave in RUNUUT. In normal troublshooting, the pod provides default values.

DMA CHANNELS

The 80188 DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory to I/O) or within the same space (e.g., Memory to Memory or I/O to I/O). Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer. Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data. This provides a maximum data transfer rate of one Mword/sec.

DMA Operation

Each channel has six registers in the control block which define each channel's specification operation. The control registers consist of a 20-bit Source pointer (2 words), a 20-bid Destination pointer (2 words), and 16-bit Transfer Counter, and a 16-bit Control Word. The format of the DMA Control Blocks is shown in Table H-7. The Transfer Count Register (TC) specifies the number of DMA transfers to be performed. Up to 64K byte or word transfers can be performed with automatic termination. The Control Word defines the channel's operation (see Figure H-9). All registers may be modified or altered during any DMA activity. Any changes made to these registers will be reflected immediately in DMA operation.

Table H-6. READY Bits Programming

R2	R1	R0	Number of WAIT States Generated
0	0	0	0 wait states, external RDY also used.
0	0	1	1 wait state inserted, external RDY also used.
0	1	0	2 wait states inserted, external RDY also used.
0	1	1	3 wait states inserted, external RDY also used.
1	0	0	0 wait states, external RDY ignored.
1	0	1	1 wait state inserted, external RDY ignored.
1	1	0	2 wait states inserted, external RDY ignored.
1	1	1	3 wait states inserted, external RDY ignored.

Table H-7. DMA Control Block Format

Register Name	Register Address	
	Ch. 0	Ch. 1
Control Word	CAH	DAH
Transfer Count	C8H	D8H
Destination Pointer (upper 4 bits)	C6H	D6H
Destination Pointer	C4H	D4H
Source Pointer (upper 4 bits)	C2H	D2H
Source Pointer	C0H	D0H

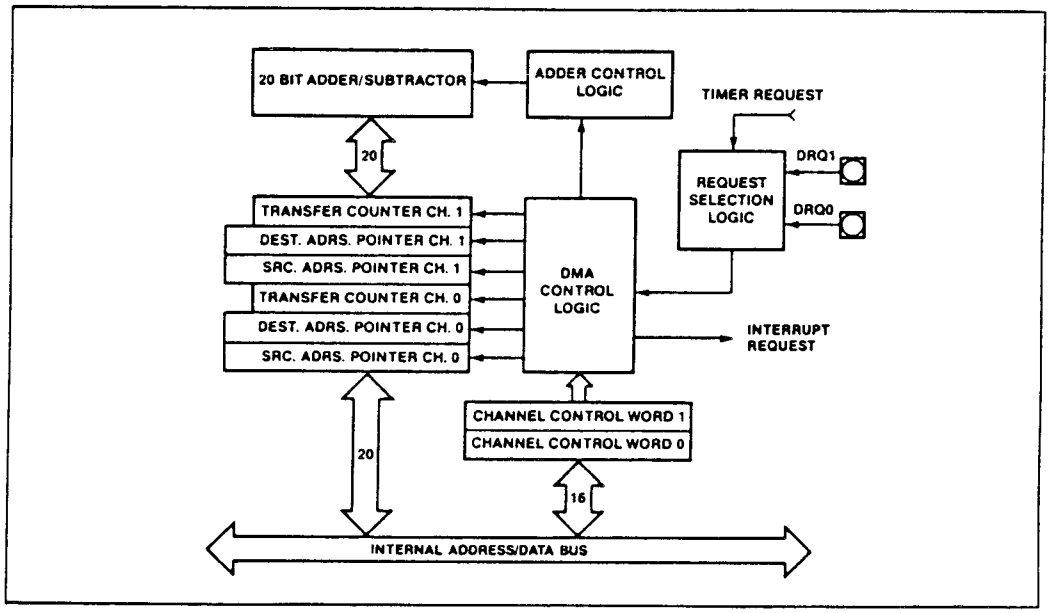


Figure H-9. DMA Block Unit Diagram

DMA Channel Control Word Register

Each DMA Channel Control Word determines the mode of operation for the particular 80188 DMA channel. This register specifies:

- the mode of synchronization;
- whether interrupts will be generated after the last transfer;
- whether DMA activity will cease after a programmed number of DMA cycles;
- the relative priority of tne DMA channel with respect to the other DMA channel;
- whether the source pointer will be incremented, decremented, or maintained constant after each transfer;
- whether the source pointer addresses memory or I/O space;
- whether the destination pointer will be incremented, decremented, or maintained constant after each transfer; and
- whether the destination pointer will address memory or I/O space.

The DMA channel control registers may be changed while the channel is operating. However, any changes made during operation will affect the current DMA transfer.

DMA Control Word Bit Descriptions

ST/$\overline{\text{STOP}}$	Start/stop (1/0) Channel.
CHG/$\overline{\text{NOCHG}}$:	Change/ Do not change (1/0)) ST/$\overline{\text{STOP}}$ bit. If this bit is set when writing to the control word, the ST/$\overline{\text{STOP}}$ bit will be programmed by the write to the control word. If this bit is cleared when writing the control word, the ST/$\overline{\text{STOP}}$ bit will not be altered. This bit is not stored; it will always be a 0 on read.
INT:	Enable Interrupts to CPU on byte count termination.
TC:	If set, DMA will terminate when the contents of the Transfer Count register reach zero. The ST/$\overline{\text{STOP}}$ bit will also be reset at this point if TC is set. If this bit is cleared, the DMA unit will decrement the transfer count register for each DMA cycle, but the DMA transfer will not stop when the contents of the TC register reach zero.
SYN: (2 bits)	00 No synchronization. NOTE: The ST bit will be cleared automatically when the contents of the TC register reach zero regardless of the state of the TC bit. 01 Source synchronization. 10 Designation synchronization. 11 Unused.
SOURCE:INC	Increment source pointer by 1 after each transfer.
M/$\overline{\text{IO}}$	Source pointer is in M/IO space (1/0).
DEC	Decrement source pointer by 1 after each transfer.
DEST: INC	Increment destination pointer by 1 after each transfer.
M/$\overline{\text{IO}}$	Destination pointer is in M/IO space (10).
DEC	Decrement destination pointer by 1 after each transfer.
P	Channel priority - relative to other channel. 0 low priority. 1 high priority. Channels will alternate cycles if both set at same priority level.

TDRQ	0: Disable DMA requests from timer 2. 1: Enable DMA requests from timer 2.
Bit 3	Bit 3 is not used.

If both INC and DEC are specified for the same pointer, the pointer will remain constant after each cycle.

DMA Destination and Source Pointer Registers

Each DMA channel maintains a 20-bit source and a 20-bit destination pointer. Each of these pointers takes up two full 16-bit registers in the peripheral control block. The lower four bits of the upper register contain the upper four bits of the 20-bit physical address (see Figure H-10a). These pointers may be individually incremented or decremented after each transfer. If word transfers are performed the pointer is incremented or decremented by two. Each pointer may point into either memory or I/O space. Since the DMA channels can perform transfers to or from odd addresses, there is no restriction on values for the pointer registers. Higher transfer rates can be obtained if all word transfers are performed to even addresses, since this will allow data to be accessed in single memory access.

DMA Transfer Count Register

Each DMA channel maintains a 16-bit transfer count register (TC). This register is decremented after every DMA cycle, regardless of the state of the TC bit in DMA Control Register. If the TC bit in the DMA control word is set, however, DMA activity will terminate when the transfer count register reaches zero.

DMA Requests

Data transfers may be either source or destination synchronized, that is either the source of the data or the destination of the data may request the data transfer. In addition, DMA transfers may be unsynchronized; that is, the transfer will take place continually until the correct number of transfers has occurred. When source or unsynchronized transfers are performed, the DMA channel may begin another transfer immediately after the end of a previous DMA transfer. This allows a complete transfer to take place every 2 bus cycles or eight clock cycles (assuming no wait states). No prefetching occurs when destination synchronization is performed, however. Data will not be fetched from the source address until the destination device signals that it is ready to receive it. When destination synchronized transfers are requested, the DMA controller will relinquish control of the bus after every transfer. If no other bus activity is initiated, another DMA cycle will begin after two processor clocks. This is done to allow the destination device time to remove its request if another transfer is not desired. Since the DMA controller will relinquish the bus, the CPU can initiate a bus cycle. As a result, a complete bus cycle will often be inserted between destination synchronized transfers. These lead to the maximum DMA transfer rates shown in Table H-8.

DMA Acknowledge

No explicit DMA acknowledge pulse is provided. Since both source and destination pointers are maintained, a read from a requesting source, or a write to a requesting destination, should be used as the DMA acknowledge signal. Since the chip-select lines can be programmed to be active for a given block of memory or I/O space, and the DMA pointers can be programmed to point to the same given block, a chip-select line could be used to indicate a DMA acknowledge.

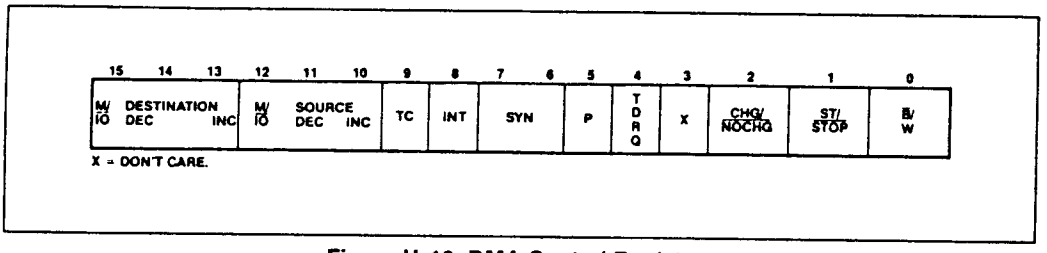


Figure H-10. DMA Control Register

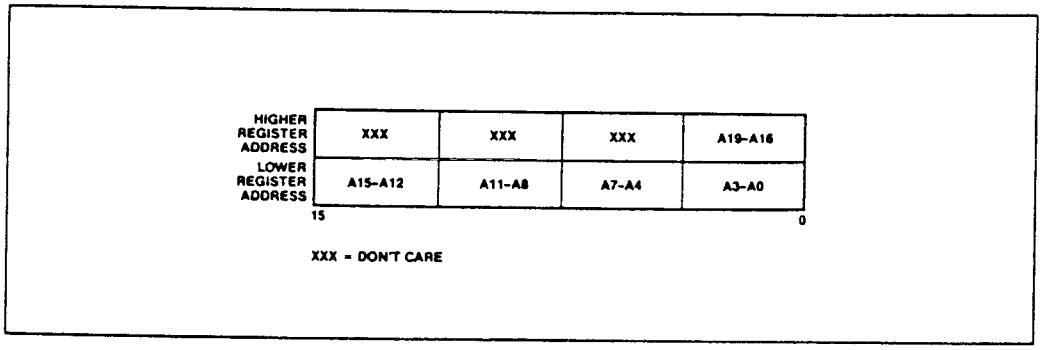


Figure H-10a. DMA Memory Pointer Register Format

Table H-8. Maximum DMA Transfer Rates

Type of Synchronization Selected	CPU Running	CPU Halted
Unsynchronized	1MBytes/sec	1MBytes/sec
Source Synch	1MBytes/sec	1MBytes/sec
Destination Synch	.65MBytes/sec	.75MBytes/sec

DMA Priority

The DMA channels may be programmed such that one channel is always given priority over the other, or they may be programmed such as to alternate cycles when both have DMA requests pending. DMA cycles always have priority over internal CPU cycles except between locked memory accesses or word accesses the odd memory locations; however, an external bus hold takes priority over an internal DMA cycle. Because an interrupt request cannot suspend a DMA operation and the CPU cannot access memory during a DMA cycle, interrupt latency time will suffer during sequences of continuous DMA cycles. An NMI request, however, will cause all internal DMA activity to halt. This allows the CPU to quickly respond to the NMI request.

DMA Programming

DMA cycles will occur whenever the ST/$\overline{\text{STOP}}$ bit of the Control Register is set. If synchronized transfers are programmed, a DRQ must also have been generated. Therefore, the source and destination transfer pointers, and the transfer count register (if used) must be programmed before this bit is set.

Each DMA register may be modified while the channel is operating. If the CHG/$\overline{\text{NOCHG}}$ bit is cleared when the control register is written, the ST/$\overline{\text{STOP}}$ bit of the control register will not be modified by the write. If multiple channel registers are modified, it is recommended that a LOCKED string transfer be used to prevent a DMA transfer from occurring between updates to the channel registers.

DMA Channels and Reset

Upon RESET, the DMA channels will perform the following actions:

- The Start/Stop bit for each channel will be reset to STOP.
- Any transfer in progress is aborted.

TIMERS

The 80188 provides three internal 16-bit programmable timers (see Figure H-11). Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, this third timer can be used as a prescaler to the other two, or as a DMA request source.

Timer Operation

The timers are controlled by 11 16-bit registers in the internal peripheral control block. The configuration of these registers is shown in Table H-9. The count register contains the current value of the timer. It can be read or written at any time independent of whether the timer is running or not. The value of this register will be incremented for each timer event. Each of the timers is equipped with a MAX COUNT register, which defines the maximum count the timer will reach. After reaching the MAX COUNT r gister value, the timer count value will reset to zero during the same clock, i.e., the maximum count value is never stored in the count register itself. Timers 0 and 1 are, in addition, equipped with a second MAX COUNT register, which enables the timers to alternate their count between two different MAX COUNT values programmed by the user. If a single MAX COUNT register is used, the timer output pin will switch LOW for a single clock, 2 clocks after the maximum count value has been reached. In the dual MAX COUNT register mode, the output pin will indicate which MAX COUNT register is currently in use, thus allowing nearly complete freedom in selecting waveform duty cycles. For the timers with two MAX COUNT registers, the RIU bit in the control register determines which is used for the comparison.

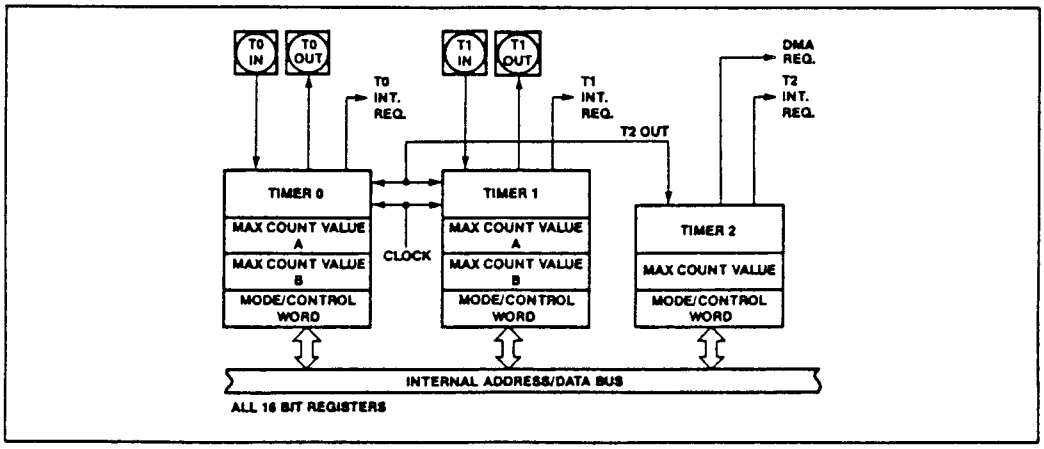


Figure H-11. Timer Block Diagram

Each timer gets serviced every fourth CPU-clock cycle, and thus can operate at speeds up to one-quarter the internal clock frequency (one-eighth the crystal rate). External clocking of the timers may be done at up to a rate of one-quarter of the internal CPU-clock rate (2 MHz for an 8 MHz CPU clock). Due to internal synchronization and pipelining of the timer circuitry, a timer output may take up to 6 clocks to respond to any individual clock or gate input.

Since the count registers and the maximum count registers are all 16 bits wide, 16 bits of resolution are provided. Any Read or Write access to the timers will add one wait state to the minimum four-clock bus cycle. However, this is needed to synchronize and coordinate the internal data flows between the internal timers and the internal bus.

The timers have several programmable options.

- All three timers can be set to halt or continue on a terminal count.
- Timers 0 and 1 can select between internal and external clocks, alternate between MAX COUNT registers and be set to retrigger on external events.
- The timers may be programmed to cause an interrupt on terminal count.

Timer Mode/Control Register

The mode/control register (see Figure H-12) allows the user to program the specific mode of operation or check the current programmed status for any of the three integrated timers.

ALT:

The ALT bit determines which of two MAX COUNT registers is used for count comparison. If ALT = 0, register A for that timer is always used, while if ALT = 1, the comparison will alternate between register A and register B when each maximum count is reached. This alternation allows the user to change one MAX COUNT register while the other is being used, and thus provides a method of generating non-repetitive waveforms. Square waves and pulse outputs of any duty cycle are a subset of available signals obtained by not changing the final count registers. The ALT bit also determines the function of the timer output pin. If ALT is zero, the output pin will go LOW for one clock, the clock after the maximum count is reached. If ALT is one, the output pin will reflect the current MAX COUNT register being used (0/1 for B/A).

Table H-9. Timer Control Block Format

Register Name	Register Offset		
	Tmr. 0	Tmr. 1	Tmr. 2
Mode/Control Word	56H	5EH	66H
Max Count B	54H	5CH	not present
Max Count A	52H	5AH	62H
Count Register	50H	58H	60H

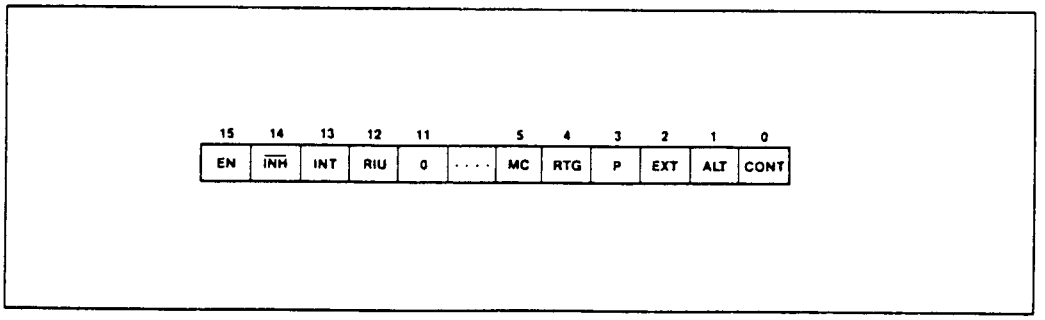


Figure H-12. Timer Mode/Control Register

CONT:

Setting the CONT bit causes the associated timer to run continuously, while resetting it causes the timer to halt upon maximum count. If CONT = 0 and ALT = 1, the timer will count to the MAX COUNT register A value, reset, count to the register B value, reset, and halt.

EXT:

The external bit selects between internal and external clocking for the timer. The external signal may be asynchronous with respect to the 80188 clock. If this bit is set, the timer will count LOW-to-HIGH transitions on the input pin. If cleared, it will count an internal clock while using the input pin for control. In this mode, the function of the external pin is defined by the RTG bit. The maximum input to output transition latency time may be as much as 6 clocks. However, clock inputs may be pipelined as closely together as every 4 clocks without losing clock pulses.

P:

The prescaler bit is ignored unless internal clocking has been selected (EXT = 0). If the P bit is a zero, the timer will count at one-fourth the internal CPU clock rate. If the P bit is a one, the output of timer 2 will be used as a clock for the timer. Note that the user must initialize and start timer 2 to obtain the prescaled clock.

RTG:

Retrigger bit is only active for internal clocking (EXT = 0). In this case it determines the control function provided by the input pin.

If RTG = 0, the input level gates the internal clock on and off. If the input pin is HIGH, the timer will count; if the input pin is LOW, the timer will hold its value. As indicated previously, the input signal may be asynchronous with respect to the 80188 clock.

When RTG = 1, the input pin detects LOW-to-HIGH transitions. The first such transition starts the timer running, clearing the timer value to zero on the first clock, and the incrementing thereafter. Further transitions on the input pin will again reset the timer to zero, from which it will start counting up again. If CONT = 0, when the timer has reached maximum count, the EN bit will be cleared, inhibiting further timer activity.

EN:

The enable bit provides programmer control over the timer's RUN/HALT status. When set, the timer is enabled to increment subject to the input pin constraints in the internal clock mode (discussed previously). When cleared, the timer will be inhibited from counting. All input pin transitions during the time EN is zero will be ignored. If CONT is zero, the EN bit is automatically cleared upon maximum count.

$\overline{\text{INH}}$:

The inhibit bit allows for selective updating of the enable (EN) bit. If $\overline{\text{INH}}$ is a one during the write to the mode/control word, then the state of the EN bit will be modified by the write. If $\overline{\text{INH}}$ is a zero during the write, the EN bit will be unaffected by the operation. This bit is not stored; it will always be a 0 on a read.

INT:

When set, the INT bit enables interrupts from the timer, which will be generated on every terminal count. If the timer is configured in dual MAX COUNT register mode, an interrupt will be generated each time the value in MAX COUNT register A is reached, and each time the value in MAX COUNT register B is reached. If this enable bit is cleared after the interrupt request has been generated, but before a pending interrupt is serviced, the interrupt request will still be in force. (The request is latched in the Interrupt Controller.)

MC:

The Maximum Count bit is set whenever the timer reaches its final maximum count value. If the timer is configured in dual MAX COUNT register mode, this bit will be set each time the value in MAX COUNT register A is reached, and each time the value in MAX COUNT register B is reached. This bit is set regardless of the timer's interrupt-enable bit. The MC bit gives the user the ability to monitor timer status through software instead of through interrupts.

RIU:

The Register In Use bit indicates which MAX COUNT register is currently being used for comparison to the timer count value. A zero value indicates register A. The RIU bit cannot be written, i.e., its value is not affected when the control register is written. It is always cleared when the ALT bit is zero.

Not all mode bits are provided for timer 2. Certain bits are hardwired as indicated below:

ALT = 0, EXT = 0, P = 0, RIU = 0

Count Registers

Each of the three timers has a 16-bit count register. The current contents of this register may be read or written by the processor at any time. If the register is written into while the timer is counting, the new value will take effect in the current count cycle.

Max Count Registers

Timers 0 and 1 have two MAX COUNT registers, while timer 2 has a single MAX COUNT register. These contain the number of events the timer will count. In timers 0 and 1, the MAX COUNT register used can alternate between the two max count values whenever the current maximum count is reached. The condition which causes a timer to reset is equivalent between the current count value and the max count being used. This means that if the count is changed to be above the max count value, or if the max count value is changed to be below the current value, the timer will not reset to zero, but rather will count to its maximum value, "wrap around" to zero, then count until the max count is reached.

Timers and Reset

Upon RESET, the Timers will perform the following actions:

- All EN (Enable) bits are reset preventing timer counting.
- All SEL (Select) bits are reset to zero. This selects MAX COUNT register A, resulting in the Timer Out pins going HIGH upon RESET.

Interrupt Controller

The 80188 can receive interrupts from a number of sources, both internal and external. The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU.

Internal interrupt sources (Timers and DMA channels) can be disabled by their own control registers or by mask bits within the interrupt controller. The 80188 interrupts controller has its own control registers that set the mode of operation for the controller.

The interrupt controller will resolve priority among requests that are pending simultaneously. Nesting is provided so interrupt service routines for lower priority interrupts may themselves be interrupted by higher priority interrupts. A block diagram of the interrupt controller is shown in Figure H-13.

The interrupt controller has a special iRMX 86 compatibility mode that allows the use of the 80188 within the iRMX 86 operating system interrupt structure. The controller is set in this mode by setting bit 14 in the peripheral control block relocation register (see iRMX 86 Compatibility Mode section). In this mode, the internal 80188 interrupt controller functions as a "slave" controller to an external "master" controller. Special initialization software must be included to properly set up the 80188 interrupt controller in iRMX 86 mode.

NON-iRMX MODE OPERATION

Interrupt Controller External Interface

For external interrupt sources, five dedicated pins are provided. One of these pins is dedicated to NMI, non-maskable interrupt. This is typically used for power-fail interrupts, etc. The other four pins may function either as four interrupt input lines with internally generated interrupt vectors, as an interrupt line and an interrupt acknowledge line (called the "cascade mode") along with two other input lines with internally generated interrupt vectors, or as two interrupt input lines and two dedicated interrupt acknowledge output lines. When the interrupt lines are configured in cascade mode, the 80188 interrupt controller will not generate internal interrupt vectors.

External sources in the cascade mode use externally generated interrupt vectors. When an interrupt is acknowledge, two INTA cycles are initiated and the vector is read into the 80188 on the second cycle. The capability to interface to external 8259A programmable interrupt controllers is thus provided when the inputs are configured in cascade mode.

Interrupt Controller Modes of Operation

The basic modes of operation of the interrupt controller in non-iRMX mode are similar to the 8259A. The interrupt controller responds identically to internal interrupts in all three modes: the difference is only in the interpretation of function of the four external interrupt pins. The interrupt controller is set into one of these three modes by programming the correct bits in the INT0 and INT1 control registers. The modes of interrupt controller operation are as follows:

Fully Nested Mode

When in the fully nested mode four pins are used as direct interrupt requests. The vectors for these four inputs are generated internally. An in-service bit is provided for

every interrupt source. If a lower-priority device requests an interrupt while the in-service bit (IS) is set, no interrupt will be generated by the interrupt controller. In addition, if another interrupt request occurs from the same interrupt source while the in-service bit is set, no interrupt will be generated by the interrupt controller. This allows interrupt service routines to operate with interrupts enabled without being themselves interrupted by lower-priority interrupts. Since interrupts are enabled, higher-priority interrupts will be serviced.

When a service routine is completed, the prior IS bit must be reset by writing the proper pattern to the EOI register. This is required to allow subsequent interrupts from this interrupt source and to allow servicing of lower-priority interrupts. An EOI command is issued at the end of the service routine just before the issuance of the return from interrupt instruction. If the fully nested structure has been upheld, the next highest-priority source with its IS bit set is then serviced.

Cascade Mode

The 80188 has four interrupt pins and two of them have dual functions. In the fully nested mode the four pins are used as direct interrupt inputs and the corresponding vectors are generated internally. In the cascade mode, the four pins are configured into interrupt input-dedicated acknowledge signal pairs. The interconnection is shown in Figure H-14. INTO is an interrupt input interfaced to an 8259A, while INT2/$\overline{\text{INTA0}}$ serves as the dedicated interrupt acknowledge signal to that peripheral. The same is true for INT1 and INT3($\overline{\text{INTA1}}$. Each pair can selectively be placed in the cascade or non-cascade mode by programming the proper value into INT0 and INT1 control registers. The use of the dedicated acknowledge signals eliminates the need for the use of external logic to generate $\overline{\text{INTA}}$ and device select signals.

The primary cascade mode allows the capability to serve up to 128 external interrupt sources through the use of external master and slave 8259As. Three levels of priority are created, requiring priority resolution in the 80188 interrupt controller, the master 8259As, and the slave 8259As. If an external interrupt is serviced, one IS bit is set at each of these levels. When the interrupt service routine is completed, up to three end-of-interrupt commands must be issued by the programmer.

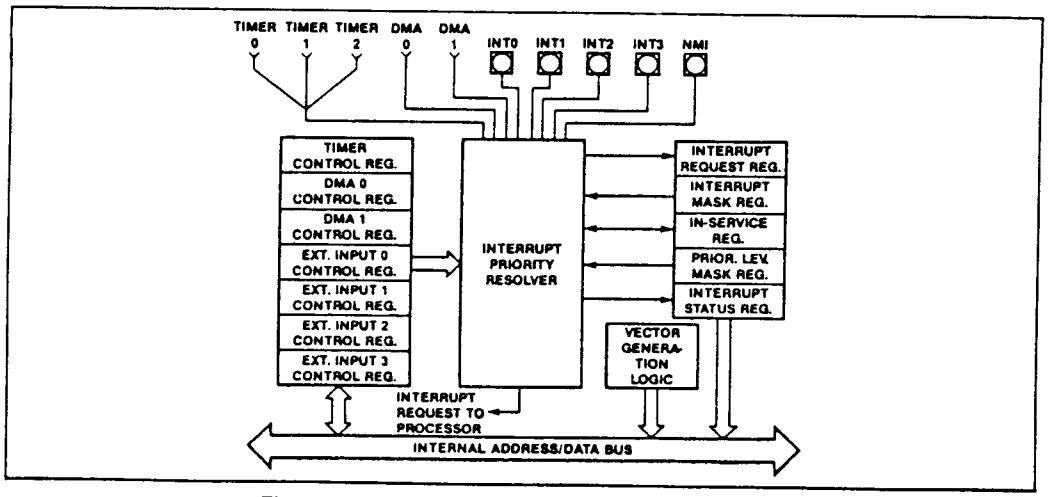


Figure H-13. Interrupt Controller Block Diagram

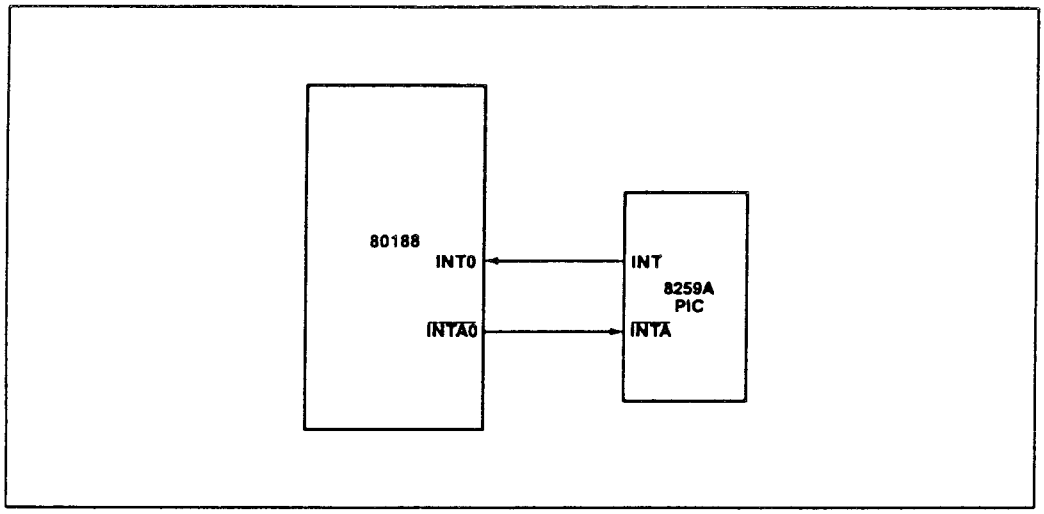


Figure H-14. Cascade Mode Interrupt Connection

Special Fully Nested Mode

This mode is entered by setting the SFNM bit in INT0 or INT1 control register. It enables complete nestability with external 8259A masters. Normally, an interrupt request from an interrupt source will not be recognized unless the in-service bit for that source is reset. If more than one interrupt source is connected to an external interrupt controller, all of the interrupts will be funneled through the same 80188 interrupt request pin. As a result, if the external interrupt controller receives a higher-priority interrupt, its interrupt will not be recognized by the 80188 controller until the 80188 in-service bit is reset. In special fully nested mode, the 80188 interrupt controller will allow interrupts from an external pin regardless of the state of the in-service bit for an interrupt source in order to allow multiple interrupts from a single pin. An in-service bit will continue to be set, however, to inhibit interrupts from other lower-priority 80188 interrupt sources.

Special procedures should be followed when reseting IS bits at the end of interrupt service routines. Software polling of the external master's IS register is required to determine if there is more than one bit set. If so, the IS bit in the 80188 remains active and the next interrupt service routine is entered.

Operation in a Polled Environment

The controller may be used in a polled mode if interrupts are undesirable. When polling, the processor disables interrupts and then polls the interrupt controller whenever it is convenient. Polling the interrupt controller is accomplished by reading the Poll Word (Figure H-1). Bit 15 is in the poll word indicates to the processor that an interrupt of high enough priority is requesting service. Reading the Poll Word causes the In-Service bit of the highest-priority source to be set.

It is desirable to be able to read the Poll Word information without guaranteeing service of any pending interrupt, i.e., not set the indicated in-service bit. The 80188 provides a Poll Status Word in addition to the conventional Poll Word to allow this to be done. Poll Word information is duplicated in the Poll Status Word, but reading the Poll Status Word does not set the associated in-service bit. These words are located in two adjacent memory locations in the register file.

Non-iRMX Mode Features

Programmable Priority

The user can program the interrupt sources into any of eight different priority levels. The programming is done by placing a 3-bit priority level (0-7) in the control register of each interrupt source. (A source with a priority level of 4 has higher priority over all priority levels from 5 to 7. Priority registers containing values lower than 4 have greater priority.) All interrupt sources have preprogrammed default priority levels (see Table H-10).

If two requests with the same programmed priority level are pending at once, the priority ordering scheme shown in Table H-10 is used. If the serviced interrupt routine reenables interrupts, it allows other requests to be serviced.

End-of-Interrupt Command

The end-of-interrupt (EOI) command is used by the programmer to reset the In-Service (IS) bit when an interrupt service routine is completed. The EOI command is issued by writing the proper pattern to the EOI register. There are two types of EOI commands, specific and nonspecific. The nonspecific command does not specify which IS bit is reset. When issued, the interrupt controller automatically resets the IS bit of the highest priority source with an active service routine. A specific EOI command requires that the programmer send the interrupt vector type to the interrupt controller indicating which source's IS bit is to be reset. This command is used when the fully nested structure has been disturbed or the highest priority IS bit that was set does not belong to the service routine in progress.

Trigger Mode

The four external interrupt pins can be programmed in either edge- or level-trigger mode. The control register for each external source has a level-trigger mode (LTM) bit. All interrupt inputs are active HIGH. In the edge sense mode or the level-trigger mode, the interrupt request must remain active (HIGH) until the interrupt request is acknowledged by the 80188 CPU. In the edge-sense mode, if the level remains high after the interrupt is acknowledged, the input is disabled and no further requests will be generated. The input level must go LOW for at least one clock cycle to reenable the input. In the level-trigger mode, no such provision is made: holding the interrupt input HIGH will cause continuous interrupt requests.

Interrupt Vectoring

The 80188 Interrupt Controller will generate interrupt vectors for the integrated DMA channels and the integrated Timers. In addition, the Interrupt Controller will generate interrupt vectors for the external interrupt lines if they are not configured in Cascade or Special Fully Nested Mode. The interrupt vectors generated are fixed and cannot be changed (see Table H-10).

Interrupt Controller Registers

The Interrupt Controller register model is shown in Figure H-15. It contains 15 registers. All registers can both be read or written unless specified otherwise.

In-Service Register

This register can be read from or written into. The format is shown in Figure H-16. It contains the In-Service bit for each of the interrupt sources. The In-Service bit is set to indicate that a source's service routine is in progress. When an In-Service bit is set, the

interrupt controller will not generate interrupts to the CPU when it receives interrupt requests from devices with a lower programmed priority level. The TMR bit is the In-Service bit for all three timers; the D0 and D1 bits are the In-Service bits for the two DMA channels; the I0-I3 are the In-Service bits for the external interrupt pins. The IS bit is set when the processor acknowledges an interrupt request either by an interrupt acknowledge or by reading the poll register. The IS bit is reset at the end of the interrupt service routine by an end-of-interrupt command issued by the CPU.

Interrupt Request Register

The internal interrupt sources have interrupt request bits inside the interrupt controller. The format of this register is shown in Figure H-16. A read from this register yields the status of these bits. The TMR bit is the logical OR of all timer interrupt requests. D0 and D1 are the interrupt request bits for the DMA channels.

The state of the external interrupt input pins is also indicated. The state of the external interrupt pins is not a stored condition inside the interrupt controller, therefore the external interrupt bits cannot be written. The external interrupt request bits show exactly when an interrupt request is given to the interrupt controller, so if edge-triggered mode is selected, the bit in the register will be HIGH only after an active-to-active transition. For internal interrupt sources, the register bits are set when a request arrives and are reset when the processor acknowledges the requests.

Table H-10. 80188 Interrupt Vectors

Interrupt Name	Vector Type	Default Priority	Related Instructions
Divide Error Exception	0	*1	DIV, IDIV
Single Step Interrupt	1	12**2	All
NMI	2	1	All
Breakpoint Interrupt	3	*1	INT
INT0 Detected Overflow Exception	4	*1	INT0
Array Bounds Exception	5	*1	BOUND
Unused-Opcode Exception	6	*1	Undefined Opcodes
ESC Opcode Exception	7	*1***	ESC Opcodes
Timer 0 Interrupt	8	2A****	
Timer 1 Interrupt	18	2B****	
Timer 2 Interrupt	19	2C****	
Reserved	9	3	
DMA 0 Interrupt	10	4	
DMA 1 Interrupt	11	5	
INT0 Interrupt	12	6	
INT1 Interrupt	13	7	
INT2 Interrupt	14	8	
INT3 Interrupt	15	9	

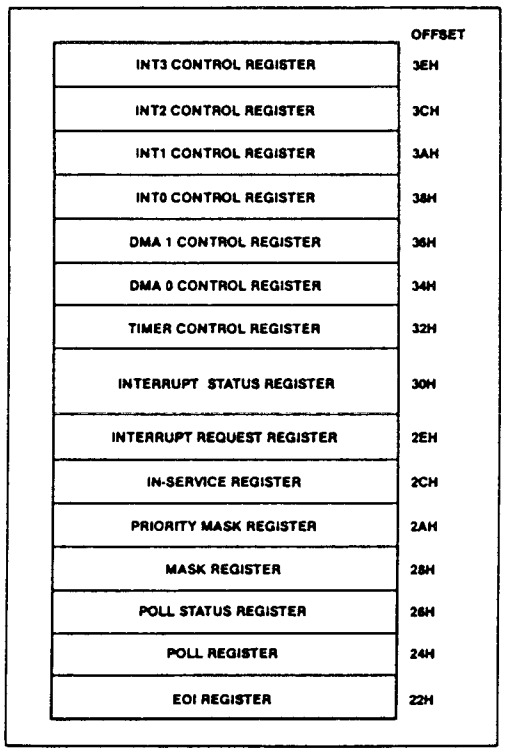


Figure H-15. Interrupt Controller Registers (Non-iRMX 86 Mode)

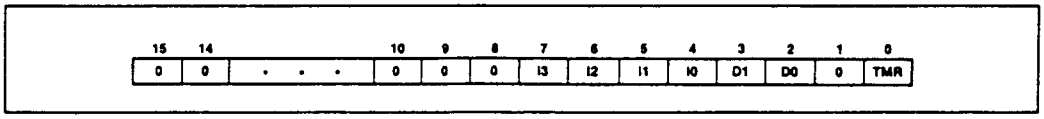


Figure H-16. In-Service, Interrupt Request, and Mask Register Formats

Mask Register

This is a 16-bit register that contains a mask bit for each interrupt source. The format for this register is shown in Figure H-16. A one in a bit position corresponding to a particular source serves to mask the source from generating interrupts. These mask bits are the exact same bits which are used in the individual control registers; programming a mask bit using the mask register will also change this bit in the individual control registers, and vice versa.

Priority Mask Register

This register is used to mask all interrupts below particular interrupt priority levels. The format of this register is shown in Figure H-17. The code in the lower three bits of this register inhibits interrupts of priority lower (a higher priority number) than the specified. For example, 100 written into this register masks interrupts of level five (101), six (110), and seven (111). The register is reset to seven (111) upon RESET so all interrupts are unmasked.

Interrupt Status Register

This register contains general interrupt controller status information. The format of this register is shown in Figure H-18. The bits in the status register have the following functions:

DHLT: DMA Halt Transfer; setting this bit halts all DMA transfers. It is automatically set whenever a non-maskable interrupt occurs, and it is reset when an IRET instruction is executed. The purpose of this bit is to allow prompt service of all non-maskable interrupts. This bit may also be set by the CPU.

IRTx: These three bits represent the individual timer interrupt request bits. These bits are used to differentiate the timer interrupts, since the timer IR bit in the interrupt request register is the "OR" function of all timer interrupt requests. Note that setting any one of these three bits initiates an interrupt request to the interrupt controller.

Timer, DMA 0, 1; Control Registers

These registers are the control words for all the internal interrupt sources. The format for these registers is shown in Figure H-19. The three bit positions PR0, PR1, and PR2 represent the programmable priority level of the interrupt source. The MSK bit inhibits interrupt requests from the interrupt source. The MSK bits in the individual control registers are the exact same bits as are in the Mask Register; modifying them in the individual control registers will also modify them in the Mask Register, and vice versa.

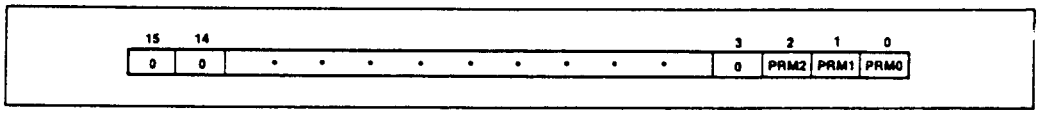


Figure H-17. Priority Mask Register Format

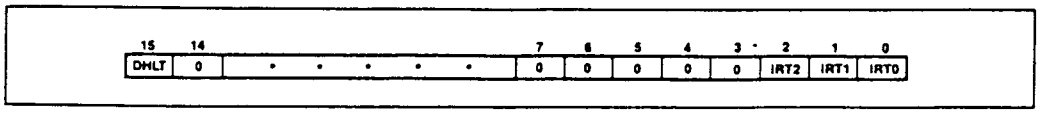


Figure H-18. Interrupt Status Register Format

INT0-INT3 Control Registers

These registers are the control words for the four external input pins. Figure H-20 shows the formal of the INT0 and INT1 Control registers; Figure H-21 shows the format of the INT2 and INT3 Control registers. In cascade mode or special fully nested mode, the control words for INT2 and INT3 are not used.

The bits in the various control registers are encoded as follows:

PRO-2: Priority programming information. Highest priority = 000, lowest priority = 111.

LTM: Level-trigger mode bit. 1 = level-triggered; 0 = edge-triggered. Interrupt Input levels are active high. In level-triggered mode, an interrupt is generated whenever the external line is high. In edge-triggered mode, an interrupt will be generated only when this level is preceded by an inactive-to-active transition on the line. In both cases, the level must remain active until the interrupt is acknowledged.

MSK: Mask bit, 1 = mask; 0 = nonmask.

C: Cascade mode bit, 1 = cascade; 0 = direct.

SFNM: Special fully nested mode bit, 1 = SFNM; 0 = normal nested mode.

EOI Register

The end of the interrupt register is a command register which can only be written into. The format of this register is shown in Figure H-22. It initiates an EOI command when written to by the 80188 CPU.

The bits in the EOI register are encoded as follows:

Sx: Encoded information that specifies an interrupt source vector type as shown in Table H-10. For example, to reset the In-Service bit for DMA channel 0, these bits should be set to 01010, since the vector type for DMA channel 0 is 10. Note that to reset the single In-Service bit for any of the three timers, the vector type for timer 0 (8) should be written in this register.

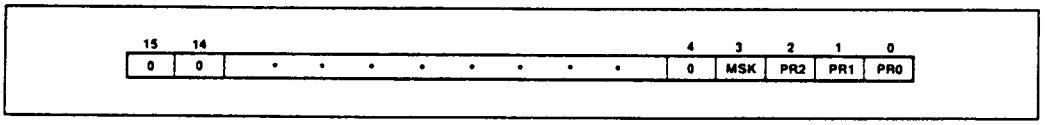


Figure H-19. Timer/DMA Register Formats

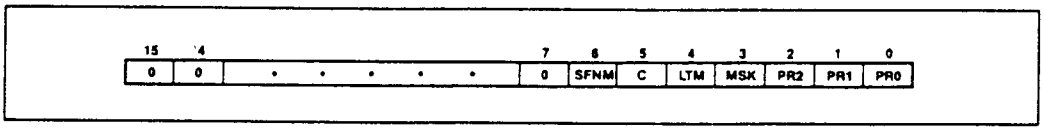


Figure H-20. INT0/INT1 Register Formats

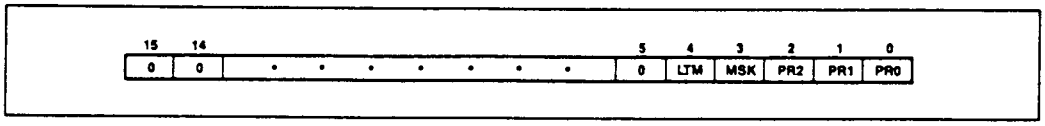


Figure H-21. INT2/INT3 Register Formats

NSPEC/SPEC: A bit that determines the type of EOI command. Nonspecific = 1, Specific = 0.

Poll and Poll Status Registers

These registers contain polling information. The format of these registers is shown in Figure H-23. They can only be read. Reading the Poll register constitutes a software poll. This will set the IS bit of the highest priority pending interrupt. Reading the poll status register will not set the IS bit of the highest priority pending interrupt; only the status of pending interrupts will be provided.

Encoding of the Poll and Poll Status register bits are as follows:

Sx: Encoded information that indicates the vector type of the highest priority interrupting source. Valid only when INTREQ = 1.

INTREQ: This bit determines if an interrupt request is present. Interrupt Request = 1; no Interrupt Request = 0.

iRMX 86 COMPATIBILITY MODE

This mode allows iRMX 86-80188 compatibility. The interrupt model of iRMX 86 requires one master and multiple slave 8259As in cascaded fashion. When iRMX mode is used, the internal 80188 interrupt controller will be used as a slave controller to an external master interrupt controller. The internal 80188 resources will be monitored through the internal interrupt controller, while the external controller functions as the system master interrupt controller.

Upon reset, the 80188 interrupt controller will be in the non-iRMX 86 mode of operation. To set the controller in the iRMX 86 mode, bit 14 of the Relocation Register should be set.

Because of pin limitations caused by the need to interface to an external 8259A master, the internal interrupt controller will no longer accept external inputs. There are however, enough 80188 interrupt controller inputs (internally) to dedicate one to each timer. In this mode, each timer interrupt source has its own mask bit, IS bit, and control word.

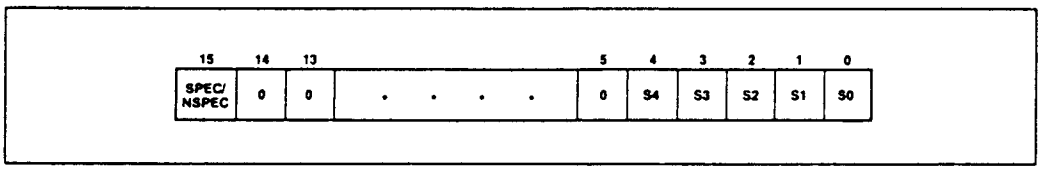


Figure H-22. EOI Register Format

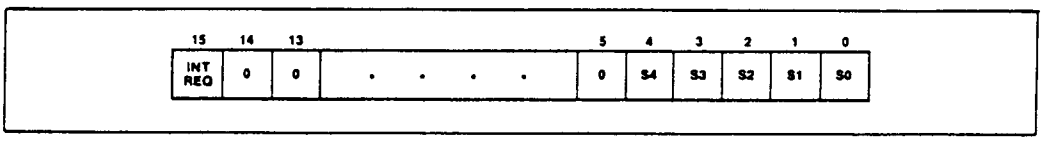


Figure H-23. Poll Register Format

The iRMX 86 operating system requires peripherals to be assigned fixed priority levels. This is incompatible with the normal operation of the 80188 interrupt controller. Therefore, the initialization software must program the proper priority levels for each source. The required priority levels for the internal interrupt sources in iRMX mode are shown in Table H-11.

These level assignments must remain fixed in the iRMX 86 mode of operation.

iRMX 86 Mode External Interface

This configuration of the 80188 with respect to an external 8259A master is shown in Figure H-24. The INT0 input is used as the 80188 CPU interrupt input. INT3 functions as an output to send the 80188 slave-interrupt-request to one of the 8 master -PIC-inputs.

Correct master-slave interface requires decoding of the slave addresses (CAS0-2). Slave 8259As do this internally. Because of pin limitations, the 80188 slave address will have to be decoded externally. INT1 is used as a slave-select input. Note that the slave vector address is transferred internally, but the READY input must be supplied externally.

$\overline{\text{INT2}}$ is used as an acknowledge output, suitable to drive the $\overline{\text{INTA}}$ input of an 8259A.

Interrupt Nesting

iRMX 86 mode operation allows nesting of interrupt requests. When an interrupt is acknowledged, the priority logic masks off all priority levels except those with equal or higher priority.

Table H-11. Internal Source Priority Level

Priority Level	Interrupt Source
0	Timer 0
1	(reserved)
2	DMA 0
3	DMA 1
4	Timer 1
5	Timer 2

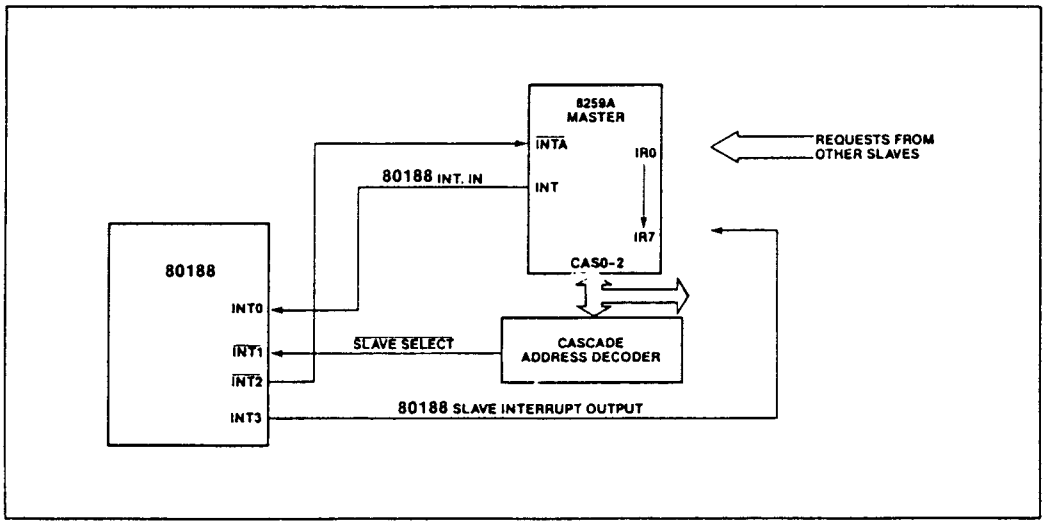


Figure H-24. iRMX 86 Interrupt Controller Interconnection

Vector Generation in the iRMX 86 Mode

Vector generation in iRMX mode is exactly like that of an 8259A slave. The interrupt controller generates an 8-bit vector which the CPU multiplies by four and uses as an address into a vector table. The significant five bits of the vector are user-programmable while the lower three bits are generated by the priority logic. These bits represent the encoding of the priority level requesting service. The significant five bits of the vector are programmed by writing to the Interrupt Vector register at F0 0122.

Specific End-of-Interrupt

In iRMX mode the specific EOI command operates to reset an in-service bit of a specific priority. The user supplies a 3-bit priority-level value that points to an in-service bit to be reset. The command is executed by writing the correct value in the Specific EOI register at F0 0122.

Interrupt Controller Registers in the iRMX 86 Mode

All control and command registers are located inside the internal peripheral control block. Figure H-25 shows the offsets of these registers.

End-of-Interrupt Registers

The end-of-interrupt register is a command register which can only be written. The format of this register is shown in Figure H-26. It initiates an EOI command when written by the 80188 CPU.

The bits in the EOI register are encoded as follows:

Lx: Encoded value indicating the priority of the IS bit to be reset.

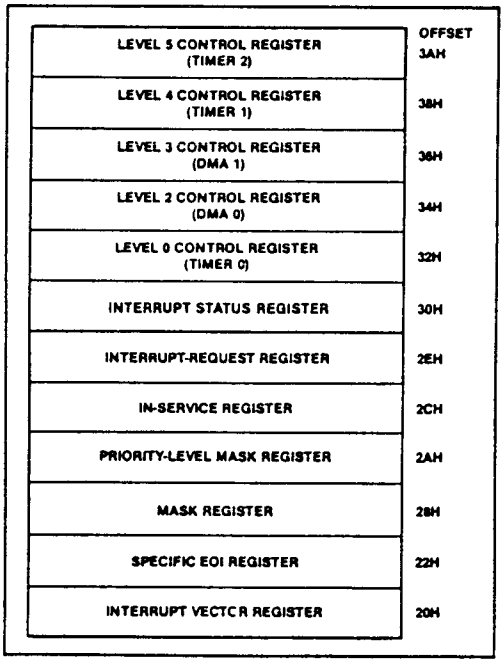


Figure H-25. Interrupt Controller Registers (iRMX 86 Mode)

In-Service Register

This register can be read from or written into. It contains the in-service bit for each of the internal interrupt sources. The format for this register is shown in Figure H-27. Bit positions 2 and 3 correspond to the DMA channels; positions 0, 4, and 5 correspond to the integral timers. The source's IS bit is set when the processor acknowledges its interrupt request.

Interrupt Request Register

This register indicates which internal peripherals have interrupt requests pending. The format of this register is shown in Figure H-27. The interrupt request bits are set when a request arrives from an internal source, and are reset when the processor acknowledges the request.

Mark Register

This register contains a mask bit for each interrupt source. The format for this register is shown in Figure H-27. If the bit in this register corresponding to a particular interrupt source is set, any interrupts from that source will be masked. These mask bits are exactly the same bits which are used in the individual control registers, i.e., changing the state of a mask bit in this register will also change the state of the mask bit in the individual interrupt control register corresponding to the bit.

Control Registers

These registers are the control words for all the internal interrupt sources. The format of these registers is shown in Figure H-28. Each of the timers and both of the DMA channels have their own Control Register.

The bits of the Control Registers are encoded as follows:

prx: 3-bit encoded field indicating a priority level for the source; note that each source must be programmed at specified levels.

msk: mask bit for the priority level indicated by prx bits.

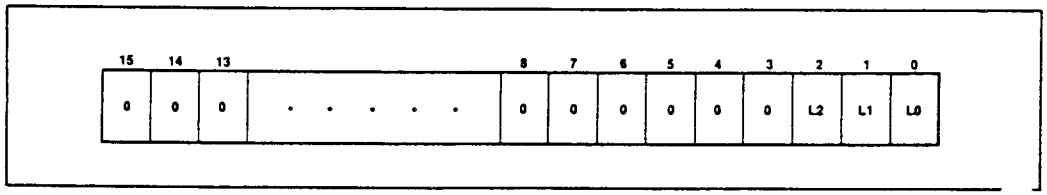


Figure H-26. Specific EOI Register Format

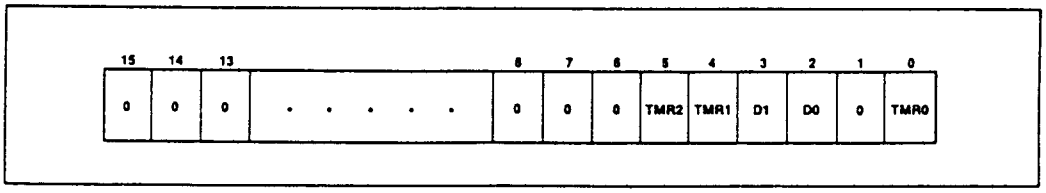


Figure H-27. In-Service, Interrupt Request, and Mask Register Format

Interrupt Vector Register

This register provides the upper five bits of the interrupt vector address. The format of this register is shown in Figure H-29. The interrupt controller itself provides the lower three bits of the interrupt vector as determined by the priority level of the interrupt request.

The format of the bits in this register is:

tx: 5-bit field indicating the upper five bits of the vector address.

Priority-Level Mask Register

This register indicates the lowest priority-level interrupt which will be serviced.

The encoding of the bits in this register is:

mx: 3-bit encoded field indication priority-level value. All levels of lower priority will be masked.

Interrupt Controller and Reset

Upon RESET, the interrupt controller will perform the following actions:

- All SFNM bits reset to 0, implying Fully Nested Mode.
- All PR bits in the various control registers set to 1. This places all sources at lowest priority (level 111).
- All LTM bits reset to 0, resulting in edge-sense mode.
- All Interrupt Service bits reset to 0.
- All Interrupt Request bits reset to 0.
- All MSK (Interrupt Mask) bits set to 1 (mask).
- All C (Cascade) bits reset to 0 (non-cascade).
- All PRM (Priority Mask) bits set to 1, implying no levels masked.
- Initialized to non-iRMX 86 mode.

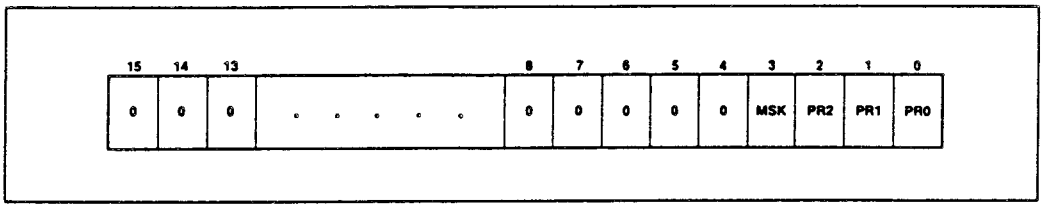


Figure H-28. Control Word Format

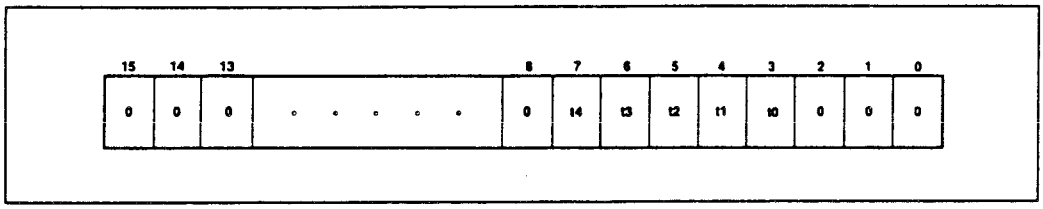


Figure H-29. Interrupt Vector Register Format

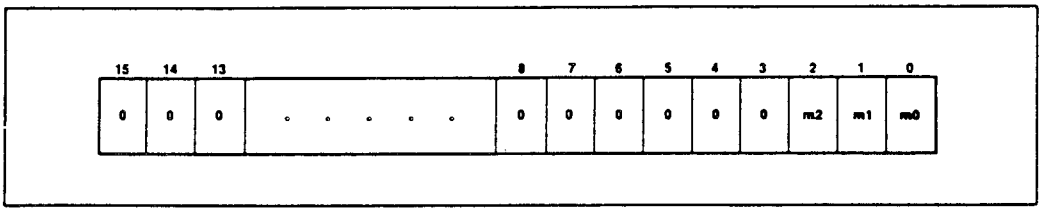


Figure H-30. Priority Level Mask Register

Appendix I
PCB Recovery Tape

INTRODUCTION I-1.

The Pod has a special feature that allows it to save the contents of the Peripheral Control Block (PCB) registers after a RUNUUT operation. This feature, which is accomplished by exiting from RUNUUT in a special way that doesn't reset the Pod's microprocessor, allows you to recover PCB values from a known-good UUT.

This method is an easy and reliable way to establish correct values for a UUT's PCB registers prior to troubleshooting. These known-good values are created by the UUT's own programming, and are used as standard values for troubleshooting other UUTs of the same type.

The procedure for automatically performing the special RUNUUT operation to recover PCB registers is on the tape that is included with the Pod. A 9010A Language Compiler source listing of the program is included in this Appendix. A description of the internal operation of the special RUNUUT function and directions for using the program follow.

SPECIAL RUNUUT I-2.

This soft exit uses the Non-Maskable Interrupt (NMI) and switches the Pod's buffers from the UUT's bus to the Pod's internal bus. This allows the Pod to recover the UUT's PCB register contents and save them in a series of Pod Function addresses in the range F003XX. This range corresponds to the normal F001XX range used by the Pod for the Peripheral Control Block (see Table I-1).

The Pod is configured by the tape to use this special exit from RUNUUT by using a WRITE @ F0 0006=1 operation before the RUNUUT is started. Then, after the UUT's program has initialized all the PCB registers, $\overline{\text{MAINSTAT}}$ is asserted low to initiate the bus switch and NMI.

A READ @ F0 0006 after RUNUUT is completed returns information about whether or not the PCB registers were successfully recovered. The significant data in F0 0006 is as follows:

Bit 1 — Bit 1 is set if the NMI routine was properly executed. If bit 1 is not set, either the UUT had already entered its NMI routine, or the processor was in a HOLD or Wait state.

Table I-1. Recovered Peripheral Control Block Special Addresses

RECOVERED FROM UUT	POD'S ADDRESS	BYTE (H=High L=Low)	DESCRIPTION	POWER-UP DEFAULT VALUE (hex)
Relocation Register Address				
F0 03FF	F0 01FF	H	Relocation Register	00
F0 03FE	F0 01FE	L		FF
DMA Controller Register Addresses				
F0 03DB	F0 01DB	H	DMA Channel 1 Control Word	00
F0 03DA	F0 01DA	L		00
F0 03D9	F0 01D9	H	DMA Channel 1 Transer Count	00
F0 03D8	F0 01D8	L		00
F0 03D6	F0 01D6		DMA Channel 1 Dest. Ptr. (upr 4 bits)	00
F0 03D5	F0 01D5		(mdl 8 bits)	00
F0 03D4	F0 01D4		(lwr 8 bits)	00
F0 03D2	F0 01D2		DMA Channel 1 Src. Ptr. (upr 4 bits)	00
F0 03D1	F0 01D1		(mdl 8 bits)	00
F0 03D0	F0 01D0		(lwr 8 bits)	00
F0 03CB	F0 01CB	H	DMA Channel 0 Control Word	00
F0 03CA	F0 01CA	L		00
F0 03C9	F0 01C9	H	DMA Channel 0 Transer Count	00
F0 03C8	F0 01C8	L		00
F0 03C6	F0 01C6		DMA Channel 0 Dest. Ptr. (upr 4 bits)	00
F0 03C5	F0 01C5		(mdl 8 bits)	00
F0 03C4	F0 01C5		(lwr 8 bits)	00
F0 03C2	F0 01C2		DMA Channel 0 Src. Ptr. (upr 4 bits)	00
F0 03C1	F0 01C1		(mdl 8 bits)	00
F0 03C0	F0 01C0		(lwr 8 bits)	00
Chip Select Register Addresses				
F0 03A9	F0 01A9	H	MPCS Register	A0
F0 03A8	F0 01A8	L		FB
F0 03A7	F0 01A7	H	MMCS Register	81
F0 03A6	F0 01A6	L		FB
F0 03A5	F0 01A5	H	PACS Register	40
F0 03A4	F0 01A4	L		3B
F0 03A3	F0 01A3	H	LMCS Register	3F
F0 03A2	F0 01A2	L		FB
F0 03A1	F0 01A1	H	UMCS Register	C0
F0 03A0	F0 01A0	L		3B
Timer Register Addresses				
F0 0367	F0 0167	H	Timer 2 Mode/Control Word Register	00
F0 0366	F0 0166	L		00
F0 0363	F0 0163	H	Timer 2 Max Count A Register	00
F0 0362	F0 0162	L		00
F0 0361	F0 0161	H	Timer 2 Count Register	00
F0 0360	F0 0160	L		00
F0 035F	F0 015F	H	Timer 1 Mode/Control Word Register	00
F0 035E	F0 015E	L		00
F0 035D	F0 015D	H	Timer 1 Max Count B Register	00
F0 035C	F0 015C	L		00

Table I-1. Recovered Peripheral Control Block Special Addresses (cont)

RECOVERED FROM UUT	POD'S ADDRESS	BYTE (H=High L=Low)	DESCRIPTION	POWER-UP DEFAULT VALUE (hex)
F0 035B	F0 015B	H	Timer 1 Max Count A Register	00
F0 035A	F0 015A	L		00
F0 0359	F0 0159	H	Timer 1 Count Register	00
F0 0358	F0 0158	L		00
F0 0357	F0 0157	H	Timer 0 Mode/Control Word Register	00
F0 0356	F0 0156	L		00
F0 0355	F0 0155	H	Timer 0 Max Count B Register	00
F0 0354	F0 0154	L		00
F0 0353	F0 0153	H	Timer 0 Max Count A Register	00
F0 0352	F0 0152	L		00
F0 0351	F0 0151	H	Timer 0 Count Register	00
F0 0350	F0 0150	L		00
Interrupt Controller Register Addresses				
F0 033F	F0 013F	H	INT3 Control Register (master mode)	00
F0 033E	F0 013E	L	Not used in iRMX mode	0F
F0 033D	F0 013D	H	INT2 Control Register (master mode)	00
F0 033C	F0 013C	L	Not used in iRMX mode	0F
F0 033B	F0 013B	H	INT1 Control Register (master mode)	00
F0 033A	F0 013A	L	Timer 2 Control Register (iRMX mode)	0F
F0 0339	F0 0139	H	INT0 Control Register (master mode)	00
F0 0338	F0 0138	L	Timer 1 Control Register (iRMX mode)	0F
F0 0337	F0 0137	H	DMA 1 Control Register (both modes)	00
F0 0336	F0 0136	L		0F
F0 0335	F0 0135	H	DMA 0 Control Register (both modes)	00
F0 0334	F0 0134	L		0F
F0 0333	F0 0133	H	Timer Control Register (master mode)	00
F0 0332	F0 0132	L	Timer 0 Control Register (iRMX mode)	0F
F0 0331	F0 0131	H	Int. Cont. Status Reg. (both modes)	80
F0 0330	F0 0130	L		00
F0 032F	F0 012F	H	Int. Request Register (both modes)	00
F0 032E	F0 012E	L		00
F0 032D	F0 012D	H	In-Service Register (both modes)	00
F0 032C	F0 012C	L		00
F0 032B	F0 012B	H	Priority Mask Register (both modes)	00
F0 032A	F0 012A	L		07
F0 0329	F0 0129	H	Mask Register (both modes)	00
F0 0328	F0 0128	L		FD
F0 0327	n/a	H	Poll Status Register (master mode)	00
F0 0326	n/a	L	Not used in iRMX mode	00

Table I-1. Recovered Peripheral Control Block Special Addresses (cont)

RECOVERED FROM UUT	POD'S ADDRESS	BYTE (H=High L=Low)	DESCRIPTION	POWER-UP DEFAULT VALUE (hex)
F0 0325	n/a	H	Poll Status Register (master mode)	00
F0 0324	n/a	L	Not used in iRMX mode	00
F0 0323	F0 0123	H	EOI Register (master mode)	00
F0 0322	F0 0122	L	Specific EOI Register (iRMX mode)	00
F0 0321	F0 0121	H	Int. Vector Register (iRMX mode)	00
F0 0320	F0 0120	L	Not used in master mode	00

Bit 2 Bit 2 is set if the RAM in the Pod has been written to (contaminated) coming out of RUNUUT. There is little chance of this happening, but it is a possibility.

Bit 3 Bit 3 is not set if the routine cannot find the PCB registers. There is the possibility that the UUT's program moves the PCB registers from their default location by means of the relocation register. In this case, the PCB Recovery program will not help you determine the values for the PCB registers.

USING THE PCB RECOVERY PROGRAM I-3.

Use the following procedure to recover PCB register values from a UUT:

1. Connect a known-good UUT to the Pod and make sure no errors occur with a BUS TEST.
2. Insert the PCB Recovery cassette with side A up.
3. Press READ TAPE to read the tape. (Press YES/ENTER in response to *READ TAPE - ARE YOU SURE?*)
4. Press EXEC 0 ENTER to run program 0. You'll see the message *80188 POD PCB RECOVERY PROGRAM* and the program will do the WRITE @ F00006 = 1 and enter RUNUUT.
5. A message will ask *READY TO RECOVER PCB REGS?* to determine whether or not the UUT's initialization routines have been completed.

NOTE

You must wait until the UUT's initialization routines have completed to ensure that the PCB register contents are valid.

6. Once you are certain that the PCB register contents have been established, press ENTER/YES. The program will cause the Pod to do a soft exit from RUNUUT.
7. The program will ask if you want to *SET POD TO RECOVERED VALUES?*. If you press YES/ENTER, then all the PCB register values from the F003XX range are copied to the F001XX addresses.

To use the recovered values, a program is normally created to write the initialization values to all the PCB registers (F001XX addresses). Then, when troubleshooting an inoperative UUT, the Pod is configured using this tape.

While using the PCB Recovery program, any of several error messages might display. The error messages that can occur are:

PCB REGISTERS NOT FOUND-

This is displayed if the UUT's program shifted the location of the PCB registers with the relocation register (bit 3 of F00006 was 0). In this case, the program cannot help you determine what the PCB registers should be.

RUNUUT EXIT FAILED- TRY AGAIN?

This is displayed when the NMI was not executed due to another NMI active or the processor being in a hold or wait state (bit 1 was set to 1). In this case, go ahead and re-execute the program—try waiting longer or putting the UUT in a different state before recovering the registers.

FAILURE OCCURRED
POD REGS MAY BE CONTAMINATED
RESET POD TO PWR-UP DEFAULT?

This message indicates that something went wrong in the exit from RUNUUT and the Pod's RAM was written to inadvertently. When this occurs, either turn power to the Pod off and back on or, if the user answers YES, the program will re-initialize all Pod registers (bit 2 of F00006 was set to 1). In this case, after the Pod has been re-initialized, try the program again. Wait for a longer period before recovering the registers or put the UUT in a different state.

```
        9000A-80188 INTERFACE POD PCB RECOVERY PROGRAM

A source program for the Fluke 9000A Language Compiler

Date: 6/14/85
Revision :0
Revision History: none

    Program 0  PCB Recovery
    Program 1  PCB Register Swap
    Program 2  Restore PCB values to Power-up default
    Program 3  Restore all Pod values to reset condition

User Instructions: 9000A-80188 Interface Pod Getting Started Manual
                   9000A-80188 Interface Pod Instruction Manual, Appx I

Tape Part Number: 768713

(c) Copyright 1985, John Fluke Mfg. Co., Inc. All rights reserved.

 This program executes a special "soft" exit from
RUNUUT that allows the PCB (Peripheral Control Block)
to be obtained from a working UUT's program memory rather
than being set-up manually.  Normally, this program would be used
once and then the PCB registers would be set up by another
programmed tape written by the user.


include "80188.POD"

declarations
        assign rega to key

setup information
        pod = 80188
        TRAP ACTIVE FORCE LINE =  NO
program main
```

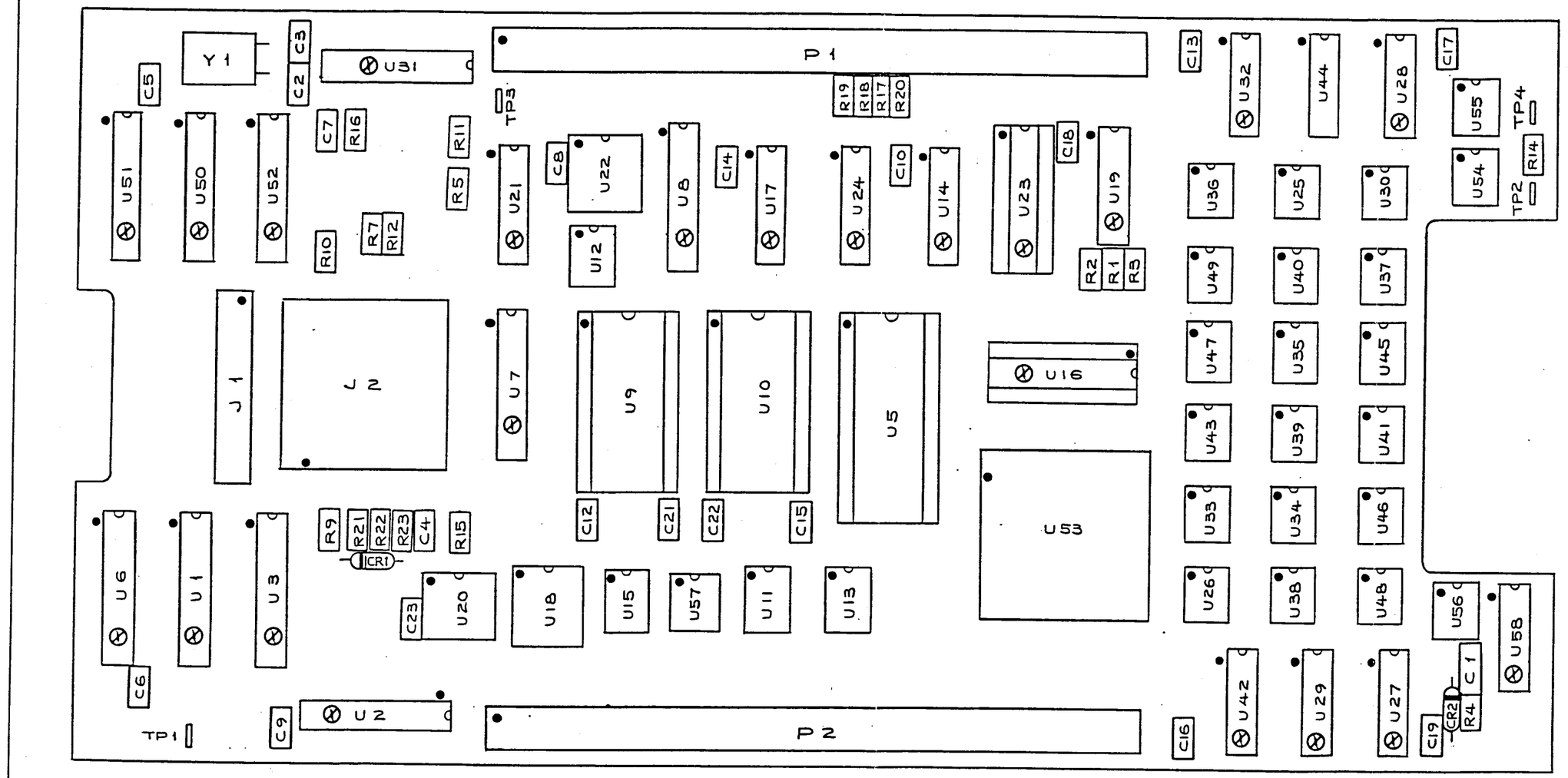
Figure I-1. PCB Recovery Program

```
begin:
        dpy 80188 POD PCB RECOVERY PROGRAM
        execute delay
redo:
        write @ F00006 = 1                          ! write to special address
                                                    ! to let Pod know that a
                                                    ! RUNUUT with soft exit
                                                    ! will take place.
        RUN UUT @ FFFF0                             ! Start the UUT's program
running:
        dpy READY TO RECOVER PCB REGS?key           ! Wait until the UUT's
                                                    ! initialization has completed
        if key = 1 goto next
        goto running
next:
        execute statwiggle                          ! lower MAINSTAT to
                                                    ! generate an NMI
        read @ F00006                               ! Special address returns
                                                    ! whether the recovery was
                                                    ! successful or not.
        regb =rege and 2                            ! Check bit 1 (if NMI
                                                    ! worked)
        if regb = 0 goto NMI_FAIL
        regb = rege and 8                           ! Check bit 3 ( if the
                                                    ! UUT's program shifted the
                                                    ! location of the PCB's)
        if regb = 0 goto pcb_shift
        regb =rege and 4                            ! Check bit 2 (if contaminated
                                                    ! RAM found in the Pod)
        if regb = 4 goto Bad_RAM

        dpy RECOVERY SUCCESSFUL
        execute delay
        dpy SET POD TO RECOVERED VALUES?key
        if key =0 goto end
        dpy WORKING...
        execute swap                                ! Trade F003XX and F001XX
        goto complt                                 ! Addresses

pcb_shift:                                          ! If the UUT's program
        dpy PCB REGISTERS NOT FOUND                 ! changes the relocation
        execute delay                               ! register contents
        dpy PCB RECOVERY NOT COMPLETED
        execute delay
        dpy TRY RECOVERY AGAIN?key
        if key = 1 goto redo
        goto end
Bad_RAM:

        dpy FAILURE OCCURRED
        execute delay
        dpy POD REGS MAY BE CONTAMINATED
        execute delay
        dpy RESET POD TO PWR-UP DEFAULT?key
        if key = 0 goto  end
        execute set_up
        dpy ALL POD REGS RESTORED TO DEFAULT
        execute delay
        goto end

NMI_FAIL:
        dpy RUNUUT EXIT FAILED-TRY AGAIN?key
        if key = 1 goto redo
        goto end

restore:
        dpy SET POD TO DEFAULT PCB VALUES?key
        if key = 0 goto end
        dpy WORKING...
        execute default
complt:
        dpy 80188 POD PCB SETUP COMPLETE
        execute delay
end:
        dpy 80188 PCB RECOVERY PROGRAM DONE

! This program swaps the contents of the PCB registers
! from the F003XX special addresses to the F001XX addresses

program swap

        regb = F00320                               ! Swap  INT. Vector Reg.
        READ @ regb                                 ! (low byte)
        WRITE @ regb and FFF1FF = rege
        regb = F00321                               ! Swap  INT. Vector Reg.
        READ @ regb                                 ! (high byte)
        WRITE @ regb and FFF1FF = rege
        regb = F00328
loop2:                                              ! Swap Interrupt
                                                    ! Controller Registers
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = regb inc 1
        if regb = F00340 goto incr2
```

Figure I-1. PCB Recovery Program (cont)

```
        goto loop2
incr2:
        regb = F00350

loop3:                                          ! Swap Timer Registers
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = regb inc 1
        if regb = F00364 goto incr3
        goto loop3
incr3:
        regb = F00366
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = F00367
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = F003A0

loop4:                                          ! Swap Chip Select regs.
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = regb inc 1
        if regb = F003AA goto incr4
        goto loop4
incr4:
        regb = F003C0

loop5:                                          ! Swap DMA controller Reg
        READ @ regb                             ! ( Channel 0)
        WRITE @ regb and FFF1FF = rege
        regb = regb inc 1
        if regb = F003CC goto incr5
        goto loop5
incr5:
        regb = F003D0
loop6:                                          ! Swap DMA controller Reg
        READ @ regb                             ! ( Channel 1)
        WRITE @ regb and FFF1FF = rege
        regb = regb inc 1
        if regb = F003DC goto incr6
        goto loop6
incr6:
        regb = F003FE
        READ @ regb
        WRITE @ regb and FFF1FF = rege
        regb = F003FF
        READ @ regb
        WRITE @ regb and FFF1FF = rege

! This program restores the PCB registers to their
! power-up default values.

program default
        write @ F00120 = 0
        write @ F00121 = 0
        write @ F00122 = 0
        write @ F00123 = 0
        write @ F00128 = FD
        write @ F00129 = 0
        write @ F0012A = 7
        write @ F0012B = 0
        write @ F0012C = 0
        write @ F0012D = 0
        write @ F0012E = 0
        write @ F00130 = 0
        write @ F00131 = 80
        write @ F00132 = F
        write @ F00133 = 0
        write @ F00134 = F
        write @ F00135 = 0
        write @ F00136 = F
        write @ F00137 = 0
        write @ F00138 = F
        write @ F00139 = 0
        write @ F0013A = F
        write @ F0013B = 0
        write @ F0013C = F
        write @ F0013D = 0
        write @ F0013E = F
        write @ F0013F = 0
        write @ F00150 = 0
        write @ F00151 = 0
        write @ F00152 = 0
        write @ F00153 = 0
        write @ F00154 = 0
        write @ F00155 = 0
        write @ F00156 = 0
        write @ F00157 = 0
        write @ F00158 = 0
        write @ F00159 = 0
        write @ F0015A = 0
        write @ F0015B = 0
        write @ F0015C = 0
        write @ F0015D = 0
```

Figure I-1. PCB Recovery Program (cont)



9000A-80188-1671

Figure 8-1. A41 Interface PCB Assembly

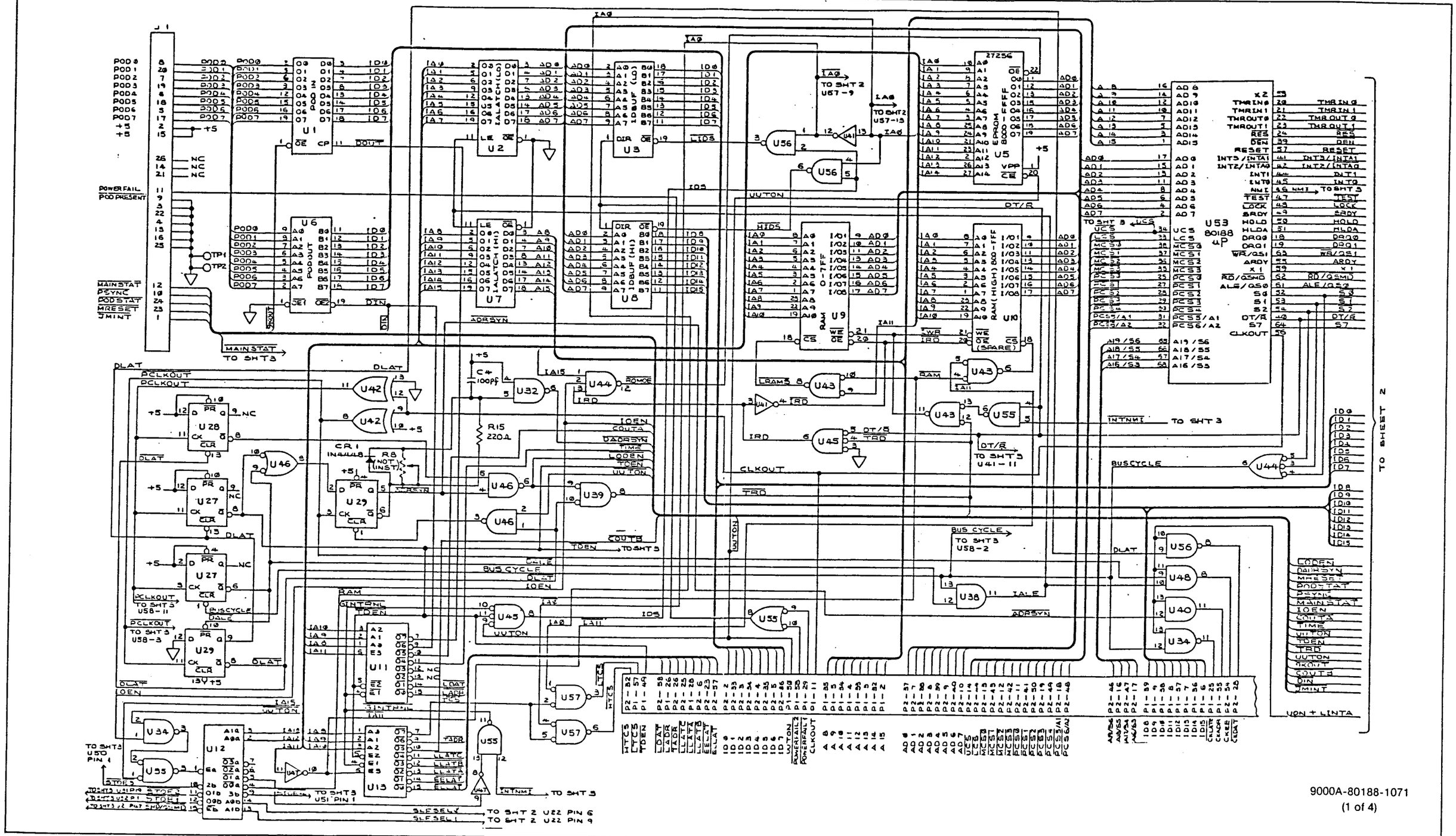


Figure 8-1. A41 Interface PCB Assembly (cont)

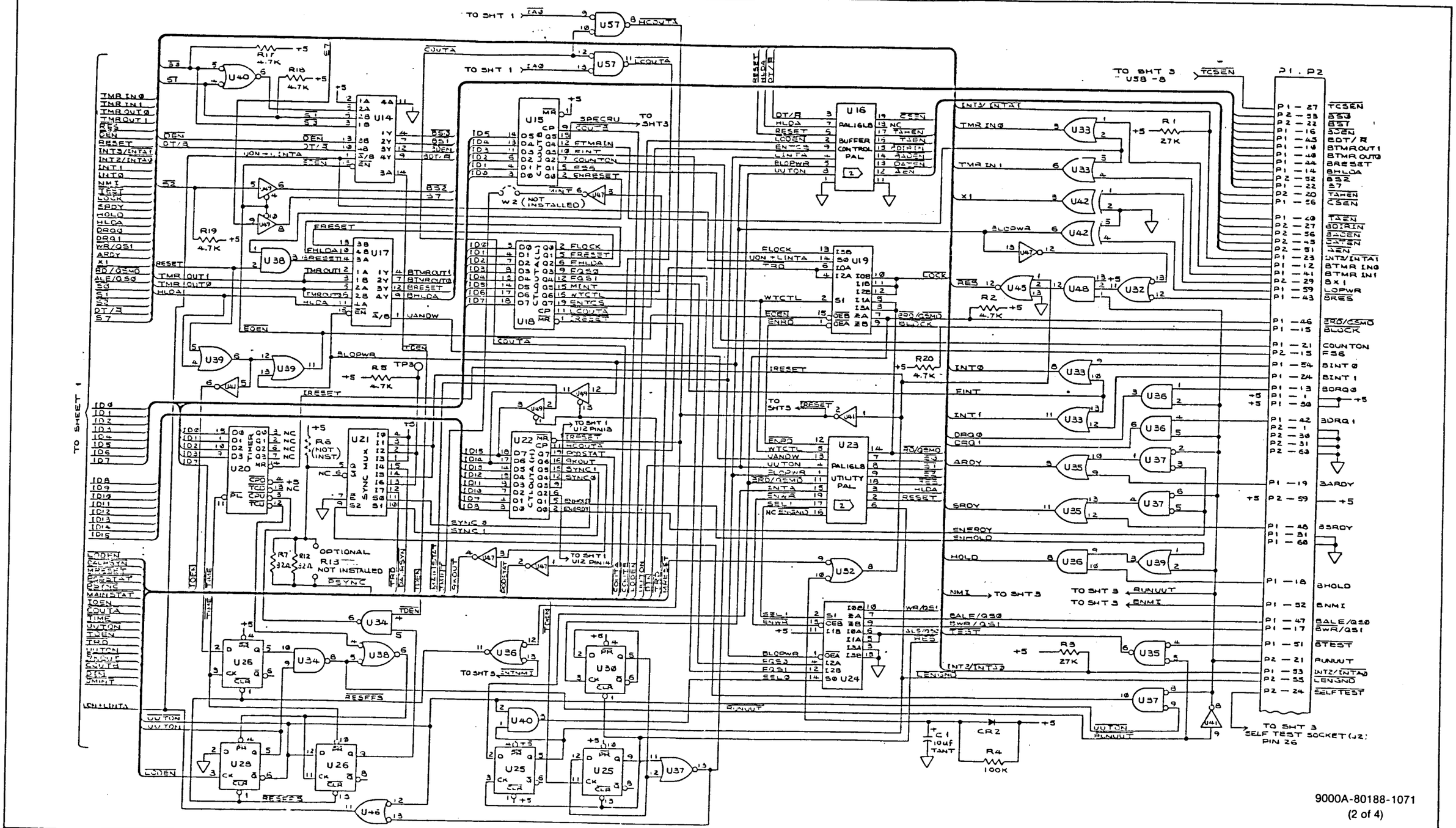
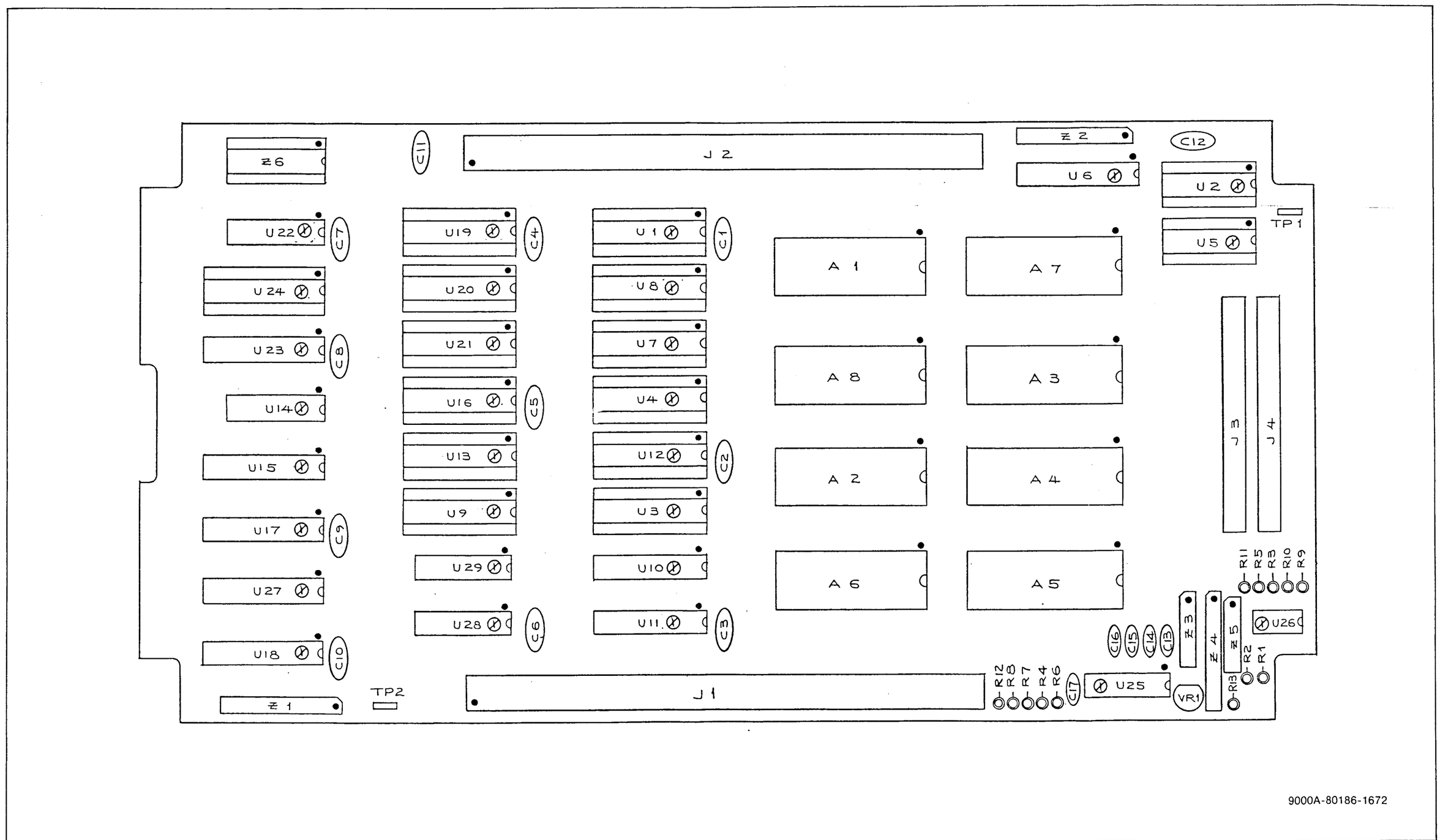


Figure 8-1. A41 Interface PCB Assembly (cont)



9000A-80186-1672

Figure 8-2. A42 Processor PCB Assembly

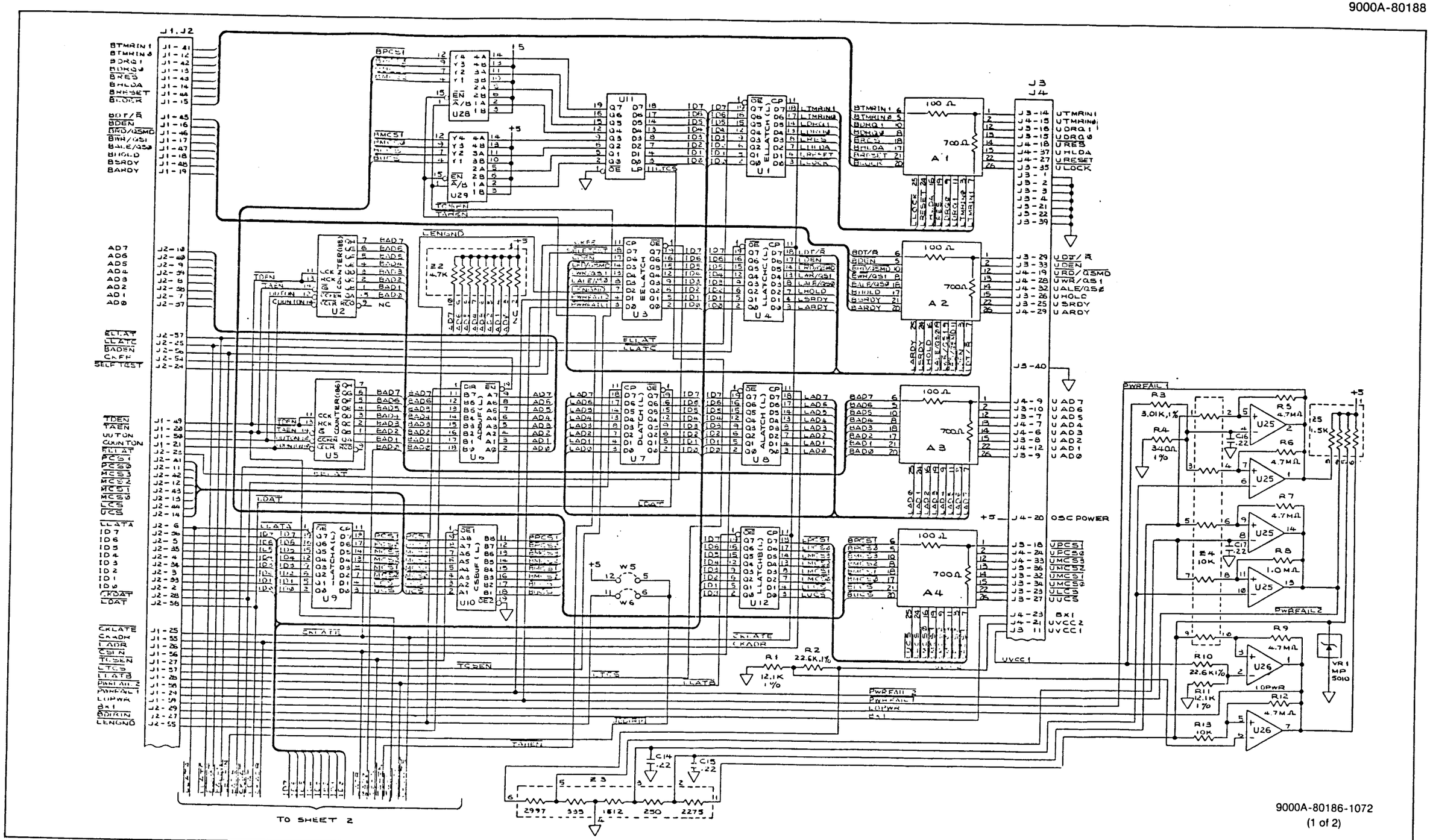
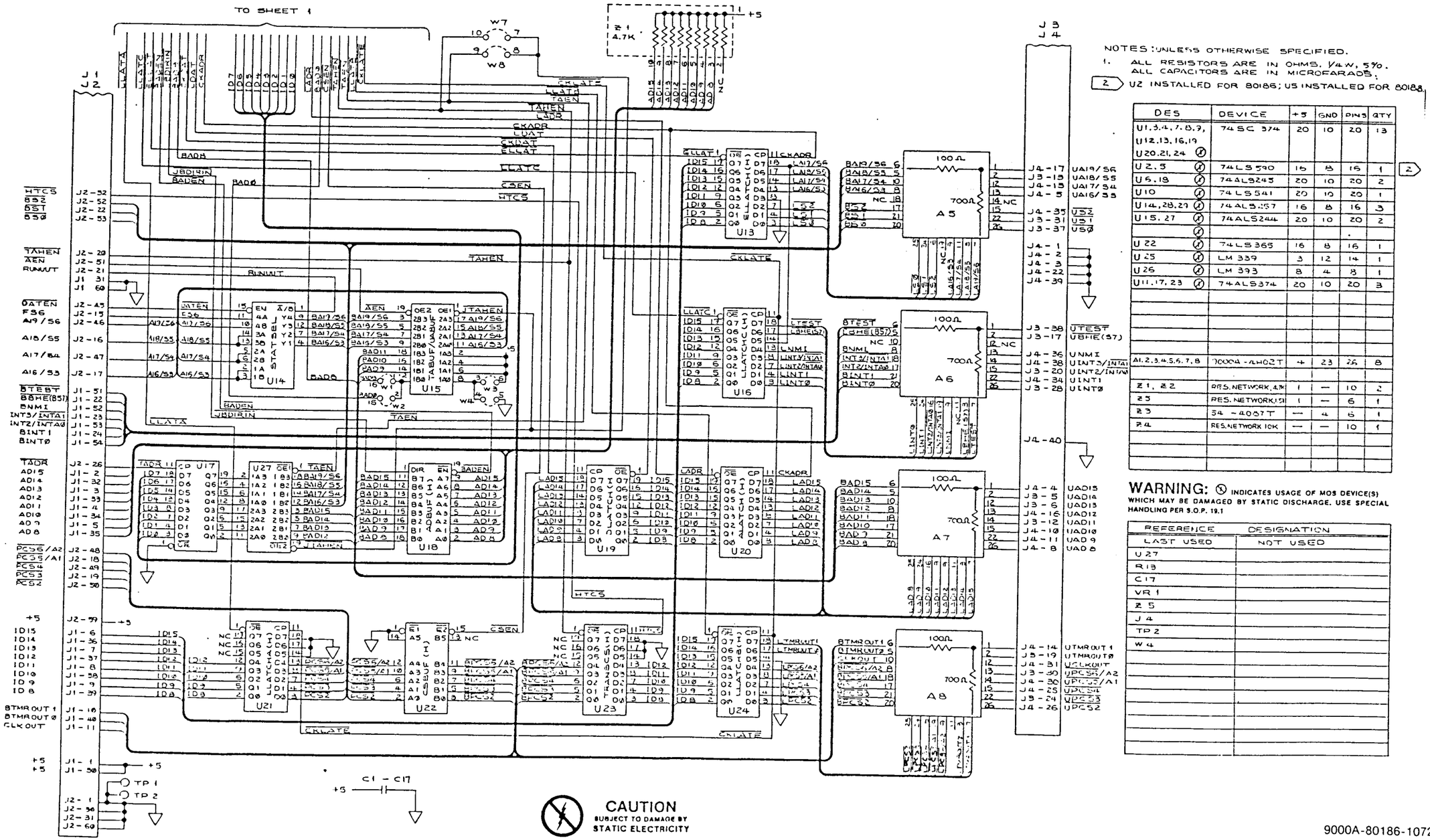


Figure 8-2. A42 Processor PCB Assembly (cont)



NOTES: UNLESS OTHERWISE SPECIFIED.

1. ALL RESISTORS ARE IN OHMS, 1/4 W, 5%.
   ALL CAPACITORS ARE IN MICROFARADS.

(2) U2 INSTALLED FOR 80186; U5 INSTALLED FOR 80188

DES	DEVICE	+5	GND	PINS	QTY
U1,3,4,7,8,9, U12,13,16,19 U20,21,24 Ⓢ	74SC 374	20	10	20	13
U2,5 Ⓢ	74LS 590	16	8	16	1
U6,18 Ⓢ	74ALS245	20	10	20	2
U10 Ⓢ	74LS541	20	10	20	1
U14,28,29 Ⓢ	74ALS.857	16	8	16	3
U15,27 Ⓢ	74ALS244	20	10	20	2
Ⓢ					
U22 Ⓢ	74LS365	16	8	16	1
U25 Ⓢ	LM 339	3	12	14	1
U26 Ⓢ	LM 393	8	4	8	1
U11,17,23 Ⓢ	74ALS374	20	10	20	3
A1,2,3,4,5,6,7,8	70004-4402T	+	23	26	8
Z1, Z2	RES. NETWORK, 4.7K	1	—	10	2
Z5	RES. NETWORK, 1K	1	—	6	1
Z3	34-4057T	—	4	6	1
Z4	RES. NETWORK 10K	—	—	10	1

WARNING: Ⓢ INDICATES USAGE OF MOS DEVICE(S) WHICH MAY BE DAMAGED BY STATIC DISCHARGE. USE SPECIAL HANDLING PER S.O.P. 19.1

REFERENCE DESIGNATION	
LAST USED	NOT USED
U27	
R13	
C17	
VR1	
Z5	
J4	
TP2	
W4	

9000A-80186-1072
(2 of 2)

Figure 8-2. A42 Processor PCB Assembly (cont)

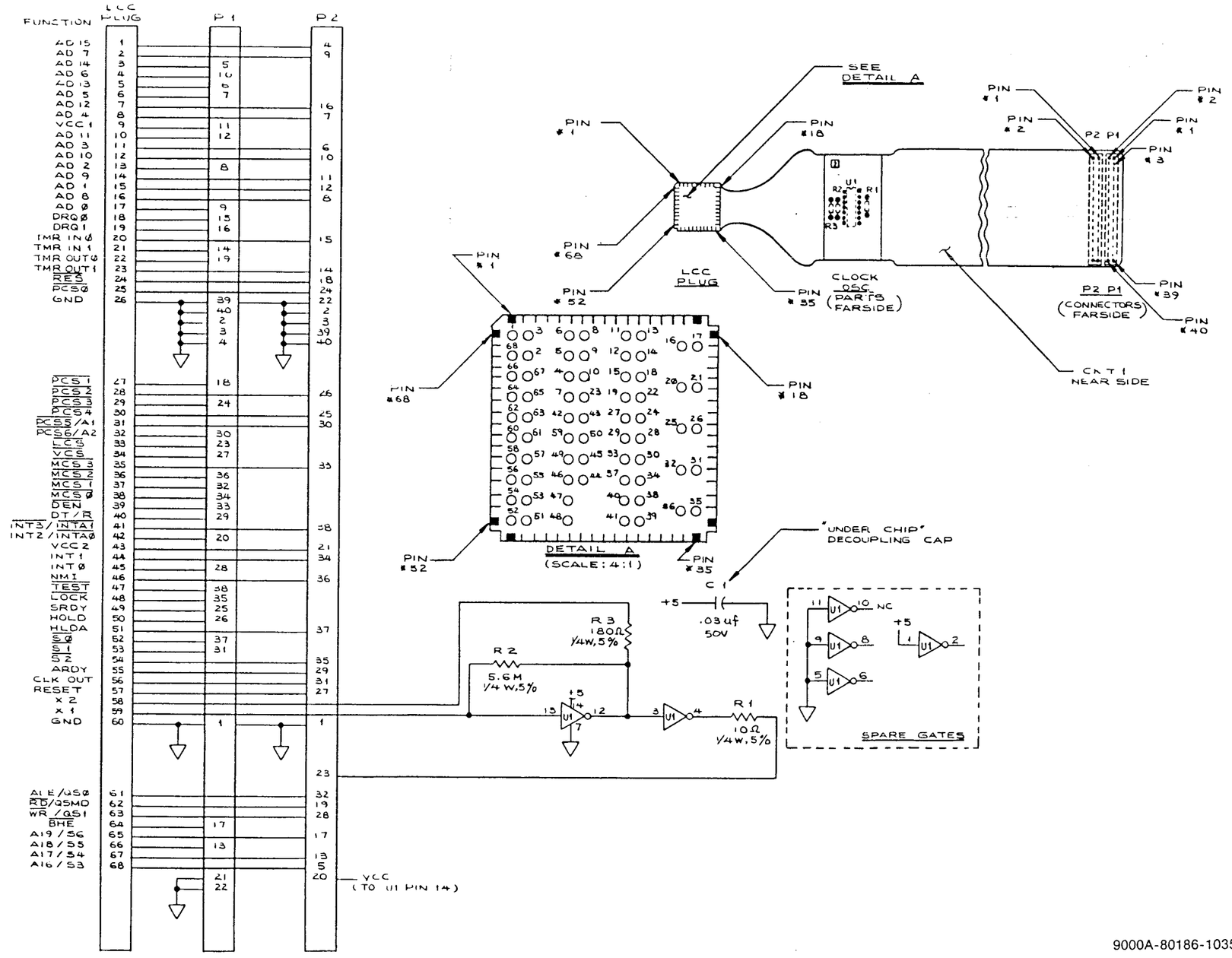


Figure 8-3. Schematic Diagram of UUT Cable