9000A-Z80QT

INTERFACE POD

Instruction Manual

P/N 859447
March 1989

FLUKE

WARRANTY

COVERAGE

Fluke warrants this interface pod to be free from defects in material and workmanship under normal use and service for a period of one (1) year from the date of shipment. This warranty extends only to the original purchaser and does not apply to any product that has been misused, altered, or has been subjected to abnormal conditions of operation.

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

SERVICE

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

DISCLAIMER

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

GETTING ANSWERS AND ADVICE

To enhance your use of this pod, Fluke will be happy to answer your questions about its applications and use. Address all correspondence to: JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WASHINGTON, 98206, ATTN: Sales Department.

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

Table of Contents

(continued on page iii)

TABLE OF CONTENTS, *(continued)*

(continued on page iii)

TABLE OF CONTENTS, *(continued)*

List of Tables

List of Illustrations

Section 1
Introduction

1-1. PURPOSE OF INTERFACE POD

The 9000A-Z80QT Interface Pod (hereafter referred to as the pod) interfaces any 9000 Series or 9100 Series tester (hereafter referred to as the mainframe) to a piece of equipment that uses a Z80 microprocessor.

The 9000 Series Digital Troubleshooters and 9100 Series Digital Test Systems are designed to test and service printed circuit boards, instruments, and systems that use bus-oriented microprocessors. The interface pod adapts the general purpose architecture of the mainframe to a specific microprocessor or microprocessor family. The interface pod adapts the mainframe to microprocessor-specific functions such as pin layout, status/control functions, interrupt handling, timing, size of memory space, and size of I/O space.

Appendix A contains information about using this pod with 9100 Series Digital Test Systems in TL/1 programs.

1-2. DESCRIPTION OF INTERFACE POD

The pod consists of a pair of printed circuit board assemblies mounted in a small break-resistant case. A shielded 24-conductor cable connects the printed circuit boards to the mainframe; a ribbon cable and connector provide connection to the unit under test, hereafter referred to as the UUT.

Figure 1-1 shows the relationship of the pod to the mainframe and to the UUT. Connection from the pod to the mainframe is via a front-mounted 25-pin connector. Connection to the UUT is made by plugging the ribbon cable plug directly into the microprocessor socket. The UUT microprocessor socket gives the mainframe direct access to all system components which normally communicate with the microprocessor.

The pod contains a Z80 microprocessor and supporting hardware and control software required to do the following:

- Perform handshaking with the mainframe
- Receive and execute commands from the mainframe
- Report UUT status to the mainframe
- Emulate the UUT microprocessor

The pod is powered by the mainframe, but is clocked by the UUT clock signal. Using the UUT clock signal allows the mainframe and pod to operate at the designed operating speed of the UUT.

Logic level detection circuits are provided on each line to the UUT. These circuits allow detection of bus shorts, stuck-high or stuck-low conditions, and any bus drive conflict (two or more drivers attempting to drive the same bus line).

Over-voltage protection circuits are also provided on each line to the UUT. These circuits guard against pod damage which could result from:

- Incorrectly inserting the ribbon cable plug in the UUT microprocessor socket.

- UUT faults that place potentially damaging voltages on the UUT microprocessor socket.

The over-voltage protection circuits guard against voltages of +12 to -7V on any one pin. Multiple faults, especially of long duration, may cause pod damage.

A power level sensing circuit constantly monitors the voltage level of the UUT power supply (+5V). If UUT power rises above or drops below an acceptable level, the pod notifies the mainframe of the power fail condition.

A self test socket provided on the pod enables the mainframe to check pod operation. The self test socket is a 40-pin zero-insertion force type connector. The ribbon cable plug must be connected to the self test socket during self test operation. The ribbon cable plug should also be inserted into this socket when the pod is not in use to provide protection for the plug.

1-3. SPECIFICATIONS

Specifications for the 9000A-Z80QT Interface Pod are listed in Table 1-1.

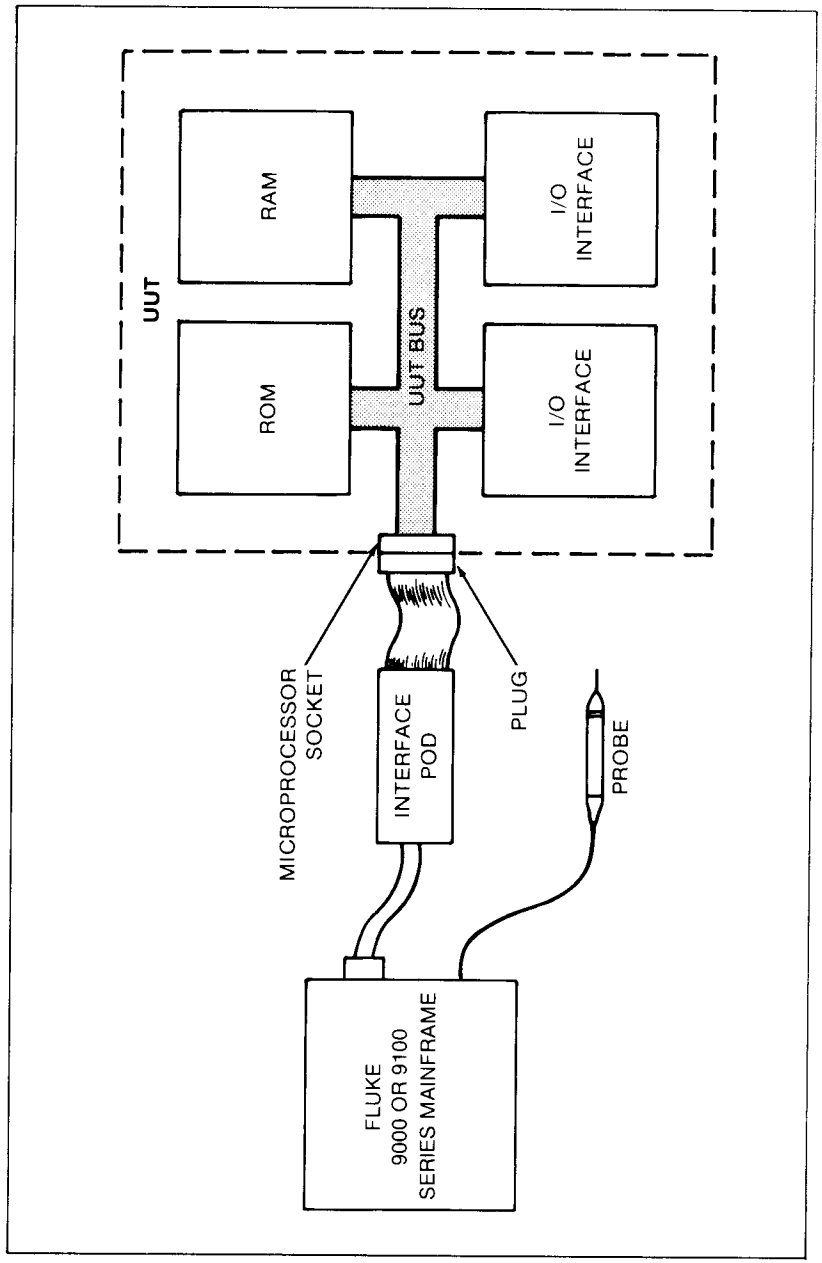


Figure 1-1. Relationship of Interface Pod

Table 1-1. Z80QT Interface Pod Specifications

ELECTRICAL PERFORMANCE	
Power Dissipation	3.0 watts maximum
Electrical Protection	-7 to +12V may be applied between ground and any single ribbon cable plug pin continuously as long as the pod is powered by the mainframe.
MICROPROCESSOR SIGNALS	
Clock Input Low	0V min., +0.45V max.
Clock Input High	+4.4V min., +5.0V max.
Input Low Voltage	0V min., +0.8V max.
Input High Voltage	+2.0V min., +5.0V max.
Output Low Voltage	+0.4V max. with Iol = 1.8 mA
Output High Voltage	+2.4V min. with Ioh = -250 μA
Tristate Output Leakage Current	$\pm$20 μA
High Level Input Current	20 μA typ. with Vih = +2.7V
Low Level Input Current	
$\overline{\text{BUSRQ}}$, $\overline{\text{WAIT}}$, $\overline{\text{RESET}}$, $\overline{\text{NMI}}$	-400 μA max. with Vil = +0.4V
ALL OTHER INPUT LINES	-20 μA typ. with Vil = +0.4V
TIMING CHARACTERISTICS	
Maximum Clock Frequency	8.0 MHz typ.
Added Delays to Z80 Signals	
LOW-TO-HIGH TRANSITIONS	20 ns typ.
HIGH-TO-LOW TRANSITIONS	24 ns typ.
UUT POWER DETECTION	
Detection of Low Vcc Fault	Vcc < +4.5V detected
Detection of High Vcc Fault	Vcc > +5.5V detected

Table 1-1. Z80QT Interface Pod Specifications (cont)

GENERAL	
Size	3.3 cm High x 10.2 cm Wide x 18.55 cm Deep (1.3 in High x 4.0 in Wide x 7.4 in Deep)
Weight	0.68 kg (1.5 lbs)
Environment	
STORAGE	-40° to +70°C, RH < 95%
OPERATING	0° to +25°C, RH < 95% +25° to +40°C, RH < 75% +40° to +50°C, RH < 45%
Protection Class 3	Relates solely to insulation or grounding properties defined in IEC 348.

Section 2
Installation

2-1. INSTALLING THE Z80QT DATABASE (9100 SERIES ONLY)

The Z80QT database for the 9100-Series mainframe is contained on one 3.5-inch floppy disk supplied with the 9000A-Z80QT Pod.

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

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

2-2. MAKING CONNECTIONS

Before a 9000 Series or 9100 Series tester (mainframe) can be used to perform a test or isolate a fault, it must be connected to the UUT. Connection is made by means of the pod, which is equipped with two cable assemblies, one shielded-type and one ribbon-type. Procedures for installing and connecting the pod are given next.

Before making any connections to the UUT, read the following precautions:

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 MICRO-PROCESSOR BEFORE CONNECTING POD.

- Be sure to install the ribbon cable plug correctly in the UUT microprocessor socket.

- The self test socket is intended for use with the ribbon cable plug only. Do not insert any microprocessor removed from a UUT, or any other device into this socket.

Connect the pod between the mainframe and the UUT as follows:

1. Remove power from the UUT. Remove power from the mainframe.

2. Using the round shielded cable, connect the pod to the mainframe as shown in Figure 2-1 (for 9000 Series) or Figure 2-2 (for 9100 Series). Secure the connector using the sliding collar.

3. Apply power to the mainframe.

4. Perform a self-test of the pod as described in Section 5 of this manual.

5. With UUT power off, unplug the microprocessor from the UUT.

6. On the pod, turn the self test socket thumbwheel to release the plug from the self test socket.

7. Align the ribbon-cable with the microprocessor socket on the UUT so that the notched corner of the ribbon cable plug aligns with pin 1 of the socket. Insert the plug into the socket as shown in Figure 2-3.

8. Electrically reassemble the UUT. Use extender boards if necessary.

CAUTION

Make sure mainframe power is on before turning UUT power on in order to activate pod protection circuits.

9. Apply power to the mainframe and the UUT.

2-3. POWER CONNECTIONS

The pod receives +5 volts, -5 volts, and +12 volts from the mainframe. No external power connections are required.

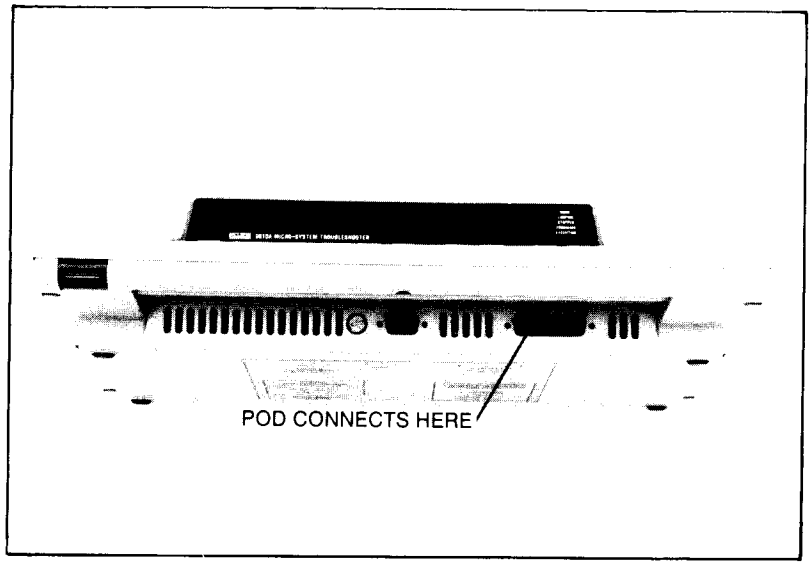


Figure 2-1. Connection of Interface Pod to 9000 Series

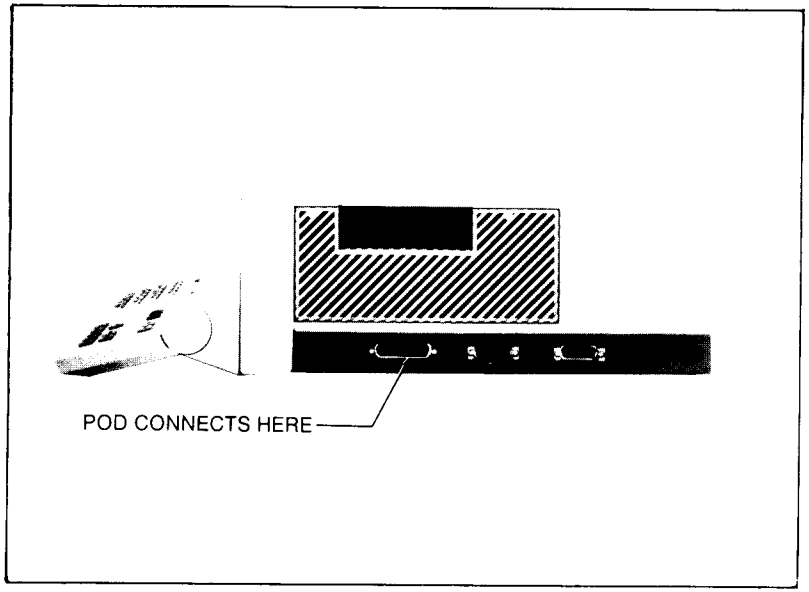


Figure 2-2. Connection of Interface Pod to 9100 Series

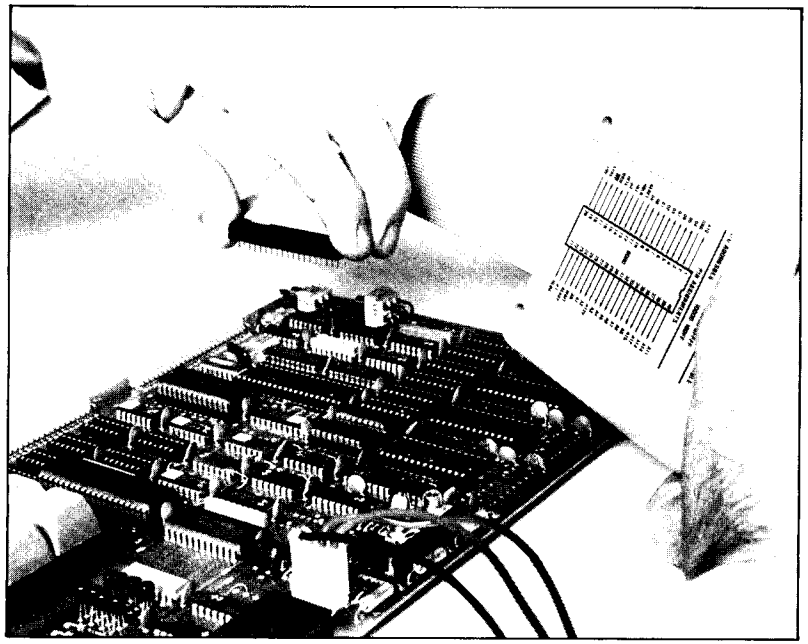


Figure 2-3. Connection of Interface Pod to UUT

Section 3
Pod Signals and Functions

3-1. INTRODUCTION

This section is a reference source for Z80QT pod-specific and Z80 microprocessor information. Information in this section includes descriptions of Z80 signals, explanations of status/control lines and address space assignment, effects the pod may have on normal UUT operation, pod capabilities and limitations, and other pod characteristics.

3-2. Z80 SIGNALS

For reference, Table 3-1 lists all of the Z80 signals and provides a brief description of each. Figure 3-1 shows the pin assignment of Z80 signals.

Table 3-1. Z80 Signals

SIGNAL NAME	DESCRIPTION
Address Lines A0 - A15	The 16 address lines are designated A0 through A15. The address lines are tri-state outputs and may be logic high, logic low, or floated by the Z80 to a high impedance state. The Z80 places the address lines in the high impedance state to allow devices other than the Z80 to control the address bus during DMA (Direct Memory Access) operations. See $\overline{\text{BUSRQ}}$.
Data lines D0 - D7	The 8 data lines are designated D0 through D7. The data lines are tri-state bi-directional lines, which are placed in the high impedance state during DMA operations. See $\overline{\text{BUSRQ}}$.
$\overline{\text{M1}}$ Line	The $\overline{\text{M1}}$ line is a control line which identifies the Z80 instruction-fetch cycle. The Z80 places $\overline{\text{M1}}$ at logic low during the instruction-fetch cycle. Also, when the Z80 acknowledges an interrupt, both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are driven low.

Table 3-1. Z80 Signals (cont)

SIGNAL NAME	DESCRIPTION
$\overline{MREQ}$ Line	The $\overline{MREQ}$ output identifies a memory access operation in progress. The Z80 places $\overline{MREQ}$ at logic low during any memory access operation. In addition, $\overline{MREQ}$ is placed in a high impedance state during DMA operations. See $\overline{BUSRQ}$.
$\overline{IORQ}$ Line	The $\overline{IORQ}$ output identifies any I/O operation in progress. When $\overline{IORQ}$ is low, address lines A0 - A15 contain a valid I/O port address. The $\overline{IORQ}$ line is also used in conjunction with the $\overline{M1}$ line as an interrupt acknowledge. When the Z80 acknowledges an interrupt, both lines are driven low. low.
$\overline{RD}$ Line	The $\overline{RD}$ output is pulled low to indicate that the Z80 is ready to read data via the data lines from either memory or an I/O device, as identified by the $\overline{MREQ}$ or $\overline{IORQ}$ line. In addition, $\overline{RD}$ is placed in a high impedance state during DMA operations. See $\overline{BUSRQ}$.
$\overline{WR}$ Line	The $\overline{WR}$ output is pulled low to indicate that the Z80 is ready to write data via the data lines to either memory or an I/O device, as identified by the $\overline{MREQ}$ or $\overline{IORQ}$ line. In addition, $\overline{WR}$ is placed in a high impedance state during DMA operations. See $\overline{BUSRQ}$.
$\overline{RFSH}$ Line	The $\overline{RFSH}$ output is a control signal which may be used in conjunction with the $\overline{MREQ}$ line to refresh dynamic memories. When $\overline{RFSH}$ is pulled low the $\overline{MREQ}$ signal and address lines A0 - A7 may be used to refresh dynamic memories.
$\overline{HALT}$ Line	The $\overline{HALT}$ output is pulled low following the execution of a halt instruction. During the halt state, the Z80 continuously executes a NOP instruction in order to maintain memory refresh activity.
$\overline{WAIT}$ Line	The $\overline{WAIT}$ line is an input which, when placed at a logic low level, causes the Z80 to enter a wait state. During the wait state, the Z80 inserts clock pulses to extend the cycle time as required by the external logic selecting the wait state.
$\overline{INT}$ and $\overline{NMI}$ Lines	The $\overline{INT}$ line is an input which permits external interrupt of the Z80 as long as interrupts are not disabled and the $\overline{BUSRQ}$ line is not at a logic low. The $\overline{NMI}$ line is a non-maskable interrupt input which cannot be disabled.

Table 3-1. Z80 Signals (cont)

SIGNAL NAME	DESCRIPTION
$\overline{RESET}$ Line	The $\overline{RESET}$ line is an input which, when placed at a logic low level, resets the program counter and other registers to zero, disables interrupt requests by the $\overline{INT}$ line, and floats all tri-state bus signals to the high impedance state.
$\overline{BUSRQ}$ Line	The $\overline{BUSRQ}$ line is an input which, when placed at a logic low level, causes the Z80 to relinquish control of the system bus by floating the address, data and associated control lines to a high impedance state.
$\overline{BUSAK}$ Line	The $\overline{BUSAK}$ output is pulled low when the Z80 acknowledges a $\overline{BUSRQ}$ input. See $\overline{BUSRQ}$.

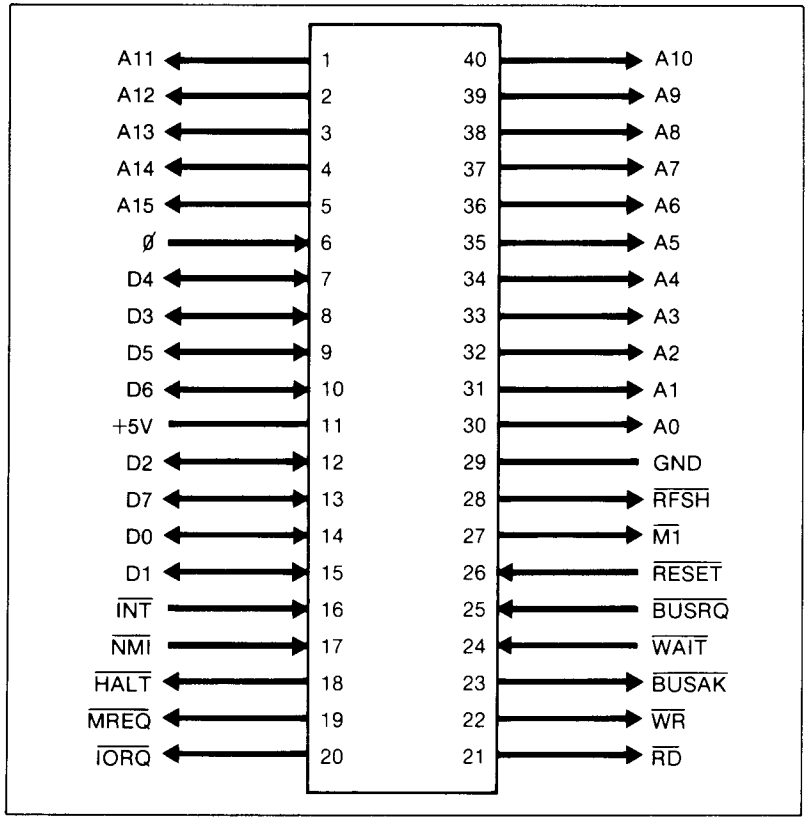


Figure 3-1. Z80 Pin Assignments

3-3. STATUS/CONTROL LINES AND ADDRESS SPACE ASSIGNMENT

3-4. Introduction

The 9000 Series and 9100 Series mainframes accommodate bus-oriented processors having up to 32 address lines, 32 data lines, 16 status lines, and 8 control lines. The pod provides an interface between the general architecture of the mainframe and the specific requirements of the Z80 microprocessor. As part of this interface task, the pod makes specific assignments between the microprocessor lines and the mainframe. These assignments include the following:

- Bit number assignment of Z80 status lines
- User-writable control lines
- Bit number assignment of control lines
- Address space assignment
- Pin assignments

These assignments are described in the following paragraphs and are summarized for convenience on the pod decal.

3-5. Bit Assignment - Status Lines

When a read status operation is performed, the mainframe displays the result in binary form, where a 1 indicates a logic high status line and a 0 indicates a logic low status line. To determine which characters of the display correspond to specific status lines, refer to Table 3-2. This table shows that each line is assigned a bit number. Bit number zero ($\overline{WAIT}$) appears at the far right of the display, while bit number 7 (POWER FAIL) appears at the far left side.

For example, if the $\overline{NMI}$ (bit number 2) and POWER FAIL (bit number 7) lines are low, and all other status lines are high, a 9000 Series mainframe would read READ @ STS = 0001 1011 OK. Bit numbers 2 ($\overline{NMI}$) and 7 (POWER FAIL) are zero to indicate a logic low, while other meaningful bits are ones to indicate logic high. Bits 5 and 6, which have no meaning as Z80 status lines, are always represented by zeros in the mainframe display message.

3-6. User-Writable Control Lines

The Z80 has two control lines to which the mainframe can write. These lines are bus acknowledge ($\overline{BUSAK}$) and halt ($\overline{HALT}$). To write to either or both of these lines, a control function is used as described in the paragraphs that follow. Note that writing to a control line only sets the line to the high or low state for approximately 20 microseconds, just long enough to verify that it can be driven.

3-7. Bit Assignment - Control Lines

There are two troubleshooting functions which require the entry of binary digits to identify user-writable control lines. These functions are write control and data toggle control.

When performing or programming either of these two functions, the user is prompted for a binary number to identify the control line(s) to be written. $\overline{BUSAK}$ or $\overline{HALT}$. Table 3-2 shows that these lines are assigned to bit numbers 0 and 1 respectively. To perform a write control operation on these two lines, enter any of the following four bit configurations in response to the prompt. As with the status lines, bit number 0 is at the far right of the display.

00	writes both lines low
01	writes $\overline{BUSAK}$ high and $\overline{HALT}$ low
10	writes $\overline{BUSAK}$ low and $\overline{HALT}$ high
11	writes both lines high

If any control line cannot be driven, the 9000 Series mainframe responds with the message CTRL ERR @ xxxxxxxx LOOP?, where x equals a binary 1 if that line cannot be driven. For example, if in the write control operation, the $\overline{HALT}$ line can be driven, but the $\overline{BUSAK}$ line cannot, the 9000 Series mainframe displays the message CTRL ERR @ 00000001 LOOP?. The $\overline{BUSAK}$ line is represented by bit number 0. (A 9100 series mainframe displays a message in English that a control line cannot be driven.)

When performing a BUS TEST and various other mainframe operations, the 9000 Series mainframe message CTL ERR xxxxxxxx-LOOP? can occur, where x represents a binary number that identifies which lines can or cannot be driven. A binary 0 represents the ability to drive a line, while a binary 1 represents the inability to drive a line. Table 3-2 lists all control lines and their respective bit numbers.

3-8. Address Space Assignment

The Z80 is capable of addressing up to 65,536 memory locations and up to 65,536 I/O locations. The mainframe uses a consistent technique of addressing multiple memory and I/O locations.

In order to access one of the 65,536 memory locations, the user provides a hexadecimal address in the range of 0000 to FFFF. In order to access one of the I/O locations, the user provides a hexadecimal address in the range of 10000 to 1FFFF. For convenience, these assignments are also summarized on the pod decal.

Table 3-2. Status and Control Lines Bit Assignments

STATUS LINES		CONTROL LINES	
BIT NO.	SIGNAL	BIT NO.	SIGNAL
7	PWR FAIL	7	$\overline{\text{MREQ}}$
6	—	6	$\overline{\text{M1}}$
5	—	5	$\overline{\text{WR}}$
4	**$\overline{\text{RESET}}$	4	$\overline{\text{RD}}$
3	$\overline{\text{INT}}$	3	$\overline{\text{IORQ}}$
2	$\overline{\text{NMI}}$	2	$\overline{\text{RFSH}}$
1	**$\overline{\text{BUSRQ}}$	1	*$\overline{\text{HALT}}$
0	**$\overline{\text{WAIT}}$	0	*$\overline{\text{BUSAK}}$

*User writeable **Forcing Lines

A special address exists within the pod address space to set the value of the I register in the Z80 microprocessor. This is useful only for UUTs that use the contents of the I register (interrupt vector register) when it is echoed on the upper eight bits of the address bus during an interrupt acknowledge cycle. The special pod address for the I register is 20000, and it may take any value from 0 to FF.

3-9. FORCING AND INTERRUPT LINES

Several mainframe messages are used to indicate errors and conditions associated with forcing lines and interrupts. Forcing lines are those lines which, when made active, force the microprocessor into some specific action. Forcing lines for the Z80 are $\overline{\text{BUSRQ}}$, $\overline{\text{WAIT}}$, and $\overline{\text{RESET}}$. Pulling $\overline{\text{BUSRQ}}$ or $\overline{\text{WAIT}}$ low could cause the pod to stop and timeout. Note that these two lines can be disabled during mainframe setup procedures. If the $\overline{\text{RESET}}$ line is pulled low, the pod reports such a condition to the mainframe, but pod operation is unaffected.

Interrupt lines for the Z80 include $\overline{\text{INT}}$ and $\overline{\text{NMI}}$. The $\overline{\text{INT}}$ input is software disabled; the $\overline{\text{NMI}}$ input is hardware disabled except during operation in the RUN UUT mode. When disabled, the NMI input is routinely checked by the pod software, and reported to the mainframe if held low by the UUT.

NOTE

During mainframe setup, disabling $\overline{\text{BUSRQ}}$ and $\overline{\text{WAIT}}$ eliminates any effect they might have on mainframe/pod operation. Not reporting (trapping) forcing line or interrupts during setup simply eliminates the corresponding mainframe message.

3-10. LINES ENABLED DURING MAINFRAME SETUP

During setup of the mainframe, the operator has the option of enabling or not enabling certain forcing lines as a means of preventing UUT faults from disabling the pod microprocessor. For the Z80, these lines include $\overline{\text{BUSRQ}}$ and $\overline{\text{WAIT}}$. Also during mainframe setup, the operator may elect to report (trap) or disregard active signals on the forcing lines. Reporting active forcing lines halts mainframe operation in order to display the forcing line message.

3-11. NON-DETECTABLE Z80 SIGNALS

The pod does not detect the absence of the $\overline{\text{M1}}$ signal. However, this signal can be observed, if necessary, by using the probe or scope trigger output of the mainframe to trigger a scope. (Figure 4-4 shows the mainframe scope trigger output signal.)

3-12. QUICK-LOOPING READ AND WRITE FUNCTIONS

The Z80QT Pod provides Quick-Looping read and write functions with repetition rates considerably faster than ordinary mainframe looping functions selected by pressing the LOOP key. Because of their increased repetition rate, Quick-Looping functions are particularly useful for enhanced viewing of signal traces on an oscilloscope synchronized to the TRIGGER OUTPUT pulse (available on the rear panel of the mainframe).

Unlike ordinary mainframe looping functions, software that controls Quick-Looping functions resides in the pod and not in the mainframe. Instructions for using the Quick-Looping read and write functions of the pod for the 9000 and 9100 Series mainframes are presented separately under the next headings.

It should be noted that diagnostics performed by the pod during execution of Quick-Looping read or write operations are less rigorous than diagnostics performed during execution of the ordinary looping function. The pod reports to the mainframe any UUT system errors detected during the first iteration only. Subsequent UUT system errors are not reported.

If both error reporting and Quick-Looping functions are desired, you may apply the ordinary mainframe looping function to the Quick-Looping read or write, such as READ @ 10 2000 LOOP in the case of the 9000 Series. The mainframe commands read operations at address 10 2000 at the normal looping speed with full error reporting. For every ordinary read operation, the pod interjects a few Quick-Looping read operations (with no error reporting) to enhance oscilloscope viewing.

CAUTION

To prevent possible damage to the probe or the UUT, do not use the probe to generate stimulus pulses while a Quick-Looping function is being performed if the UUT crystal or clock frequency is less than 1 MHz.

The preceding caution is necessary because the combination of the high repetition rate of a Quick-Looping function and a slow UUT clock (below 1 MHz) greatly increases the on time of the pulser. If the on time is excessively long, probe stimulus pulses can cause damage to the probe or the UUT. Note that the response capability of the probe, such as logic level reading with the Read Probe operation, is unaffected by high duty cycles.

3-13. Using the 9000 Series for Quick-Looping Read and Write

NOTE

Special addresses mentioned in the following paragraphs are also valid for 9100 Series mainframes. However, they are not necessary because the 9100 Series provides softkeys to directly access these pod functions. The recommended method of running pod-controlled quick tests from the 9100 Series is given under a separate heading.

The operator selects these functions by using the special addresses listed in Table 3-3. For example, a read operation at address 10 0F00 causes the pod to perform the Quick-Looping read operation at address 00 0F00.

3-14. Using the 9100 Series for Quick Looping Read and Write

To start Quick-Looping read using the 9100 Series mainframe, proceed as follows:

1. Press the POD key.
2. Press the QWK_RD softkey, followed by ENTER.

Table 3-3. Quick-Looping Read and Write Test Addresses

OPERATION	PARAMETERS
WRITE @ 1X XXXX=YY	X XXXX = Write Address YY = Write Data
READ @ 1X XXXX	X XXXX = Read Address

3. Enter the address you want to read, followed by ENTER.
4. The mainframe displays the contents of the address (from the first read), and the pod is in control of the looping process.

To start Quick-Looping write using the 9100 Series mainframe, proceed as follows:

1. Press the POD key.
2. Press the QWK_WR softkey, followed by ENTER.
3. Enter the address desired, followed by the right-arrow key.
4. Enter the data byte value in hex, followed by ENTER.
5. The pod is now in control of the looping process.

3-15. QUICK MEMORY TESTS

Quick Memory tests are controlled by software inside the pod, rather than software inside the mainframe. Quick Memory tests execute much faster than the corresponding tests built into the mainframe. Instructions for using the Quick RAM and Quick ROM functions of the pod using the 9000 and 9100 Series mainframes are presented separately after a description of the tests.

3-16. Quick RAM Test Description

The Quick RAM Test allows the operator to test RAM address blocks more quickly than with the 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.

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

- The normal RAM test consists of two phases: the first test phase is a read/write check, while the second phase checks address decoding. The read/write check is performed by writing and reading a 1 and a 0 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 has been written to each address. For the address decoding check, the pod reads each address and compares the read data with the unique word that is expected.

- The pattern verification test simply verifies that memory contains expected data. The test should be used following a normal Quick RAM test to verify that the memory still contains the correct data after a longer period than is checked by the normal RAM test. The pattern verification test is provided primarily for testing dynamic RAM memory to assure that the memory retains information properly. If problems with dynamic RAM are suspected, use the pattern verification test following the normal RAM test.

3-17. Quick ROM Test Description

The Quick ROM test allows the operator to test ROM address blocks more quickly than with the ordinary ROM test. When the Quick ROM test is performed, the pod obtains a checksum that may be compared with a checksum obtained by performing the Quick ROM test over the same address block of a known good UUT. The value of this Quick ROM test checksum is also available in Table 5-5. Note that this checksum is not the same value as the signature that is obtained with the ordinary mainframe ROM test.

The Quick ROM test is not as rigorous and reliable as the signature analysis used by the ordinary ROM test, nor is error reporting in the Quick ROM test as extensive. 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.

3-18. Using the 9000 Series for Quick RAM Testing

NOTE

Special addresses mentioned in the following paragraphs are also valid for 9100 Series mainframes. However, they are not necessary because the 9100 Series provides softkeys to directly access these pod functions. The recommended method of running pod-controlled quick tests from the 9100 Series is given under a separate heading.

Starting and ending addresses for the Quick RAM test are specified in a different manner than for the usual RAM test. The starting and ending addresses are specified by writing to special addresses. The starting address is defined by a WRITE @ 2X XXXX=0, where X XXXX is the address to be used to start the Quick RAM test. The ending address, address increment, and test specification are defined by a WRITE @ 2Y YYYY=ZN, where Y YYYY is the desired ending address, Z is the desired increment, and N is the test specification. If Z is omitted or specified as 0, the address increment defaults to 1. N may be either 1 (normal Quick RAM test) or 2 (pattern verification test). The ending address must be greater than or equal to the starting address.

For example, to specify a normal Quick RAM test over RAM addresses 5000 through 5FFF with the default address increment of 1, do the following two operations:

```
WRITE @ 20 5000=0
WRITE @ 20 5FFF=1
```

To follow that test with a pattern verification test over the same address space, rewrite the ending address with the new specification:

```
WRITE @ 20 5FFF=2
```

The Quick RAM test begins execution as soon as the operator completes the entry of the ending address. During and after execution of the test, the mainframe does not display any information about the progress or results of the test unless requested by the operator. The test may be aborted before completion by selecting another operation.

To determine if the Quick RAM test is still in progress or what the test results are, the mainframe operator should perform a READ @ ENTER operation (which commands a READ operation at the last entered address). In response, the pod returns a byte indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-4.

Read-only special addresses in the F0 20XX range contain additional information about the Quick RAM test, including errors and records of addresses used. The special addresses for the Quick RAM test are described in Table 3-4. Make sure that you first specify the READ @ ENTER to find out if the test has been completed before reading at any of the special addresses. Unless the test has been completed (or failed), the information contained at the special addresses will pertain to a previous test rather than the current test, and the current test will be aborted.

For example, if any error is reported by the test, you can find the least-significant byte of the address where the error occurred by performing a READ @ F0 2008. You can get a hex mask of any bad data bits by performing a READ @ F0 2012.

If the status code is F0, the bad data bits mask contains the bits that were not read correctly after being written. If the status code is F1, the mask probably contains bit positions with incorrect address decoding. If the status code is F2, the mask contains the bits in which data was not retained correctly.

Table 3-4. Quick RAM Test Addresses and Status Codes

OPERATION	PARAMETERS
WRITE @ 2X XXXX=0 WRITE @ 2Y YYYY=ZN	X XXXX = Start Address Y YYYY = End Address Z = Increment N = Test Specification: 1 = Perform RAM Test 2 = Pattern Verify Check
READ @ ENTER	Returns Status Code: 00 = No test requested A0 = Aborted, new command entered A1 = Aborted, illegal data in command A2 = Aborted, illegal address in command B0 = Busy, read/write check B1 = Busy, address decoding check B2 = Busy, pattern verify check C0 = Complete, no errors F0 = Failed, read/write error F1 = Failed, address decoding error F2 = Failed, pattern verify error
READ-ONLY ADDRESS	**FUNCTION**
F0 2000 F0 2001 F0 2002	Start address, LSB Start address, 2nd byte Start address, MSB
F0 2004 F0 2005 F0 2006	End address, LSB End address, 2nd byte End address, MSB
F0 2008 F0 2009 F0 200A	Error address, LSB Error address, 2nd byte Error address, MSB
F0 200C F0 200E	Expected data Actual data
F0 2010 F0 2012	Most recent code Hex mask - bad data bits
F0 2014 F0 20F0	Returns increment and function type Most recent code

3-19. Using the 9000 Series for Quick ROM Testing

NOTE

Special addresses mentioned in the following paragraphs are also valid for 9100 Series mainframes. However, they are not necessary because the 9100 Series provides softkeys to directly access these pod functions. The recommended method of running pod-controlled quick tests from the 9100 Series is given under a separate heading.

The Quick ROM test is specified in a manner similar to the Quick RAM test. The starting and ending addresses are specified by writing to special addresses. The starting address is defined by a WRITE @ 3X XXXX=0, where X XXXX is the address to be used to start the Quick ROM test. The ending address and address increment are defined by a WRITE @ 3Y YYYY=Z1, where Y YYYY is the desired ending address and Z is the optional increment. If Z is not specified or is specified as 0, the increment will default to 1. The ending address must be greater than the starting address.

For example, to specify a Quick ROM test over ROM addresses 0000 through 0FFF, do the following two operations:

WRITE @ 30 0000=0
WRITE @ 30 0FFF=1

The Quick ROM test begins execution as soon as the operator completes the entry of the ending address. During and after execution of the test, the mainframe will not display any information about the progress or results of the test unless requested by the operator. The test may be aborted before completion by selecting another operation.

To determine if the Quick ROM test is still in progress, or what the test results are, the mainframe operator should perform a READ @ ENTER operation (which commands a READ operation at the last entered address). In response, the pod returns a byte indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-5.

Read-only special addresses in the F0 30XX range contain additional information about the Quick ROM test, including errors and records of addresses used. The special addresses for the Quick ROM test are described in Table 3-5.

Table 3-5. Quick ROM Test Addresses and Status Codes

OPERATION	PARAMETERS
WRITE @ 3X XXXX=0 WRITE @ 3Y YYYY=Z1	X XXXX = Start Address Y YYYY = End Address Z = Increment
READ @ ENTER	Returns Status Code: 00 = No test requested A0 = Aborted, new command entered A1 = Aborted, illegal data in command A2 = Aborted, illegal address in command B0 = Busy C0 = Complete, no errors C1 = Complete, inactive bits detected
READ-ONLY ADDRESS	**FUNCTION**
F0 3000 F0 3001 F0 3002	Start address, LSB Start address, 2nd byte Start address, MSB
F0 3004 F0 3005 F0 3006	End address, LSB End address, 2nd byte End address, MSB
F0 300C F0 300D F0 300E	Checksum, LSB Checksum, MSB Hex mask — inactive bits
F0 3010 F0 3014 F0 30F0	Most recent code Returns increment Most recent code

For more information about the test results, the operator may specify read operations at the special addresses listed in Table 3-5. You should first perform a READ @ ENTER to find out if the test has been completed before reading at any of the special addresses. Unless the test has been completed (or failed), the information contained at the special addresses will pertain to a previous test rather than the current test, and the current test will be aborted.

3-20. Using the 9100 Series for Quick RAM Testing

Proceed as follows to start a Quick RAM test using the 9100 Series mainframe:

1. Press the RAM key, followed by the right-arrow key if necessary to display the QUICK softkey.
2. Press the QUICK softkey, followed by ENTER.
3. Enter the beginning address, followed by the right-arrow key.
4. Enter the ending address.
5. (Optional) To select an address step other than the default of 1, press the right-arrow key, followed by the step (up to 15).
6. (Optional) To select verify instead of the default of test, press the right-arrow key; then select 2.
7. Press ENTER to start the test.
8. When the BUSY indicator goes off and no error messages are returned, the UUT has passed.

3-21. Using the 9100 Series for Quick ROM Testing

Proceed as follows to start a Quick ROM test using the 9100 Series mainframe:

1. Press the ROM key.
2. Press the TEST softkey.
3. Press the right-arrow key, then the QUICK softkey, followed by ENTER.
4. Enter the beginning address, followed by the right-arrow key.
5. Enter the ending address.
6. (Optional) To select an address step other than the default of 1, press the right-arrow key, followed by the step (up to 15).
7. Press ENTER to start the test.
8. A checksum is returned when the test is complete.

3-22. QUICK FILL AND VERIFY

Quick Fill and Verify fills blocks of memory with user-selected data, then verifies the accuracy of the contents. Quick Fill and Verify is controlled by writing setup information into special addresses as described below.

Quick Fill and Verify tests are much faster than the mainframe's normal memory tests. In addition, they allow the user to customize special memory tests, such as might be desirable when testing a memory-mapped video display.

Three variations of Quick Fill and Verify are available. The variations are specified when writing the ending address (using the 9000 Series) or entering 1, 2, or 3 at the keyboard (using the 9100 Series).

- Quick Fill (specified by 1) writes the data that is contained in the starting address to all of the addresses in the block.
- Quick Verify (specified by 2) reads data from all of the addresses in the block and compares each one to the data contained in the starting address. Errors are reported via the special addresses described below (for 9000 Series) or to the display (9100 Series).
- Quick Fill and Verify (specified by 3) combines Quick Fill and Quick Verify into one step.

3-23. Using the 9000 Series for Quick Fill and Verify

NOTE

Special addresses mentioned in the following paragraphs are also valid for 9100 Series mainframes. However, they are not necessary because the 9100 Series provides softkeys to directly access these pod functions. The recommended method of running pod-controlled quick tests from the 9100 Series is given under a separate heading.

Quick Fill and Verify tests are specified in a manner similar to the Quick RAM test. The data to be written to all addresses in the block is first written to the starting address, then the starting and ending addresses are specified by writing to special addresses. The starting address is defined by a WRITE @ 4X XXXX=0, where X XXXX is the address to be used to start the test. The ending address, address increment, and test specification are defined by a WRITE @ 4Y YYYY=ZN, where Y YYYY is the desired ending address, Z is the optional increment, and N is the test specification. If Z is not specified or is specified as 0, the increment defaults to 1. N may be either 1 (fill block), 2 (verify block), or 3 (fill and verify block). The ending address must be greater than the starting address.

For example, to specify a Quick Fill for RAM memory 0000 through 0FFF with an increment of 1 and data AA, do the following three operations:

WRITE @ 00 0000=AA (Write data AA to starting address)

WRITE @ 40 0000=0 (Specify starting address)

WRITE @ 40 0FFF=11 (Specify ending address, increment, and test)

Quick Fill or Verify begins execution as soon as the operator completes the entry of the ending address. During and after execution of the test, the mainframe will not display any information about the progress or results of the test unless requested by the operator. The test may be aborted before completion by selecting another operation.

Special addresses in the range of F0 40XX contain additional information about Quick Fill and Verify. These read-only locations contain records of the addresses used, errors, and other information. The special addresses for Quick Fill and Verify are described in Table 3-6.

To determine if Quick Fill or Verify is still in progress or what the test results are, the operator should perform a READ @ ENTER operation (which commands a READ operation at the last entered address). In response, the pod returns a byte indicating the status of the test or the test results. The status codes and their meanings are shown in Table 3-6.

For more information about the test results, the operator may specify read operations at the special addresses listed in Table 3-6. You should first READ @ ENTER to find out if the test has been completed before reading at any of the special addresses. Unless the test has been completed (or failed), the information contained at the special addresses will pertain to a previous test rather than the current test, and the current test will be aborted.

3-24. Using the 9100 Series for Quick Fill and Verify

Proceed as follows to run Quick Fill, Quick Verify, or Quick Fill and Verify using a 9100 Series mainframe.

1. Press the POD key.
2. Press the QWK_FILL softkey, followed by ENTER.

Table 3-6. Quick Fill and Verify Addresses and Status Codes

OPERATION	PARAMETERS
WRITE @ X XXXX=DD	X XXXX = Start Address DD = Fill Data
WRITE @ 4X XXXX=0 WRITE @ 4Y YYYY=ZN	X XXXX = Start Address Y YYYY = End Address Z = Increment N = Test Specification: 1 = Fill Memory 2 = Verify 3 = Fill and Verify
READ @ ENTER	Returns Status Code: 00 = No test requested A0 = Aborted, new command entered A1 = Aborted, illegal data in command A2 = Aborted, illegal address in command B0 = Busy, filling B1 = Busy, verifying C0 = Complete, no errors F0 = Failed, verify

READ-ONLY ADDRESS	FUNCTION
F0 4000	Start address, LSB
F0 4001	Start address, 2nd byte
F0 4002	Start address, MSB
F0 4004	End address, LSB
F0 4005	End address, 2nd byte
F0 4006	End address, MSB
F0 4008	Error address, LSB
F0 4009	Error address, 2nd byte
F0 400A	Error address, MSB
F0 400C	Expected data
F0 400E	Actual data
F0 4010	Most recent code
F0 4012	Hex max — bad data bits
F0 4014	Returns increment and function type
F0 40F0	Most recent code

3. Enter the beginning address, followed by the right-arrow key.

4. Enter the ending address, followed by the right-arrow key.

5. Enter the data byte in hex, followed by the right-arrow key.

6. (Optional) To select an address step other than the default of 1, press the right-arrow key, followed by the step (up to 15).

7. (Optional) To select a function other than the default of fill (1), press the right-arrow key; then enter 2 for verify or 3 for fill then verify.

8. Press ENTER to start the test.

3-25. MARGINAL UUT PROBLEMS

3-26. Introduction

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 actual microprocessor installed, but tend to exhibit errors with the pod plugged in. The pod differences tend to make marginal UUT problems more obvious and easier to troubleshoot. Different UUT and pod operating conditions that may reveal marginal problems are described in the paragraphs that follow.

3-27. UUT Operating Speed and Memory Access

UUTs designed to operate at speeds that approach the time limits for memory access may operate marginally. Inherent delays present in the pod may result in the reporting of errors in memory.

3-28. UUT Noise Levels

UUTs operate with a certain amount of noise, and as long as the noise level is low enough, it does not affect normal operation. 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 mainframe.

3-29. Bus Loading

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.

3-30. Clock Loading

The pod increases the normal load on the UUT clock. While this loading will rarely have any affect on clock operation, it may make marginal clock sources more obvious.

3-31. POD DRIVE CAPABILITY

As a driving source on the UUT bus, the pod provides equal to or better than normal Z80 current drive capability. All pod inputs and outputs (except the clock) are TTL compatible.

3-32. POWER FAILURE DETECTION LIMITS

A power sensing circuit within the pod produces a power fail output to the mainframe whenever the +5V power supply in the UUT drops below or increases above certain limits. The power failure detection limits are listed in the specifications table, Table 1-1.

Section 4
Theory of Operation

4-1. INTRODUCTION

This section contains two block diagram descriptions of the pod. The first is generalized; it describes the operating concept of the pod and the relationship of the pod to the 9000 Series or 9100 Series mainframe and the UUT. The second description covers pod operation in more detail.

4-2. GENERAL POD OPERATION

The pod may be divided into the following three major areas:

- Processor Section
- UUT Interface Section
- Timing Section

4-3. Processor Section

The Processor Section, shown in Figure 4-1, is made up of a microprocessor, RAM, a ROM, and an I/O interface to the mainframe. These elements comprise a small computer system which receives mainframe commands and directs all pod operations during execution. All reset, non-maskable interrupts, and other disrupting inputs are hardware disabled, or may be software disabled, to prevent UUT faults from disabling the pod microprocessor.

The Processor Section has the capability of operating with the mainframe, or with the UUT, but not with both concurrently. The microprocessor spends most of its time monitoring the mainframe I/O interface for commands. During this time, the data and address buses of the Processor Section are isolated from the UUT Interface Section (although the pod sends signals to the UUT so that continuous read operations at the reset address appear to be taking place in order to refresh any dynamic RAM).

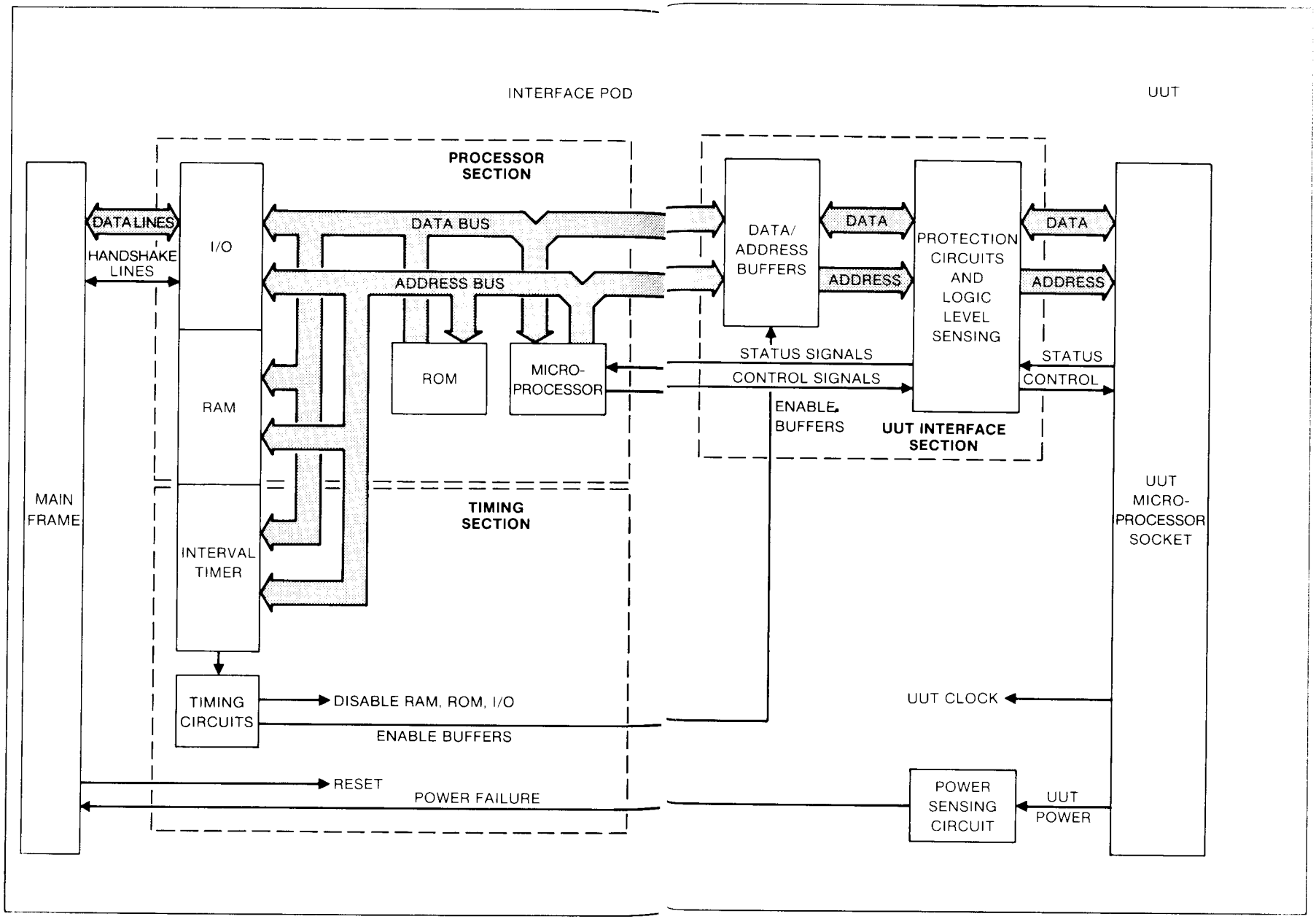


Figure 4-1. General Block Diagram

4-4. UUT Interface Section

The UUT Interface Section, shown in Figure 4-1, include the following elements:

- Data and address buffers
- Protection circuits for signal lines
- Logic level detection circuits for data, address, status and control lines

The data and address buffers are enabled to connect the microprocessor to the UUT, or disabled to isolate the microprocessor from the UUT. Control of the buffers is maintained by the timing section.

Each line to the UUT contains a protection circuit. A protection circuit consists of a 100-ohm series-resistor and clipping diodes. This circuit prevents over voltage conditions from damaging pod components.

Each line to the UUT contains a detection circuit. A detection circuit consists of a latch connected to the UUT side of the 100-ohm protection resistor. The latch senses the level at the UUT side of the protection circuit, and at the conclusion of each UUT operation, stores the level of the UUT line. Each latch is then individually addressed and read by the Processor Section. Their contents are then compared with the desired results as a means of detecting UUT bus faults.

4-5. Timing Section

The primary function of the timing section is to cause the microprocessor to work with either the Processor Section or the UUT Interface Section at a time pre-determined by the microprocessor itself. Causing the microprocessor to work with one section or the other as required during the execution of mainframe commands, permits the use of only one microprocessor in the pod.

The Timing Section of the pod, shown in Figure 4-1, consists of an interval timer and an arrangement of timing circuits. The interval timer, preset by the microprocessor, determines the time at which the microprocessor switches from addressing the Processor Section (RAM, ROM and I/O) to addressing the UUT Interface Section (and UUT). This timing is critical, since any attempt by the microprocessor to address the Processor Section with addresses meant for the UUT, or vice versa, would result in improper operation.

In their reset state, the timing circuits cause the microprocessor to operate as a part of the Processor Section, which includes an I/O port to the mainframe. When the mainframe issues a pod command which calls for a UUT read or write operation, the microprocessor sets the interval timer to a specific value. The value set on the interval timer corresponds to the time needed by the Processor Section to prepare for command execution prior to actually addressing the UUT.

When the interval timer reaches timeout, the timing circuits produce an output to disable RAM, ROM, and I/O, and to enable the buffers of the UUT Interface Section. This action causes the microprocessor to control the UUT Interface Section instead of the Processor Section. At the same time, the microprocessor, having completed preparation for command execution, places a UUT address on the address bus, and UUT data on the data bus (if the command being executed is a write).

After a fixed period of time, the timing circuits terminate the addressing of the UUT, and the microprocessor returns to controlling the RAM, ROM, and I/O elements of the Processor Section. The timing circuits also operate the latches within the logic level detection circuits to store the state of the UUT bus during the UUT bus transaction.

When the RUN UUT mode is commanded, the Timing Section causes the microprocessor to change from controlling the Processor Section to controlling the UUT Interface Section, but does not return control back to the Processor Section. In addition, the $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{WAIT}}$, and $\overline{\text{BUSRQ}}$ inputs are enabled in the RUN UUT mode. The RUN UUT mode is terminated by a reset signal from the mainframe to the pod, which returns control back to the Processor Section.

4-6. UUT Power Sensing

Figure 4-1 also shows a power sensing circuit which constantly monitors the UUT power supply. This circuit produces an output to the mainframe in the event UUT power drops below or rises above established limits. See Table 1-1.

4-7. DETAILED BLOCK DIAGRAM DESCRIPTION

The block diagram description that follows covers each of the three pod sections identified in the previous general description of pod operation. A detailed block diagram of the pod is presented in Figure 4-2.

4-8. Processor Section

Refer to Figure 4-2. The Processor Section of the pod is made up of the following components:

- Microprocessor, U5
- ROM, U2
- RAM (128 X 8), U19
- I/O ports A and B, U21-U24
- Address decoder, U7, U31
- Status line buffers, U7

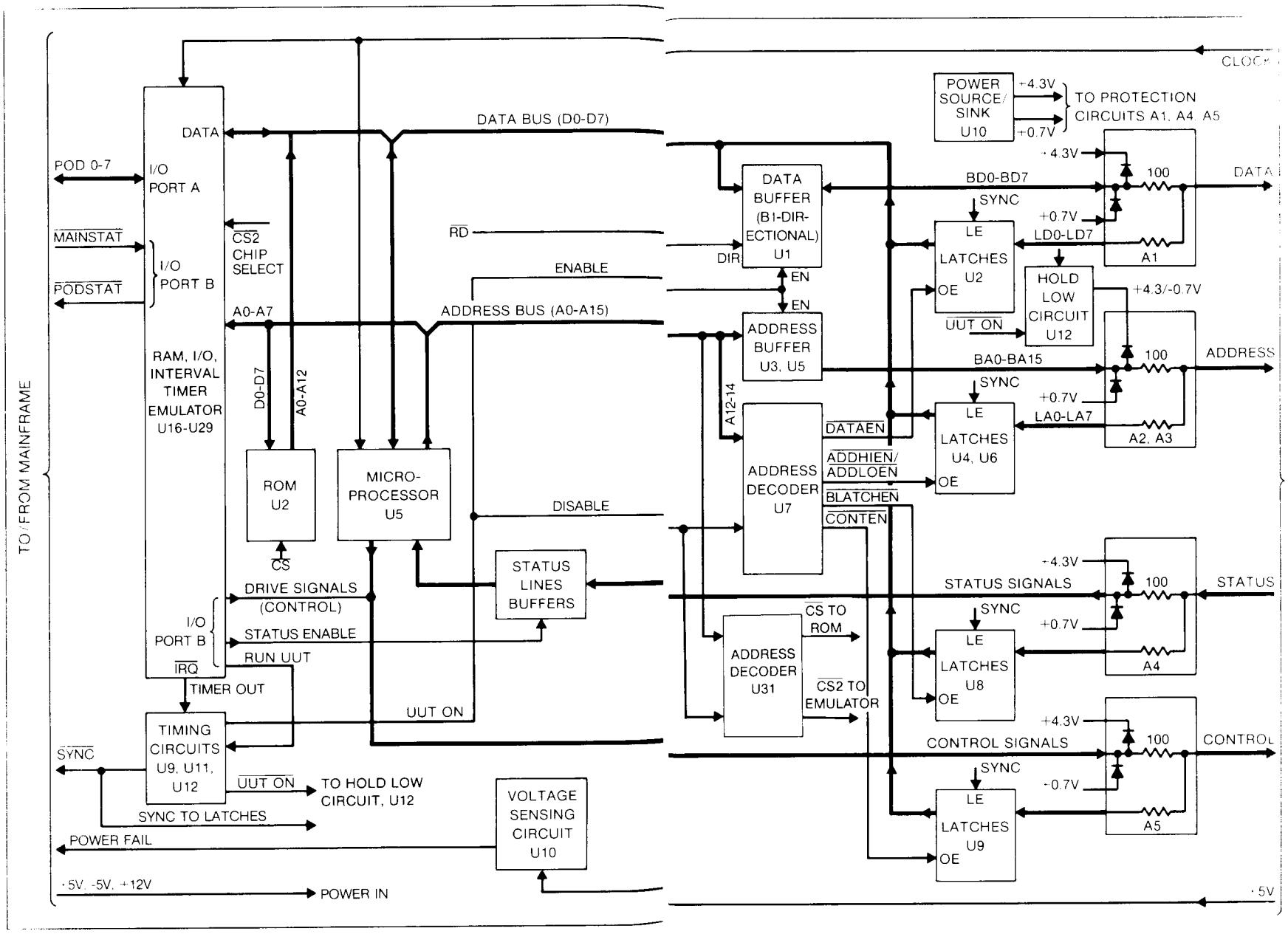


Figure 4-2. Detailed Block Diagram

The Processor monitors the handshake line, $\overline{\text{MAINSTAT}}$, at I/O port B, waiting for mainframe commands. The microprocessor addresses I/O port B by means of address lines A0-A7 and address decoder, U31. The address decoder decodes address lines A12-A14 to produce the $\overline{\text{CS2}}$ signal that selects either the RAM, I/O, or Interval Timer.

The mainframe places a low on the $\overline{\text{MAINSTAT}}$ line when a command is placed on lines POD0-7. The microprocessor responds by addressing I/O port A of the emulator and reading each byte of the mainframe command. As each byte is received, the handshaking lines operate as shown in the upper portion of Figure 4-3 to insure that no data is lost.

Each mainframe commands causes the microprocessor to execute a corresponding routine contained in ROM U2. This routine, when executed, performs the mainframe command by first setting the interval timer and then performing all necessary internal operations in preparation for addressing the UUT. For example, if the mainframe command calls for a write to the UUT, the microprocessor must perform the steps necessary to assemble the UUT address, ready the data to be written, and perform housekeeping operations associated with the command.

In addition, the routine directs the actual write and read functions of the UUT, transmits any response data back to the mainframe, and produces a status byte which reflects the current condition of the pod and UUT. During the transmission of data and status back to the mainframe, the handshake lines operate as shown in the lower portion of Figure 4-3. The handshake insures that no data is lost during the transmission process.

The microprocessor has the capability of software-driving control lines $\overline{\text{BUSAK}}$ and $\overline{\text{HALT}}$, as a means of verifying that they can be driven. Also, the microprocessor can control the enabling or disabling of status lines $\overline{\text{BUSRQ}}$ and $\overline{\text{WAIT}}$, as a means of preventing stuck UUT status lines from interfering with pod operation. Both the drive signals for the control lines and the enable signals for the status lines are written by the microprocessor through I/O port B of the emulator.

4-9. UUT Interface Section - General

Refer to Figure 4-2. The UUT Interface Section includes the following components shown in Figure 4-2:

- Bidirectional data buffer, U1
- Protection circuits, A1 - A5

- Address buffers, U3 and U5
- Sensing latches, U2, U4, U6, U8 and U9
- Hold low circuit, U12 and associated components, to hold address lines at 0000 when the UUT is not accessed
- Power source/sink U10 for protection circuits

4-10. UUT Interface Section - Data Lines

The Data Buffer U1 is disabled by the timing circuits whenever the microprocessor is controlling the Processor Section. This disabling prevents data not meant for the UUT from reaching the UUT. Conversely, the data buffer is enabled by the timing circuits when the microprocessor is not controlling the Processor Section, such as, during a UUT read/write operation via data lines BD0-7. The direction of the data buffer is controlled by the DIRIN line, a function of the microprocessor $\overline{\text{RD}}$ line.

All data passing between the pod and the UUT is fed through a series of protection circuits; one circuit per line. Each protection circuit consists of a 100-ohm resistor in series with the line, and a pair of clipping diodes. The diodes clip the data line at zero and +5 volts.

The data lines are also equipped with logic level detection circuits; one circuit per line. The detection circuits consist of a series of latches coupled to the UUT side of the respective protection circuits. A series resistor at the input of each latch provides overvoltage protection.

The data lines are coupled to the inputs of latches U2 by lines LD0-7. The input to each latch is logic high if the line is driven high, and logic low if the line is driven low. The LATCH signal from the Timing Section latches the data line logic levels, at the time shown in Figure 4-4, to store the logic levels representing the state of each data line.

At the conclusion of a UUT write operation, latch U2 are addressed by the microprocessor. Address decoder U7 produces the $\overline{\text{DATAEN}}$ signal to place the contents of the latches on the data bus. The microprocessor compares the contents of the addressed latch with the intended write data. Any difference between the contents of the latch and the intended data is considered a data error.

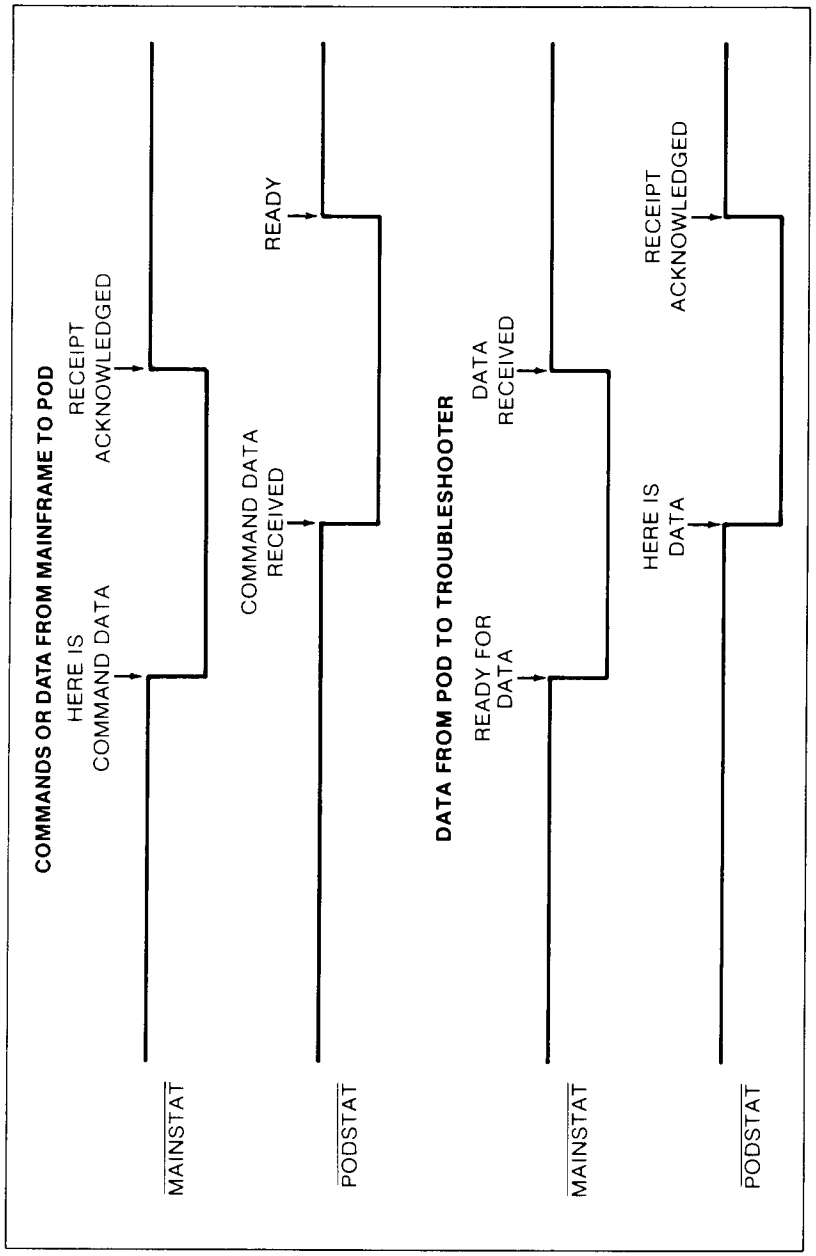


Figure 4-3. Handshaking Signals

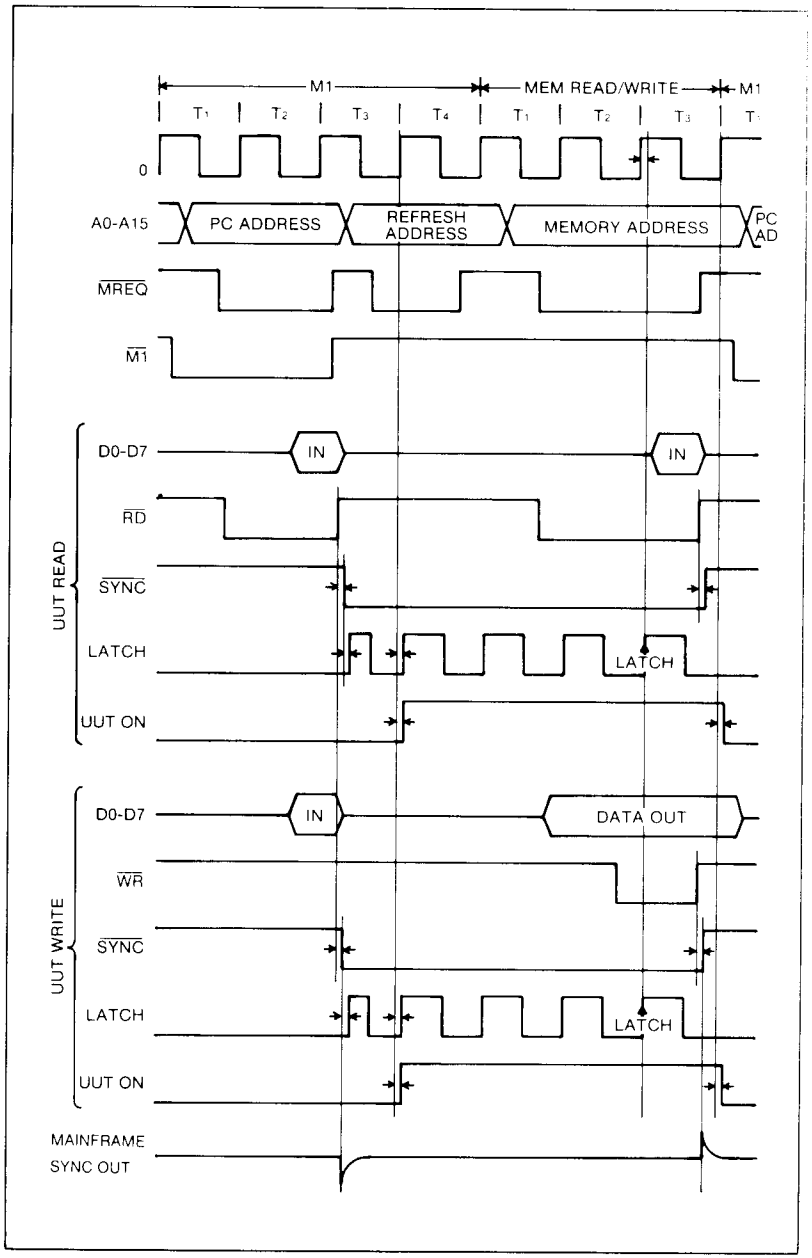


Figure 4-4. UUT ON Signal and Latch Times

4-11. UUT Interface Section - Address Lines

In a manner similar to that described for the data lines, all UUT addresses are fed through a series of protection circuits equipped with resistors and clipping diodes. The diodes used to protect the address lines perform the additional function of holding the address lines at zero volts any time the UUT Interface Section is not controlled by the microprocessor.

Address buffers U3 and U5 are enabled when the microprocessor is controlling the UUT Interface Section. Conversely, the address buffers are disabled to isolate the microprocessor from the UUT whenever the microprocessor is controlling the Processor Section. This isolation prevents the microprocessor from addressing the UUT when operating as part of the Processor Section. In addition, the address lines are held at zero volts by the diodes used in the protection circuits.

This holding action is provided by the hold low circuit, made up of U12 and associated components. This circuit drives the +4.3-volt diode clipping voltage down to -0.7 volts whenever the UUT is not being addressed, creating a UUT address of 0000. Maintaining the UUT at address 0000 prevents any inadvertent operation of the UUT and associated systems equipment.

As described for the data lines, the address lines are equipped with logic level detection circuits; one circuit per line. The detection circuits consist of a series of latches coupled to the UUT side of the respective protection circuits. A series resistor at the input of each latch provides overvoltage protection.

The address lines are coupled to the inputs of latches U4 and U6 by lines LA0-LA15. The input to each latch is logic high if the line is driven high, and logic low if the line is driven low. The LATCH signal from the Timing Section latches the address line logic levels, at the time shown in Figure 4-4, to store the logic levels representing the state of each data line.

At the conclusion of a UUT operation, latches U4 and U6 are separately addressed by the microprocessor. Address decoder U7 produces the $\overline{\text{ADDLOEN}}$ and $\overline{\text{ADDHIEN}}$ signals to place the contents of the latches on the data bus, one byte at a time. The microprocessor compares the contents of the addressed latches with the actual address. Any difference between the contents of the latches and actual address is considered an address error.

4-12. UUT Interface Section - Status and Control Lines

The status and control lines are provided with protection circuits, logic level detection circuits and latches. These circuits operate in a manner similar to those provided with the data and address lines, and described in the previous paragraphs.

4-13. Timing Section

The timing section consists of the interval timer contained in U25 and U26, and a series of timing circuits made up of U9, U11 and U12. As mentioned in the description of the Processor Section, the microprocessor executes the mainframe command by first setting the interval timer and then performing all necessary internal operations in preparation for addressing the UUT. The interval timer is set to a time equal to the amount of time required by the microprocessor to perform all necessary internal operations.

At the time the interval timer is set, and until the timer times out, a high $\overline{\text{IRQ}}$ output from the timer holds the timing circuits in their reset state. When the timer times out, the $\overline{\text{IRQ}}$ output goes low to enable the timing circuits and produce the $\overline{\text{SYNC}}$ signal. At the time of the next clock pulse, the timing circuits produce a high $\overline{\text{UUT ON}}$ signal to enable the data and address buffers, and disable the address decoder. Refer to Figure 4-4 for the timing of the $\overline{\text{UUT ON}}$ signal. A corresponding low $\overline{\text{UUT ON}}$ signal disables the hold low circuit, U12, releasing the address bus from the forced UUT address of 0000.

With the data and address buffers enabled, and the hold low circuit disabled, data and addresses placed on the buses by the microprocessor are directed to the UUT. The Processor Section is disabled at this point by the address decoder U7 and U31, which receives the UUT ON signal generated by the timing circuits. With the address decoder disabled, the ROM and RAM-I/O-Interval Timer are not selected.

At the end of the instruction cycle (except in the RUN UUT mode), the timing circuits return to their reset state to disable the data and address buffers and enable address decoder, U7 and U31. This action switches the microprocessor back to control the Processor Section instead of the UUT Interface Section.

Just prior to returning to their reset state, while all UUT lines are stable, the timing circuits terminate the $\overline{\text{SYNC}}$ signal to latch the UUT line logic levels. The latches store the condition of all UUT lines. When addressed by the microprocessor, via address decoder U7, each latch places the condition of the associated UUT line on the data bus. The microprocessor compares the detected UUT line levels with the known expected result and considers any difference to be an error. Any error conditions are indicated in the status byte sent to the mainframe at the conclusion of each command.

When RUN UUT starts and the interval timer produces the low $\overline{\text{IRQ}}$ signal, and the timing circuits produce the high UUT ON signal as previously described for the non-RUN UUT mode. However, the RUN UUT command causes the timing circuits to be held in a state which maintains the $\overline{\text{UUT ON}}$ signal and dedicates the microprocessor to the UUT. In this mode, the $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{BUSRQ}}$, and

$\overline{\text{WAIT}}$ inputs are enabled, allowing the UUT to utilize the pod microprocessor in place of the microprocessor removed to facilitate pod connection.

The RUN UUT mode continues until a $\overline{\text{RESET}}$ signal is received from the mainframe. The $\overline{\text{RESET}}$ signal from the mainframe causes the microprocessor to resume control of the Processor Section.

Section 5
Maintenance

5-1. INTRODUCTION

This section provides maintenance information for the pod, and includes self test information, repair precautions, disassembly procedures, and trouble-shooting information.

5-2. SELF TEST

The 9000 Series or 9100 Series mainframe can perform a self test on any pod which is operational enough to communicate with the mainframe. Self test provides fault location to several areas of the pod by creating appropriate display messages on the mainframe. In order to perform self test, the Processor Section (Z80, RAM, ROM, I/O, and buses) must be operational. Operation of the processor section is necessary in order for the pod to accept and execute self test commands issued by the mainframe.

NOTE

Self test does not examine the pod for all conceivable faults, and may indicate an okay pod when not completely operable. An alternative method of checking pod operability is exercising with a known-good UUT and mainframe, observing any reported UUT failures.

Performance of self test requires that the ribbon cable connector be inserted into the self test socket located on the pod. When the ribbon cable plug is inserted into the self test socket, the following electrical connections are made to facilitate testing (refer also to the schematic diagram contained in Section 7):

- The high order address lines are connected back to the data lines through series resistors. This connection allows the high order address bits to become data during a test read operation.

- An 8 MHz clock signal is applied to the clock input of the pod. This clock signal replaces the clock normally supplied by the UUT to operate the pod.
- All forcing lines and interrupts are set to the active state. Setting these lines allows testing of the individual hardware or software buffering.
- +5V dc is applied to pin 11 to simulate UUT power and check the power fail sensing circuit.
- Ground is applied through the ribbon cable to pin 29 to notify the mainframe that the pod is in the self test configuration.

To perform self test with a 9000 Series, proceed as follows:

1. If not already connected, connect the interface pod to the mainframe as shown in Figure 2-1. Secure the connector using the sliding collar.

2. Open the pins of the self test socket by operating the adjacent thumbwheel. Insert the ribbon cable plug into the socket and close the socket using the thumbwheel.

3. Turn the mainframe on and press BUS TEST to initiate self test.

4. If the mainframe and pod are operating normally, the mainframe display reads POD SELF-TEST Z80QT OK.

5. If the pod is defective but not completely dead, the mainframe displays POD SELF-TEST Z80QT FAIL xx, where xx represents the pod fault listed in Table 5-1. Refer to the troubleshooting procedures to further isolate the problem.

6. If the pod is inoperative, the mainframe reads POD TIMEOUT -ATTEMPTING RESET. This message indicates that the pod is not responding to commands issued by the mainframe. Refer to the troubleshooting procedures to isolate the problem.

To perform self test with a 9100 Series, proceed as follows:

1. If not already connected, connect the interface pod to the mainframe as shown in Figure 2-2. Secure the connector using the sliding collar.

Table 5-1. Self Test Failure Code Summary

9000-SERIES POD SELF TEST	
FAILURE CODE	**DESCRIPTION**
00	UUT read access failed
01	UUT write access failed
02	Control lines cannot be driven
03	Enableable status line(s) failed
9100-SERIES POD SELF TEST	
FAILURE CODE	**DESCRIPTION**
0	Read access failure
1	Write access failure
2	Control lines failure
3	Disable enableable lines failure
4	Sync mode failure
5	Address lines failure
6-13	Data lines failure
40-45	Address space access failures
2001	Not in self test
2002	Unexpected powerfail
2006	Unexpected pod timeout
2007	Unexpected response when disabling enableable lines

2. Open the pins of the self test socket by operating the adjacent thumbwheel. Insert the ribbon cable plug into the socket and close the socket using the thumbwheel.

3. Turn the mainframe on, then press the MAIN MENU key and use the left-arrow key to move the cursor to the left-most field.

4. Press the SELFTEST softkey.

5. Move the cursor to the next field and press the POD softkey.

6. The display should read:

   MAIN: SELFTEST POD

7. Press the ENTER key to perform the self-test. If the self test fails, a failure message is displayed. If a POD TIMEOUT message is displayed, the pod is inoperative. If another failure, in the form POD SELFTEST FAILURE = n appears, refer to Table 5-1 for the meaning of the failure code.

5-3. REPAIR PRECAUTIONS

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 mainframe.
- Perform all repairs at a static-free work station.
- Do not handle ICs or PCB assemblies by their connectors.
- Wear a static ground strap.
- Use conductive foam to store replacement or removed ICs.
- Remove all plastic, vinyl, and styrofoam from the work area.
- Use a grounded soldering iron.

The soldering iron used in pod repair should have a rating of 25 watts or less to prevent overheating the PCB assembly.

5-4. TROUBLESHOOTING

5-5. Introduction

Pod failure is usually identifiable from the mainframe display. Two types of messages which indicate pod failure are:

- A pod timeout message. When a pod timeout message is displayed, the pod does not respond to mainframe commands or reset pulses. This type of message may also be due to stuck forcing lines not disabled during mainframe setup procedures described in the mainframe manual.
- Any recurring UUT test failure or error message when testing a known-good UUT indicates pod failure. Since the UUT is known to be good, errors attributed to the UUT by the mainframe are actually pod errors.

Troubleshooting the pod is similar to troubleshooting any other microprocessor-based UUT; and requires the equipment listed in Table 5-2. The troubleshooting

information presented in the following paragraphs does not provide step-by-step fault isolation procedures, but provides a troubleshooting guide for use while employing normal fault isolation techniques.

Figure 5-1 shows the non-component side of the interface PCB with component outlines and identification superimposed. Refer to this figure to locate various electrical points on the interface PCB during troubleshooting procedures.

The troubleshooting information should be used in conjunction with the schematic diagrams contained in Section 7 and the Theory of Operation presented in Section 4.

The troubleshooting guidelines presented in the following paragraphs are intended to assist in the isolation of faults within the pod. If attempted troubleshooting fails to reveal the pod fault, return of the pod to the nearest Fluke Service Center is recommended. Refer to the mainframe Service Manual for a list of Fluke Service Centers.

5-6. Pod Defective or Inoperative?

Before attempting to repair a faulty pod, the level of failure should be determined. A faulty pod can be categorized as either defective or inoperative, depending upon the result of the self test.

If the pod fails the self test, but does not timeout, the pod is considered to be defective but not inoperative. Troubleshoot a defective pod as described under the heading Troubleshooting a Defective Pod. Select a suitable UUT as described under the heading Selecting a UUT for Pod Testing.

NOTE

It is possible for a pod to pass a self test and still be faulty. Such a pod causes the display of test failure or error messages on the mainframe when used to test a known-good UUT. In this case, errors attributed to the UUT are actually pod errors.

Table 5-2. Required Test Equipment

EQUIPMENT TYPE	REQUIRED TYPE
Digital Test System Mainframe	Fluke 9000 or 9100 Series
Interface Pod	Fluke 9000A-Z80QT
Digital Multimeter	Fluke 77 or equivalent
Oscilloscope	Philips PM 3065 or equivalent

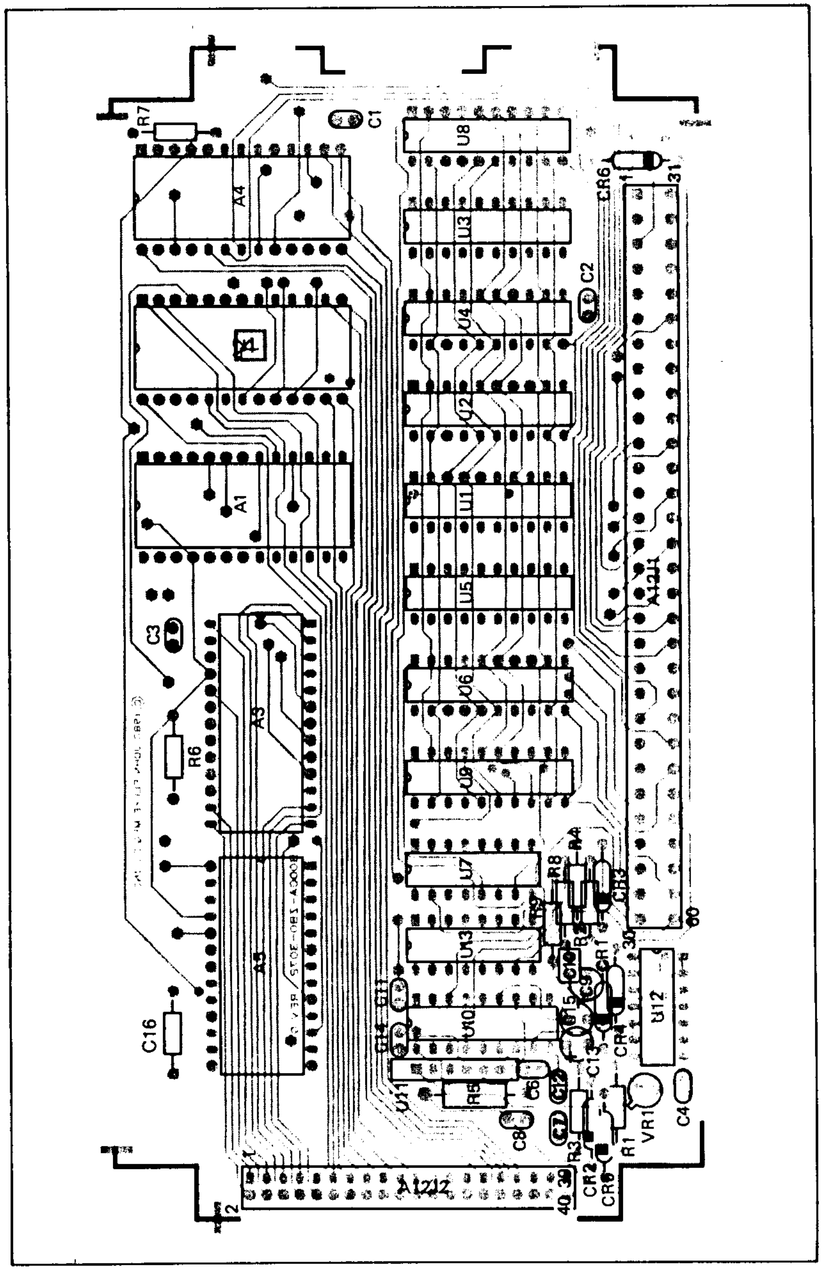


Figure 5-1. Interface PCB, Non-Component Side

If the result of a self test, or any other mainframe operation produces a pod timeout message, the pod is considered to be inoperative. Troubleshoot an inoperative pod as described under the heading Troubleshooting an Inoperative Pod. Select a suitable UUT as described under the heading Selecting a UUT for Pod Testing.

NOTE

The pod timeout message can also result from stuck UUT forcing lines which can disable the pod. Forcing lines can be disabled during mainframe setup procedures as described in the mainframe manual.

5-7. Selecting a UUT for Pod Testing

In order to troubleshoot a pod, a known-good UUT must be connected to the pod via the ribbon cable and connector. The UUT may be any device which normally employs a Z80 microprocessor and to which power can easily be applied. The UUT is needed to provide the following functions during pod testing:

- RAM and ROM for performing read/write operations
- Z80 compatible clock signal to drive the pod
- +5V dc UUT power to check the UUT power sensing circuit

Instead of connection to a known-good UUT, the ribbon cable connector may be connected to the self test socket on the pod. The self test socket provides a Z80 compatible clock signal, +5V dc, and also simulates ROM by connecting the high order address lines back to the data lines (refer to the schematic diagram for details).

However, insertion of the ribbon cable connector directly into the self test socket places pin 29 at ground. The pod senses the ground at pin 29 and notifies the mainframe of the self test connection. As a result, the mainframe inhibits normal operation and only allows self testing.

During pod troubleshooting procedures, normal mainframe operation must be allowed. Consequently, the pod must be prevented from sensing the ribbon cable connector in the self test socket. To prevent the pod from sensing the self test connection, pin 29 of the connector must be effectively removed.

To effectively remove pin 29 of the connector, obtain a 40-pin IC socket or connector and modify it as follows:

1. Remove pin 29 from the IC socket.

2. Insert the modified IC socket into the self test socket.

3. Insert the ribbon cable connector into the modified IC socket.

In addition to effectively modifying the ribbon cable connector, be sure to disable all forcing line and interrupt inputs, and set all forcing line and interrupt traps to NO during setup editing as described in the mainframe manual. Disabling these inputs and messages is necessary when utilizing the self test socket since all lines are wired to the active state.

5-8. Troubleshooting a Defective Pod

NOTE

The following paragraphs reference three distinct areas of the pod identified as the Processor Section, the UUT Interface Section, and the Timing Circuits. The components which make up these sections are identified in the Theory of Operation, presented in Section 4.

A pod is considered defective when it fails a self test. Pod faults are summarized in Table 5-1. For more detailed self testing error analysis, refer to Table 5-3. The fact that a self test can be performed indicates operation of the Processor Section, since operation of the Processor Section is necessary for mainframe/pod communication. With the Processor Section proven to be good, the UUT Interface Section or the Timing Circuits contain the fault.

Prepare to troubleshoot the defective pod as follows:

1. Disassemble the pod by removing the PCB assemblies from the case, and the shield from the PCB assemblies. (Refer to disassembly information under the heading Disassembly.) It is not necessary to separate the PCB assemblies at this point.

2. Connect the pod to the mainframe, and the ribbon cable connector to the UUT, as shown in Figure 5-2. Note that the mainframe is connected by means of the shielded cable, and not by means of a second pod connected to the microprocessor socket. Also, Figure 5-2 shows the self test socket as the UUT, although any suitable UUT may be used. (Refer to Selecting a UUT for Pod Testing.)

NOTE

All references to data and addresses in the following troubleshooting guide are in hexadecimal notation. Unless otherwise noted, all mainframe probe operations are performed in the synchronized mode.

Table 5-3. Recreating Self Test Routines

9000-SERIES POD SELF TEST		
FAILURE CODE	**POD OPERATION**	**OPERATOR ACTIONS TO RECREATE TEST**
00	Reset Pod	Cycle Power
	READ @ 0FF0 = 0F	READ @ 0FF0 (If a powerfail error message occurs, check the power detection circuits.)
NOTE 01	WRITE @ 0FF0 = 0F	WRITE @ 0FF0 = 0F
02	Test control lines	BUS TEST
03	Send command to enable each enableable line and verify that a pod timeout occurs. (This timeout is transparent to the user.)	Enable BUSRQ and WAIT separately; verify that a timeout occurs when either line is enabled.

9100-SERIES POD SELF TEST		
FAILURE CODE	**POD OPERATION**	**OPERATOR ACTIONS TO RECREATE TEST**
0	Reset Pod	Cycle Power
	Read special address 0FF00FF0	READ SPECIAL ADDR 00FF0 (If a powerfail error message occurs, check the power detection circuits.)
NOTE 1	Write data 0F, Special address 0FF00FF0	WRITE DATA 0F TO SPECIAL ADDR 00FF0
2	Test control lines	TEST BUS AT ADDR FFF
3	Disable all enableable	SETUP POD ENABLE BUSRQ OFF SETUP POD ENABLE WAIT OFF
4	Sync on ADDR	SYNC PROBE TO POD ADDR
5	Read special address 0000F00F	READ SPECIAL ADDR OF 00F
6	Write data FE, special address 00000001	WRITE DATA FE TO SPECIAL ADDR 00001
7	Write data FD, special address 00000002	WRITE DATA FD TO SPECIAL ADDR 00002

Table 5-3. Recreating Self Test Routines (cont)

9100-SERIES POD SELF TEST		
FAILURE CODE	**POD OPERATION**	**OPERATOR ACTIONS TO RECREATE TEST**
8	Write data FB, special address 00000004	WRITE DATA FB TO SPECIAL ADDR 00004
9	Write data F7, special address 00000008	WRITE DATA F7 TO SPECIAL ADDR 00008
10	Write data EF, special address 00000010	WRITE DATA EF TO SPECIAL ADDR 00010
11	Write data DF, special address 00000020	WRITE DATA DF TO SPECIAL ADDR 00020
12	Write data BF, special address 00000040	WRITE DATA BF TO SPECIAL ADDR 00040
13	Write data 7F, special address 00000080	WRITE DATA 7F TO SPECIAL ADDR 00080
38	Enable first enableable line, verify pod timeout occurs	SETUP POD ENABLE BUSRQ ON
	Disable first enableable line again	SETUP POD ENABLE BUSRQ OFF
39	Enable second enableable line, verify pod timeout occurs.	SETUP POD ENABLE WAIT ON
	Disable second enableable line again	SETUP POD ENABLE WAIT OFF
40	Write data FF, special address 00000000	WRITE DATA FF TO SPECIAL ADDR 00000
41	Write data FF, special address 0001FFFF	WRITE DATA FF TO SPECIAL ADDR 1FFFF
42	Write data FF, low address of MEMORY space	WRITE DATA FF TO ADDR 0000 ADDR OPTION: MEMORY
43	Write data FF, high address of MEMORY space	WRITE DATA FF TO ADDR FFFF ADDR OPTION: MEMORY
44	Write data FF, low address of I/O space	WRITE DATA FF TO ADDR 0000 ADDR OPTION: I/O

Table 5-3. Recreating Self Test Routines (cont)

9100-SERIES POD SELF TEST		
FAILURE CODE	**POD OPERATION**	**OPERATOR ACTIONS TO RECREATE TEST**
45	Write data FF, high address of I/O space	WRITE DATA FF TO ADDR FFFF ADDR OPTION: I/O
2001	Pod UUT cable connector in self test socket	None; check that pod is in self test and the internal SELFTEST signal is pulled low at the self test socket by UUT cable pin 29 (GND).
2002	VCC pin on self test is high (no power-fail)	None; check VCC line through the UUT cable, and the POWERFAIL signal.
2006	Pod timeout not expected	None; troubleshoot as a defective pod.
2007	BUSRQ and WAIT are disabled	Disable enableable lines. Pod should not report timeout.

NOTE: For more information on these error codes, see the text.

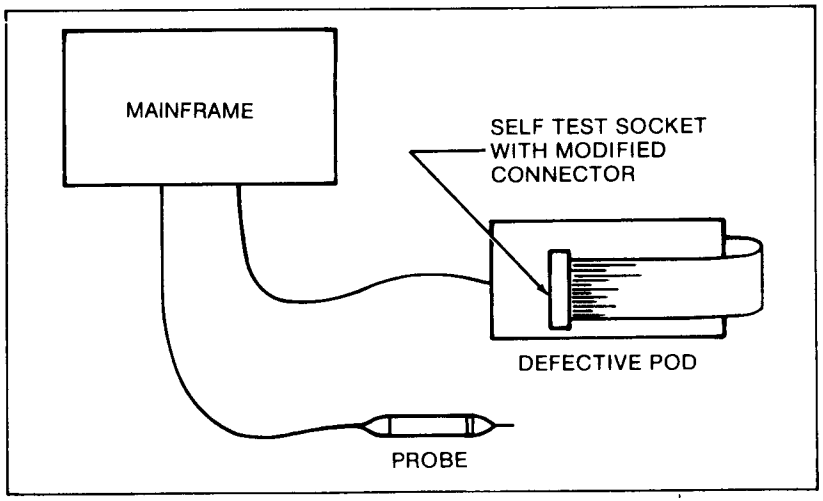


Figure 5-2. Troubleshooting a Defective Pod

NOTE

When troubleshooting a pod, perform looping tests of the most simple type (such as reads and writes as opposed to ROM and RAM tests) to reveal a fault symptom. A synchronized probe can then be used to trace a fault once such a looping test has it isolated.

5-9. SELF TEST CODE 0

If a self test produces a failure code of 0, a UUT read operation has failed, and one or more of the following problems is indicated:

- UUT power sensing circuit failure
- Control line(s) cannot be driven
- Address line(s) cannot be driven
- Wrong data read

To further isolate the trouble, proceed as follows:

1. Check operation of the UUT power sensing circuit by verifying the +5V UUT supply at the ribbon cable connector and 0V on the Power Fail line. Check the Power Fail line at the PCB-to-PCB connector, and if necessary, at the shielded cable connector.

2. Perform a read operation. Use address 0FF0 if using the self test socket as the UUT. (The self test socket sends the upper address byte to the data lines. During self test, read operations at 0FF0 and F00F take place.) Use any address containing known data if using some other UUT.

   a. If the mainframe indicates a control line error, examine the entire mainframe display to determine the stuck control line(s). While looping on the error, use the probe or a scope to locate the point of control line failure.

   b. If the mainframe indicates an address line error, note the failed address line(s) indicated on the mainframe display. While looping on the error, use the probe or a scope to locate the point of address line failure.

   c. If the data read, indicated on the mainframe display, is not 0F when using the self test socket, or is not identical to the known data of the UUT used for this test, a data line or address line failure is indicated. Determine the failed line(s) from the display and locate the point of failure using the synchronized probe or a scope while performing a looping read operation.

3. Repeat steps 2b and 2c at different addresses and for different data in order to toggle each of the address and data lines.

4. Check for operation of the interval timer and timing circuits by observing pin 18 of U20 ($\overline{IRQ}$) for a low-going output each time a read operation is executed. If the IRQ signal is present, check for a $\overline{SYNC}$ signal at pin 10 of the shielded cable connector, and for a UUT ON signal at pin 19 of the PCB-to-PCB connector. The absence of these signals allows the pod to communicate with the mainframe, but prevents the latches from detecting addresses, data, and control signals sent to the UUT (or self test socket). Failure of these signals may also prevent data read from the UUT from reaching the pod microprocessor.

5-10. SELF TEST CODE 1

If a self test produces a failure code of 1, one or more of the following failures is indicated:

- UUT power sensing circuit failure
- Control line(s) cannot be driven
- Address line(s) cannot be driven
- Data line(s) cannot be driven

To further isolate the trouble, proceed as follows:

1. Check the operation of the UUT power sensing circuit by verifying the +5V UUT supply at the ribbon cable connector and 0V on the Power Fail line. Check the Power Fail line of the PCB-to-PCB connector, and if necessary, at the shielded cable connector.

2. Perform a write operation; use 0FF0 for the address and 0F for the data.

   a. If the mainframe indicates a control line error, examine the entire mainframe display to determine the stuck control line(s). While looping on the error, use the probe or a scope to locate the point of control line failure.

   b. If the mainframe indicates an address line error, note the failed address line(s) indicated on the mainframe display. While looping on the error, use the probe or a scope to locate the point of address line failure.

   c. If the mainframe indicates a data line error, note the failed line(s) indicated on the mainframe display. While looping on the error, use the probe or a scope to locate the point of failure.

3. Repeat steps 2b and 2c using F00F for the address and F0 for the data to check address and data lines in the opposite state.

4. Check for operation of the interval timer and timing circuits by observing pin 18 of U20 ($\overline{IRQ}$) for a low-going output each time a write operation is executed. If the $\overline{IRQ}$ signal is present, check for a $\overline{SYNC}$ signal at pin 10 of the shielded cable connector, and for a UUT ON signal at pin 19 of the PCB-to-PCB connector. The absence of these signals allows the pod to communicate with the mainframe, but prevents the latches from detecting addresses, data, and control signals sent to the UUT (or self test socket). Failure of these signals may also prevent write data from reaching the UUT.

5-11. SELF TEST CODE 2

If a self test produces a failure code of 2, failure of one or more of the control lines is indicated. To check each of the control lines, use the mainframe to perform a BUS TEST. Refer to the heading Bit Assignment -Control Lines, located in Section 3, for interpretation of the mainframe message.

5-12. SELF TEST CODE 3 (APPLIES TO 9000 SERIES ONLY)

If a self test produces a failure code of 3, failure of one or more status line buffers is indicated. Each of the status (forcing) lines, which have the ability to interrupt or otherwise interfere with microprocessor operation, are selectively buffered from the microprocessor.

Buffering of the $\overline{RESET}$, $\overline{NMI}$, $\overline{BUSRQ}$, and $\overline{WAIT}$ lines is accomplished by means of gates which are enabled or inhibited by port B outputs of the RAM-I/0-Interval Timer.

5-13. Troubleshooting an Inoperative Pod

NOTE

The following paragraphs reference three distinct areas of the pod identified as the Processor Section, the UUT Interface Section, and the Timing Circuits. Components that make up these sections are identified in the Theory of Operation, presented in Section 4.

A pod is considered inoperative when the performance of self test, or any other mainframe operation, produces a pod timeout message. Such a message results from a lack of response by the pod to mainframe commands. Since it is the function of the Processor Section to respond to mainframe commands, lack of response indicates failure of the Processor Section.

Prepare to troubleshoot the inoperative pod as follows:

1. Disassemble the pod. Refer to Disassembly.

2. Remove the microprocessor from its socket.

3. Connect the pod under test to +5V and -5V power supplies. Apply power to the connector normally coupled to the mainframe; use pins 2 and 15 for +5V, pin 21 for -5V, and pin 25 for ground. If available, use a second mainframe and shielded cable to provide self test power to the pod.

4. Connect a mainframe to a second pod. Apply power to the mainframe, then connect the second pod ribbon cable to the microprocessor socket of the pod under test.

CAUTION

Do not apply or remove any power with ribbon cable connected between second pod and inoperative pod.

NOTE

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

Refer to the Theory of Operation in Section 4 and the schematic diagram in Section 7 to troubleshoot an inoperative pod using the following steps as a guide:

1. Reset the pod by momentarily shorting pins 22 and 23 of the shielded cable connector located on the upper PCB assembly.

2. Perform a BUS test. Because of a contention situation, do not use the default address. Instead, set the BUS test address to 2000.

3. Perform a RAM test. The RAM addresses are listed in Table 5-4.

4. Perform a ROM test. The ROM addresses are listed in Table 5-4. The ROM checksum returned by a Quick ROM test of the pod ROM is given in Table 5-5.

5. Check the output operation of I/O port A (contained in U21 and U23) as follows:

   a. Perform a write operation to the port A direction register to set all lines of I/O port A (PA0-PA7) as outputs. The write address is 2081; write data is FF.

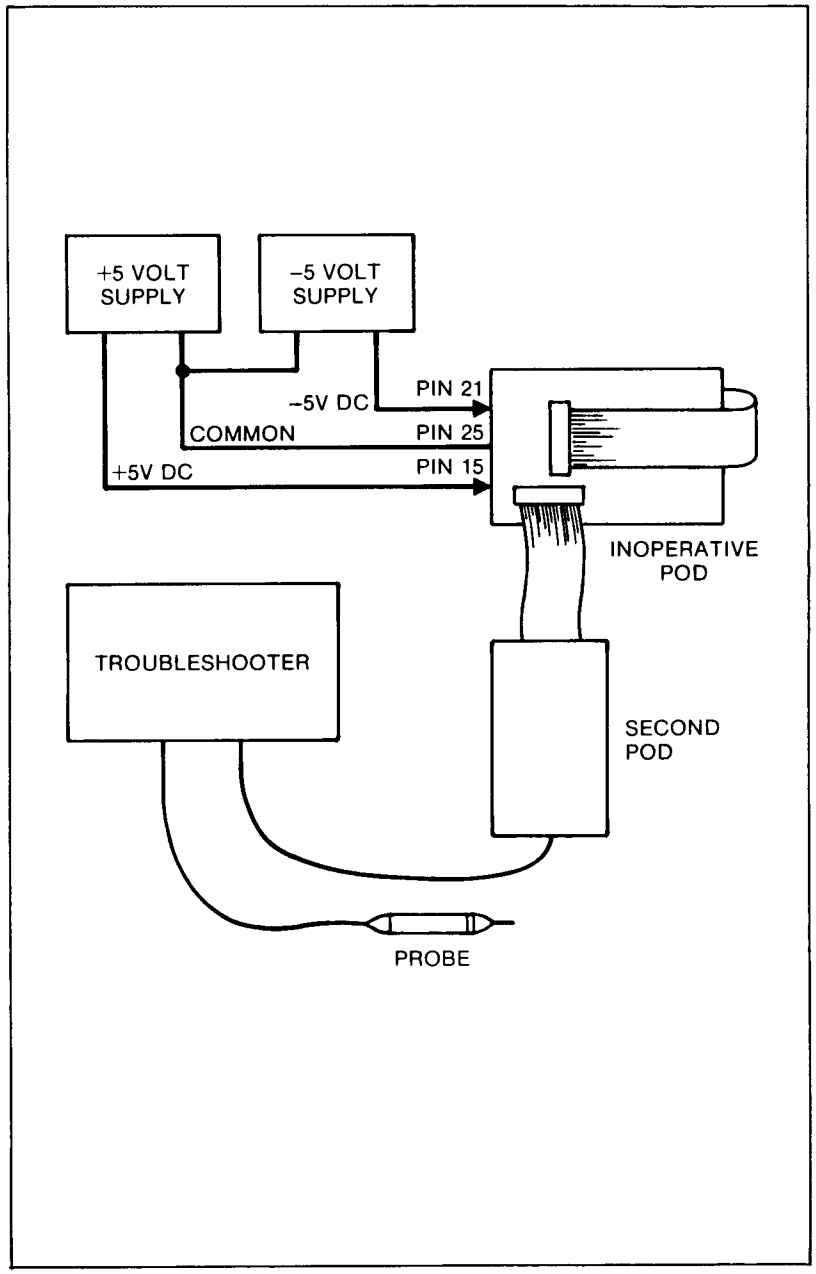


Figure 5-3. Troubleshooting an Inoperative Pod

Table 5-4. Z80QT Pod Memory and I/O Addresses

ADDRESSABLE DEVICE	ADDRESS (HEX)
RAM	2000 - 207F
ROM	0000 - 1FFF (See Table 5-5)
I/O-Port A Direction Register	2081
-Port A Data Register	2080
-Port B Direction Register	2083
-Port B Data Register	2082
Interval Timer - Divide by 1	209C
Interval Timer Disable	2094
UUT Address line latch (high byte)	3000
UUT Data line latch	4000
UUT Control line latch	5000
UUT Address line latch (low byte)	6000
UUT Status line latch	7000

Table 5-5. Z80QT Interface Pod Quick ROM Checksum

ROM ADDRESS RANGE	CHECKSUM	SOFTWARE VERSION
0000 to 1FFF	1822	1.0

b. Perform a write operation to the port A data register to set all bits high. The write address is 2080; write data is FF.

c. Check the port A lines (PA0-PA7) with the probe or scope for all logic high levels.

d. Repeat step b with 00 as the write data.

e. Repeat step c, checking for all logic low levels.

6. Check the input operation of I/O port A (contained in U21 and U23) as follows:

a. Perform a write operation to the port A direction register to set all lines of I/O port A (PA0-PA7) as inputs. The write address is 2081; write data is 00.

b. Perform a read operation at the port A data register, address 2080, while sequentially applying both +5V and ground to each of the port A input pins (pins 2-9 of U21) and observing the mainframe display. The mainframe should indicate each high and low applied to the inputs of port A.

7. Check the output operation of port B (contained in U22 and U24) by repeating step 5 and using address 2083 for the port B direction register, and address 2082 for the port B data register.

8. Check the input operation of port B, line PB7 ($\overline{\text{MAINTSTAT}}$) by repeating step 6. Use address 2083 for the port B direction register and write data 00 to set line PB7 as an input. Perform the looping read at address 2082, and apply both +5V and ground to pin 12 of the shielded cable connector.

9. Check operation of the interval timer (contained in U25 and 26) by first writing 00 to address 2094 to reset the timer, then writing 0F to address 209C. Verify that the $\overline{\text{IRQ}}$ output at U20, pin 18, goes low in response to the second write operation.

10. Check for the occurrence of the UUT ON signal (produced by the timing circuits as a result of the low $\overline{\text{IRQ}}$ signal) at pin 19 of the PCB-to-PCB connector, by repeating step 9.

11. Check for the occurrence of the $\overline{\text{SYNC}}$ signal at pin 10 of the shielded cable connector, by repeating step 9.

12. Check both address decoders by performing read operations at addresses 0000, 2000, 3000, 4000, 5000, 6000, and 7000. Verify that the respective decoder output goes low when addressed.

13. If repairs have been made to the inoperative pod as a result of the preceding checks, reinstall the pod's microprocessor and attempt self test. If self test operates, but the pod fails, refer to Troubleshooting a Defective Pod.

The troubleshooting guidelines presented in the preceding paragraphs are intended to assist in the isolation of faults within the pod. If attempted troubleshooting fails to reveal the pod fault, return of the pod to the nearest Fluke Service Center is recommended. Refer to the mainframe Service Manual for a list of Fluke Service Centers.

5-14. DISASSEMBLY

To gain access to the two PCB assemblies within the pod, proceed as follows.

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

2. Remove the four Phillips-head screws holding the pod case halves together and carefully open the case.

3. With the PCB assemblies removed from the case halves, remove the screw which retains the shield. Remove the shield.

NOTE

To troubleshoot the pod, it may not be necessary to separate the two PCB assemblies except to replace components. Figure 5-1 shows the location of each component on the lower PCB assembly relative to the accessible non-component side of the board.

4. If it is not necessary to separate the two PCB assemblies, temporarily replace the shield retaining screw; otherwise, remove the other screws from their standoffs and carefully pull the boards apart at the connector.

5. To operate the pod with the two printed circuit boards separated from each other, reconnect them in a side-by-side fashion using the test adapter, Fluke part no. 613828. Make sure that correct pin-to-pin relationships are maintained.

Section 6
List of Replaceable Parts

6-1. INTRODUCTION

This section contains an illustrated parts breakdown of the instrument. Components are listed alphanumerically by assembly.

Parts lists include the following information:

1. Reference Designation.

2. Description of Each Part.

3. FLUKE Stock Number.

4. Federal Supply Code for Manufacturers. (See the 9000 Series Troubleshooter Service Manual for Code-to-Name list).

5. Manufacturer's Part Number.

6. Total Quantity of Components Per Assembly.

7. Recommended quantity: This entry indicates the recommended number of spare parts necessary to support one to five instruments for a period of 2 years. This list presumes an availability of common electronic parts at the maintenance site. For maintenance for 1 year or more at an isolated site, it is recommended that at least one of each assembly in the instrument be stocked.

6-2. HOW TO OBTAIN PARTS

Components may be ordered directly from 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 REC QTY column for 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.

CAUTION

★

Indicated devices are subject to damage by static discharge.

Table 6-1. 9000A-Z80QT Interface Pod Final Assembly
(See Figure 6-1.)

REFERENCE DESIGNATOR -A>-NUMERICS---->	S	DESCRIPTION----------	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	N R O S T -Q -E-
A 11	*	PROCESSOR PCB ASSEMBLY	859777	89536	859777	1	
A 12	*	INTERFACE PCB ASSEMBLY	582155	89536	582155	1	
H 1		SCREW,MACH,PH,P,STL,4-40X0.500	152132	89536	152132	4	
H 2		SCREW,MACH,PHP SEMS,STL,4-40X1/4	185918	89536	185918	3	
MP 3		LABEL, STATIC CAUTION	605808	89536	605808	1	
MP 4		LABEL, UUT CAUTION	634030	89536	634030	1	
MP 5		SHELL, TOP	579565	89536	579565	1	
MP 6		SHELL, BOTTOM	579573	89536	579573	1	
MP 7		ACTUATOR	582916	89536	582916	1	
MP 8		DECAL, POD	859418	89536	859418	1	
MP 9		DECAL, SPEC	536706	89536	536706	1	
MP 10	*	SHIELD,ALUM MYLAR	586479	89536	586479	1	
MP 11		ACCESSORY KIT 40PIN BASIC	648907	89536	648907	1	
MP 12		PROGRAMMED DISK	859421	89536	859421	1	
P 1		CABLE, POD W/RESISTOR	581819	89536	581819	1	
TM 1		9000A-Z80QT MANUAL	859447	89536	859447	1	
TM 2		Z80QT QUICK REFERENCE CARD	859462	89536	859462	1	
TM 3		DISK LOADING INSTRUCTIONS	863394	89536	863394	1	
W 1		CABLE, UUT SHIELDED 40 POS	685461	89536	685461	1	

An * in 'S' column indicates a static-sensitive part.

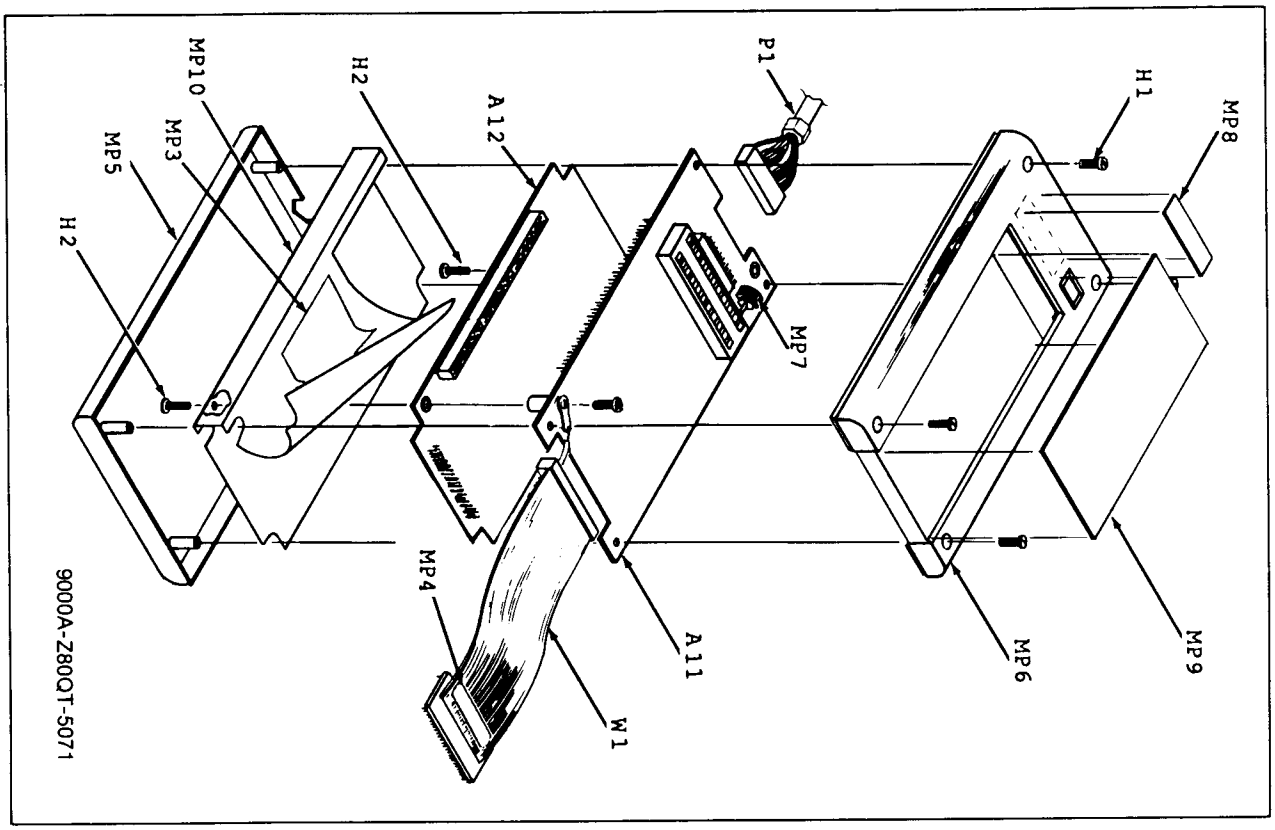


Figure 6-1. 9000A-Z80QT Interface Pod Final Assembly

Table 6-2. A11 Processor PCB Assembly
(See Figure 6-2.)

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, 2		CAP,CER,10PF,+-10%,100V,C0G,1206	806943	89536	806943	2		
C 3- 31		CAP,CER,0.1UF,+-10%,25V,X7R,1206	747287	89536	747287	29		
CR 1	*	DIODE,SI,BV=75V,IO=250MA,SOT23	830489	89536	830489	1	1	
CR 2	*	DIODE,SI,BV=70.0V,IO=50MA,DUAL,SOT23	742320	89536	742320	1		
J 1		HEADER,2 ROW,.100CTR,RT ANG,26 PIN	512590	00779	1-87230-3	1		
J 2		SOCKET, 40 PIN ZIF	585133	89536	585133	1		
MP 1		SPACER,SWAGED,RND,BRASS,4-40X0.415	602284	89536	602284	3		
P 1		HEADER,1 ROW,.150CTR,30 PIN	807073	89536	807073	2		1
R 1	*	RES,CERM,1.2M,+-5%,.125W,250PPM,1206	806240	89536	806240	1		
R 2, 3, 14-	*	RES,CERM,4.7K,+-5%,.125W,250PPM,1206	740522	89536	740522	9		
R 20	*		740522					
R 4	*	RES,CERM,39,+-5%,.125W,250PPM,1206	746255	89536	746255	1		
R 5	*	RES,CERM,0.05MAX,.125W,1206	810747	89536	810747	1		
R 6- 13	*	RES,CERM,2K,+-5%,.125W,250PPM,1206	746461	89536	746461	8		
R 21	*	RES,CERM,3K,+-5%,.125W,250PPM,1206	746511	89536	746511	1		
U 2	*	EPROM 2764-Z PROGRAMMED	859426	89536	859426	1	1	
U 3, 12	*	IC,LSTTL,DUAL JK F/F,-EDG TRIG,SOIC	741256	18324	N74LS112DT	2	1	
U 4, 6	*	IC,LSTTL,HEX INVERTER,SOIC	741017	18324	N74LS04DT	2	1	
U 5	*	IC,NMOS,8 BIT MPU,8MHZ	853882	89536	853882	1	1	
U 7, 16, 28	*	IC,LSTTL,QUAD 2 INPUT OR GATE,SOIC	740878	18324	N74LS32D	3	1	
U 9, 14, 15	*	IC,LSTTL,QUAD 2 INPUT AND GATE,SOIC	740860	18324	N74LS08DT	3	1	
U 10	*	IC,LSTTL,QUAD BUS,SOIC	740977	18324	N74LS125ADT	1	1	
U 11	*	IC,FTTL,QUAD 2 INPUT NAND GATE,SOIC	772897	18324	N74F00DT	1	1	
U 13	*	IC,CMOS,HEX INVERTER,UNBUFFERED,SOIC	806893	18324	74HCU04DT	1	1	
U 17, 18	*	IC,LSTTL,DUAL D F/F,+EDG TRG,SOIC	740985	18324	S74LS74ADT	2	1	
U 19	*	IC,2K X 8 STATIC RAM,120NSEC,SOIC	742783	S0482	CXK5816M-12L	1	1	
U 20	*	IC,LSTTL,OCTL LINE DRVR,SOIC	742122	18324	N74LS244DT	1	1	
U 21, 24	*	IC,ALSTTL,OCTAL BUFFER/LINE DRVR,SOIC	838110	89536	838110	2	1	

Table 6-2. A11 Processor PCB Assembly (cont.)
(See Figure 6-2.)

REFERENCE DESIGNATOR -A->-NUMERICS---->	S	DESCRIPTION	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	RS -Q	NOT -E-
U 22, 23	*	IC,ALSTTL,OCTAL D F/F,+EDG TRG,SOIC	799569	01295	SN74ALS374DWR	2	1	
U 25, 26	*	IC,LSTTL, DIVIDE BY 16 BIN CNTR,SOIC	740936	89536	740936	2	1	
U 27	*	PALI6LA PROGRAMMED	859442	89536	859442	1	1	
U 29	*	IC,LSTTL,QUAD 2 INPUT NAND GATE,SOIC	741033	18324	N74LS00DT	1	1	
U 30	*	IC,LSTTL,TRIPLE 3 INPUT NOR GATE,SOIC	740993	18324	N74LS27D	1	1	
U 31	*	IC,LSTTL,3-8 LINE DCDR W/ENABLE,SOIC	740969	18324	N74LS138DT	1	1	
XU 2		SOCKET,IC,28 PIN	448217	09922	DILB28P-108	1		
XU 5		SOCKET,IC,40 PIN	429282	09922	DILB40P-108	1		
XU 27		SOCKET,IC,20 PIN	454421	09922	DILB20P-108	1		
Y 1		CRYSTAL,8MHZ,+-0.5%,HC-18/U	485060	89536	485060	1		

An * in 'S' column indicates a static-sensitive part.

Note 1: P1 reference designator is two (2) parts. See Schematic Diagram for reference.

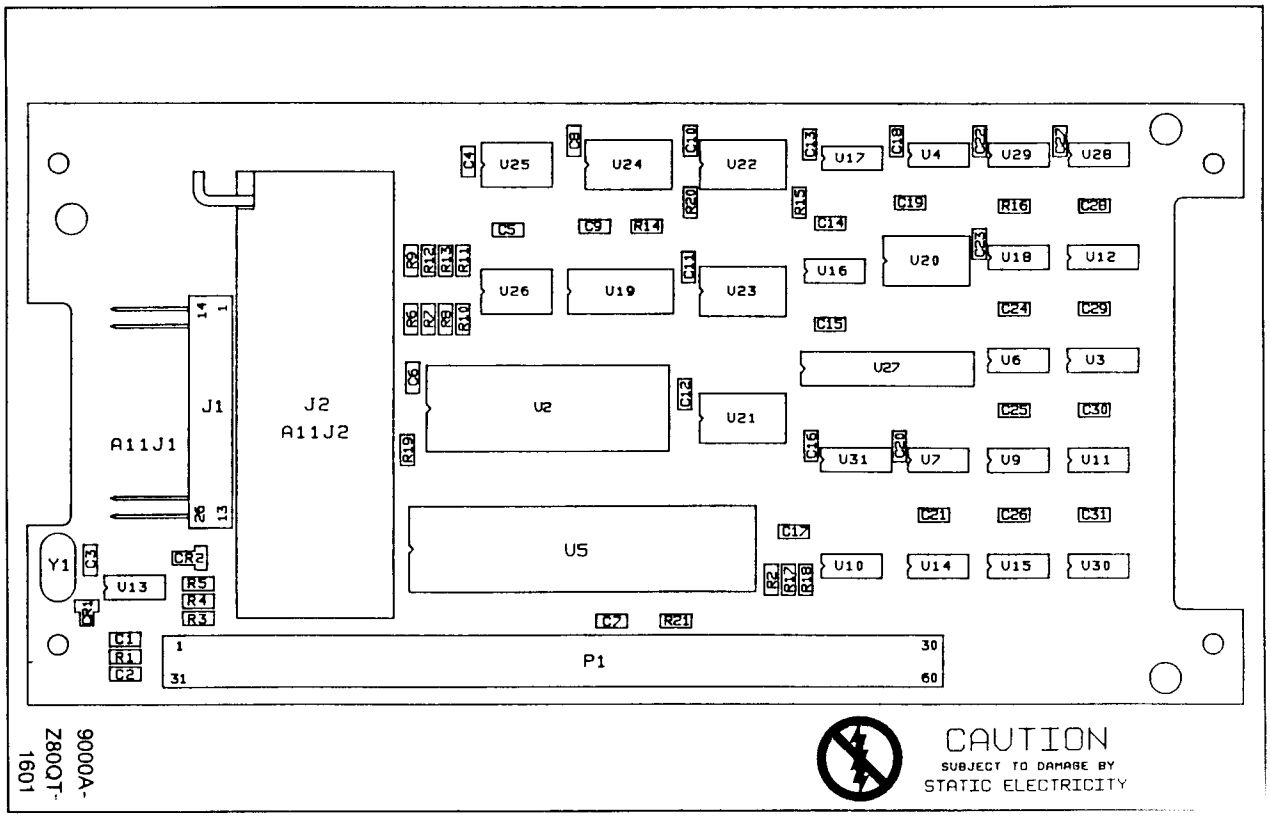


Figure 6-2. A11 Processor PCB Assembly

Table 6-3. A12 Interface PCB Assembly
(See Figure 6-3.)

REFERENCE DESIGNATOR -A>-NUMERICS---->	S	DESCRIPTION	FLUKE STOCK --NO--	MFRS SPLY -CODE-	MANUFACTURERS PART NUMBER -OR GENERIC TYPE-----	TOT QTY-	R S -Q	N O T -E-
A 1- 4		9000A HYBRID PROTECTION CIRCUIT	583021	89536	583021	4		
A 5		HYBRID ASSY, 700> TESTED	582270	89536	582270	1		
C 1- 4, 6-		CAP,CER,0.22UF,+-20%,50V,Z5U	519157	51406	RPE121-127-Z5U224M50V	10		
C 8, 11, 12,			519157					
C 14			519157					
C 9		CAP,TA,1UF,+-10%,35V	161919	56289	196D105X0035HA1	1		
C 10		CAP,CER,0.01UF,+-20%,100V,X7R	407361	04222	3419-1000-103M	1		
C 13, 15		CAP,TA,10UF,+-20%,15V	193623	56289	195D106X0015A1	2		
C 16		CAP,CER,1000PF,+-5%,50V,COG	528539	05397	C320C102J5G5EA	1		
CR 1- 5	*	DIODE,SI,BV=75V,IO=150MA,500MW	203323	15818	1N4448	5	1	
CR 6		DIODE,SI,20 PIV,1.0 AMP	507731	89536	507731	1		
J 1		SOCKET,2 ROW,PWB,0.150CTR,60 POS	602813	89536	602813	1		
J 2		HEADER,2 ROW,.100CTR,40 PIN	756833	89536	756833	1		
R 1		RES,CF,2.2K,+-5%,0.25W	343400	80031	1-4-5P2K2	1		
R 2, 3		RES,CF,1K,+-5%,0.25W	343426	80031	1-4-5P1K	2		
R 4		RES,CF,1.2K,+-5%,0.25W	441378	80031	1-4-5P1K2	1		
R 5		RES,CC,56,+-10%,0.5W	109009	89536	109009	1		
R 6, 7		RES,CF,200,+-5%,0.25W	441451	80031	1-4-5P200E	2		
R 8		RES,CF,820,+-5%,0.25W	442327	80031	1-4-5P820E	1		
R 9		RES,CF,0.51,+-5%,0.25W	381954	89536	381954	1		
U 1, 3, 5	*	IC,ALSTTL,OCTAL BUS XCVR W/3-STATE	647214	01295	SN74ALS245AN	3	1	
U 2, 4, 6,	*	IC,CMOS,OCTL D F/F W/3-STATE,+EDG TRG	585364	04713	MC74HCT374N	5	1	
U 8, 9	*		585364					
U 7	*	IC,LSTTL,3-8 LINE DCDR W/ENABLE	407585	01295	SN74LS138N	1	1	
U 10	*	PROTECTION I.C.	585992	89536	585992	1		
U 11	*	RES NET THICK FILM ASSY,TESTED-9000	583476	89536	583476	1	1	
U 12	*	IC,STTL,DUAL 4 INPUT NAND LINE DRIVER	585414	01295	SN74S140N	1	1	
U 13	*	IC,LSTTL,HEX BUFFER W/NOR ENABLE	483800	01295	SN74LS367AN	1	1	
VR 1	*	IC,1.22V,50 PPM T.C.,VOLTAGE REF	812024	89536	812024	1	1	

Table 6-3. A12 Interface PCB Assembly (cont.)
(See Figure 6-3.)

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-
XU 1- 6, 8,		SOCKET,IC,20 PIN	454421	09922	DILB20P-108	8		
XU 9			454421					
XU 10		SOCKET,IC,18 PIN	418228	91506	218-AG39D	1		
XVR 1		SPACER,TRANSISTOR MOUNT,DAP	175125	07047	10172-DAP	1		

An * in 'S' column indicates a static-sensitive part.

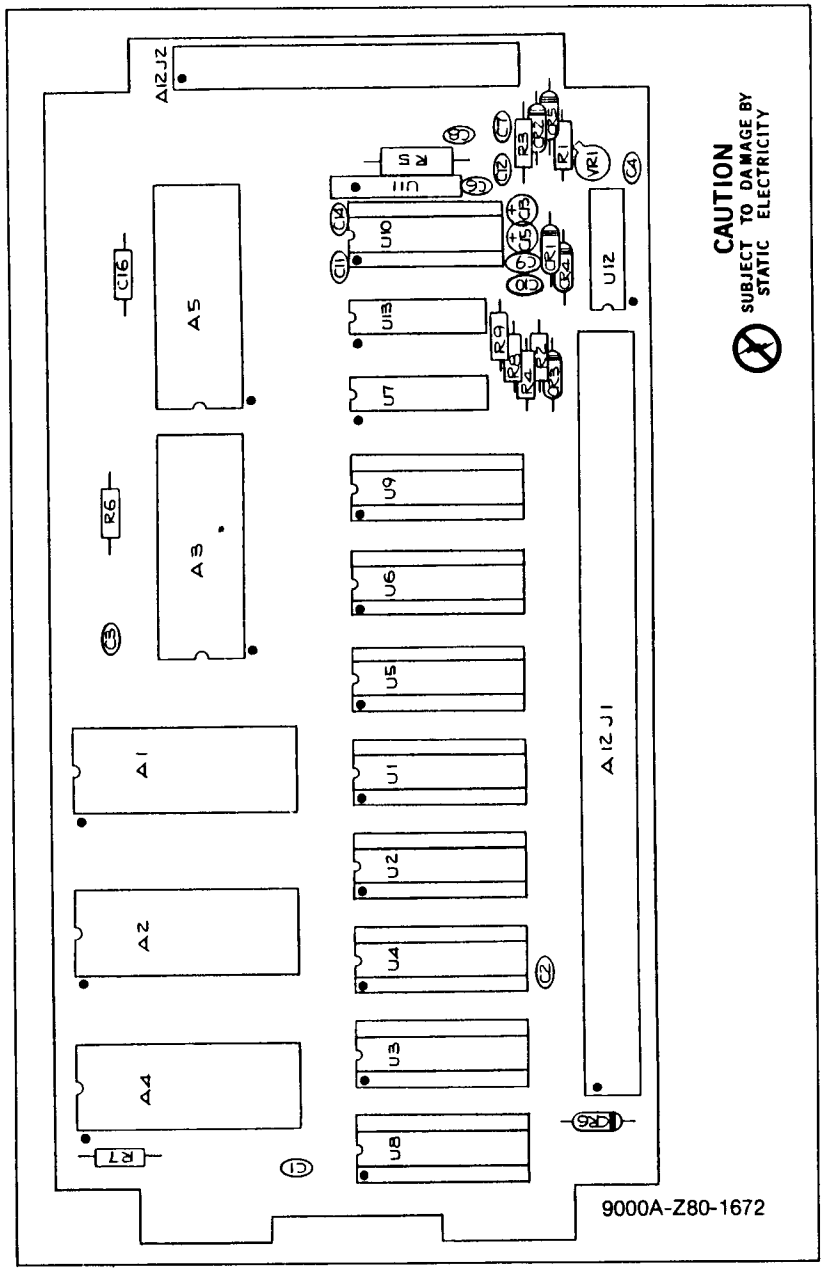


Figure 6-3. A12 Interface PCB Assembly

Section 7
Schematic Diagrams

TABLE OF CONTENTS

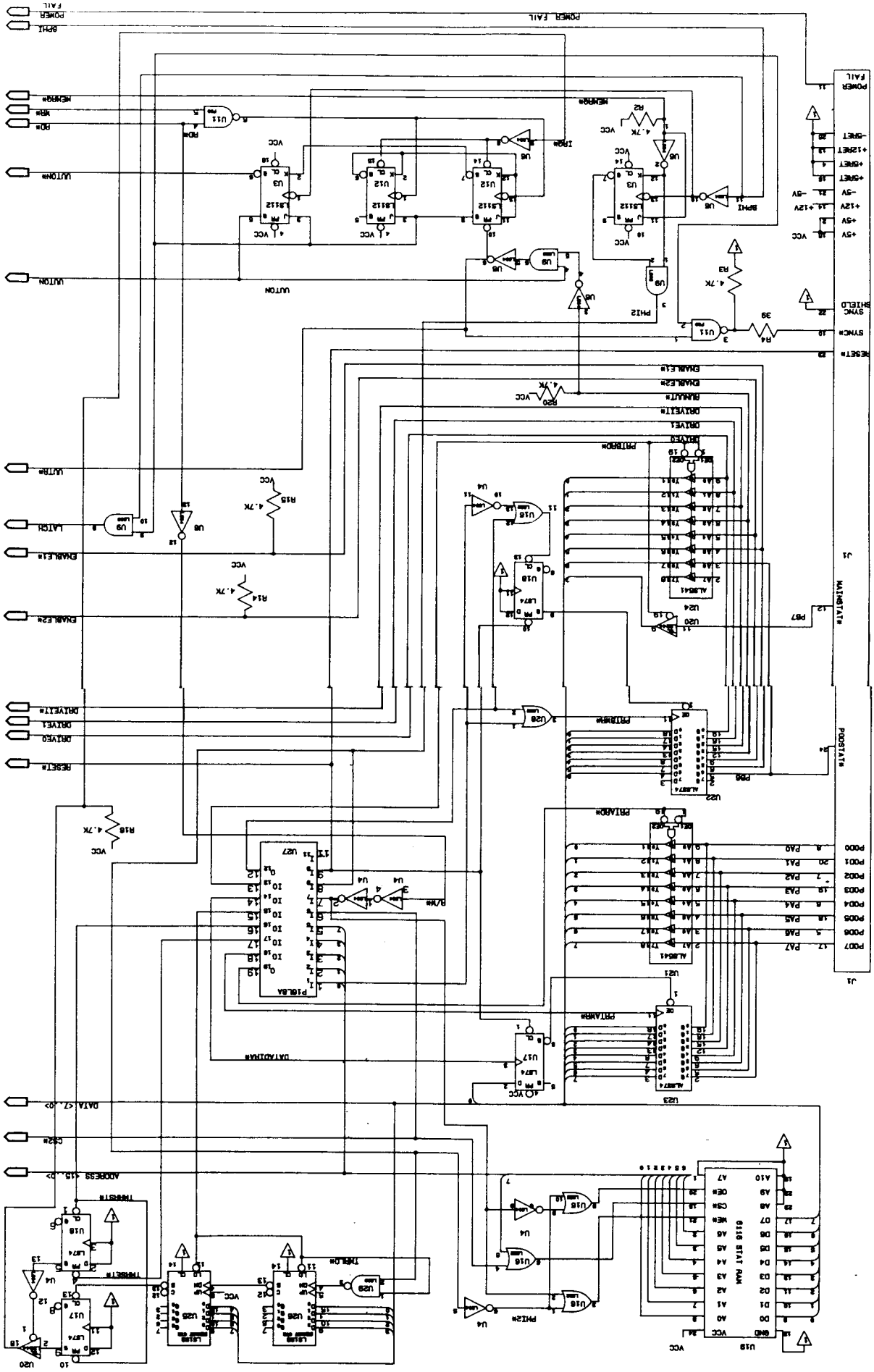


Figure 7-1. A11 Processor PCB Assembly

Z80QT

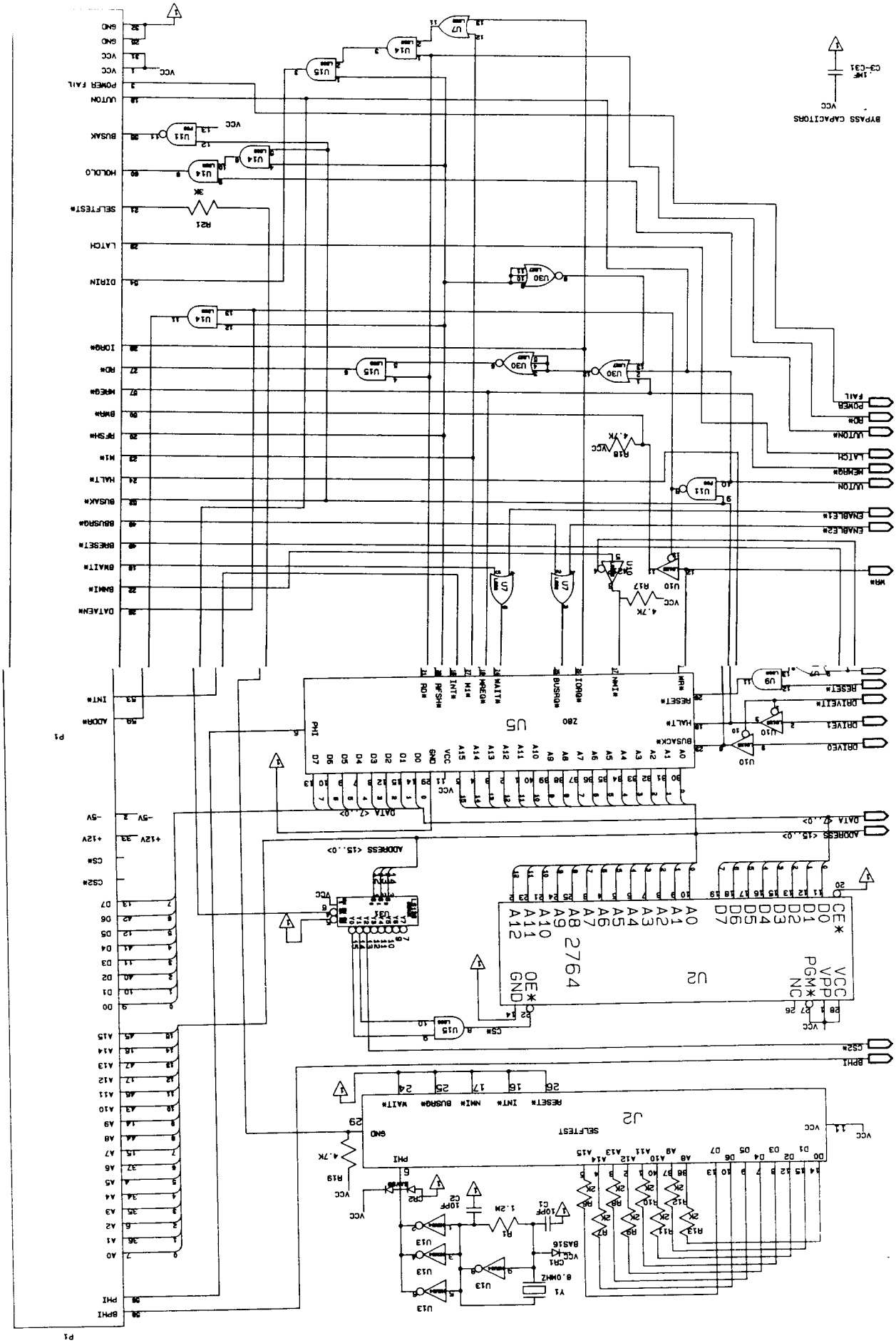
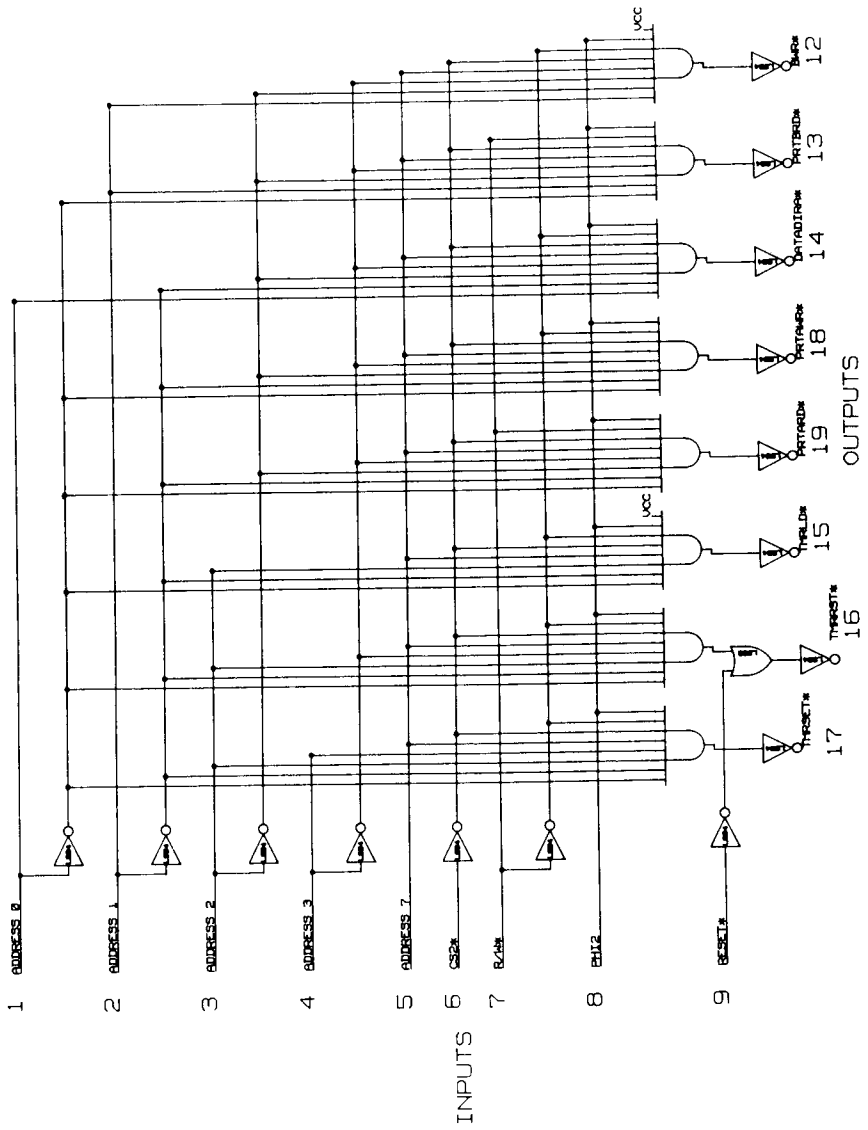


Figure 7-1. A11 Processor PCB Assembly (cont)



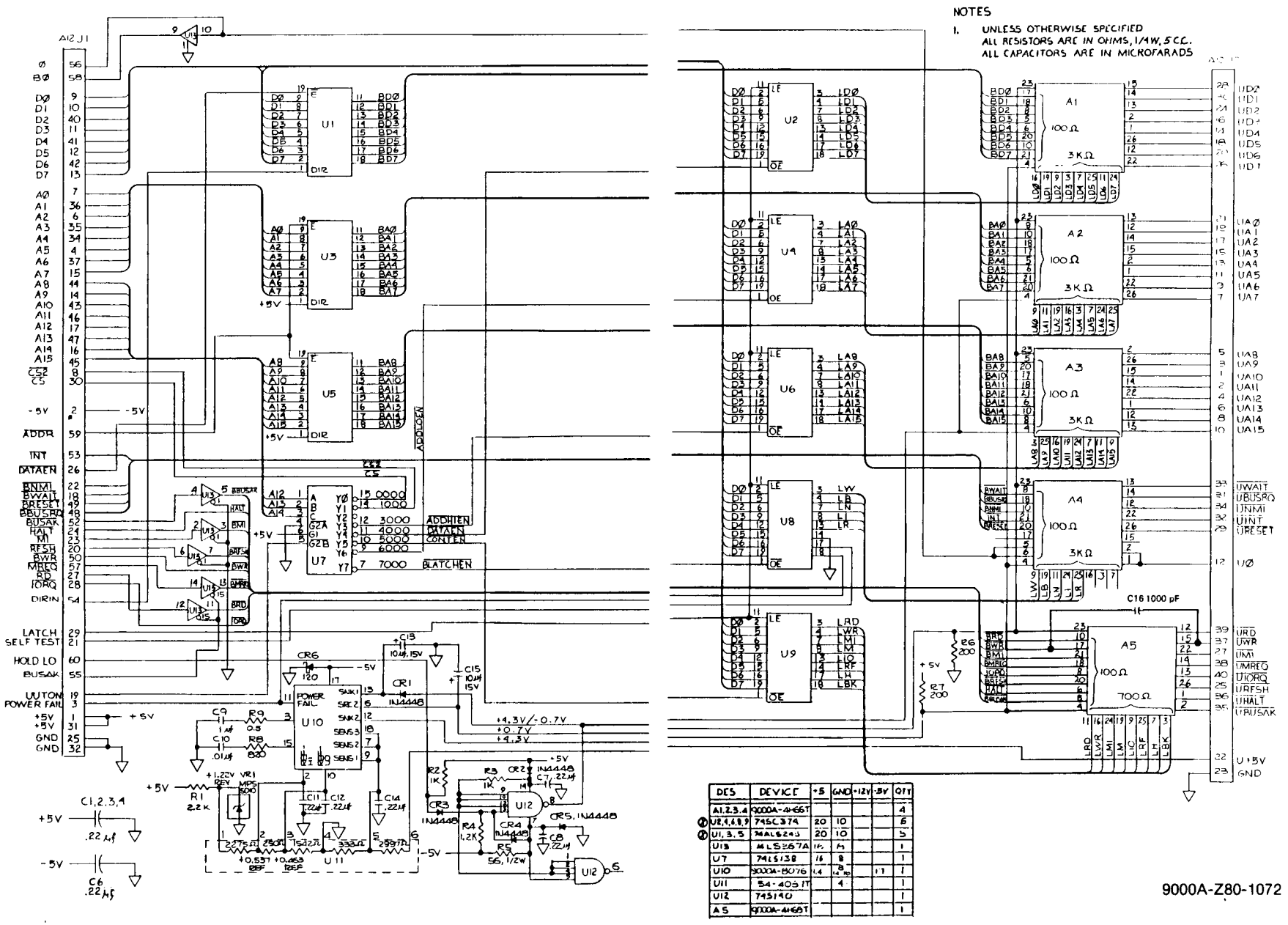
PAL (U27) LOGIC

Figure 7-1. A11 Processor PCB Assembly (cont)

REF DES	DEVICE	VCC	GND	PINS
U2	2764-2 EPROM	28	14	28
U3	74LS112	16	8	16
U4	74LS04	14	7	14
U5	Z80 8MHZ CPU	11	29	40
U6	74LS04	14	7	14
U7	74LS32	14	7	14
U9	74LS08	14	7	14
U10	74LS125	14	7	14
U11	74F00	14	7	14
U12	74LS112	16	8	16
U13	74HCU04	14	7	14
U14	74LS08	14	7	14
U15	74LS08	14	7	14
U16	74LS32	14	7	14
U17	74LS74	14	7	14
U18	74LS74	14	7	14
U19	6116 RAM	24	12	24
U20	74LS244	20	10	20
U21	74ALS541	20	10	20
U22	74ALS374	20	10	20
U23	74ALS374	20	10	20
U24	74ALS541	20	10	20
U25	74LS193	16	8	16
U26	74LS193	16	8	16
U27	16L8A PAL	20	10	20
U28	74LS32	14	7	14
U29	74LS00	14	7	14
U30	74LS27	14	7	14
U31	74LS138	16	8	16

Figure 7-1. A11 Processor PCB Assembly (cont)

Z80QT



DES	DEVICE	+5	GND	+12V	-5V	QTY
	A1,2,3,4	9000A-A466T					4
②	U2,4,6,8,9	74SC374	20	10			5
②	U1,3,5	74ALS245	20	10			3
	U13	MLS367A	16	8			1
	U7	74LS138	16	8			1
	U10	9000A-B076	14	4		1	1
	U11	54-405 1T		4			1
	U12	745140					1
	A5	9000A-A468T					1

Figure 7-2. A12 Interface PCB Assembly

Appendix A

Using the Z80QT Pod from TL/1 Programs

A-1. READING THE DATABASE VERSION NUMBER

The Z80QT Pod database is contained in a disk file on the 9100-Series Mainframe. The disk file contains information that determines the Pod's interface to the rest of the system. To read the version number of the database from the front panel, press SETUP, then the right arrow key (→), SOFT KEYS, the POD NAME softkey, and the ENTER key.

A-2. READING THE POD SOFTWARE VERSION NUMBER

To determine the Pod software version, you perform a read at a special address. From the front panel, press the READ key, the SPECIAL softkey, and the right arrow key (→). Then enter address F0 0012 and press ENTER.

A-3. TL/1 PROGRAMMING APPLICATIONS

A-4. Pod Address Space Options

The following Pod address space options are available:

SPACE
MEMORY
I/O

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

```
program example 1

    s = getspace space "I/O"
    setspace space s

end example1
```

A-5. Pod-Specific Setup Information

Pod-specific setup information is listed in Table A-1, along with data values, defaults, and ranges.

The following program demonstrates syntax use in TL/1 commands for the podsetup statement.

NOTE

TL/1 hexadecimal data requires a " $ " prefix character.

```
program setup

!-------------------------------------------------------------------------------
!   This program is an example of how to use the podsetup function
!   with Pod-specific setup parameters.
!
!   IMPORTANT NOTES:
!
!        All character strings are case insensitive.
!
!        Standard (non Pod-specific) setups use syntax with single-quoted
!        arguments and double-quoted values, e.g.,
!
!             podsetup 'report intr' "off"
!
!        Pod-specific setups either use a single single-quoted argument or a
!        single-quoted argument followed by a non-quoted value, e.g.,
!
!             podsetup 'intr_ack on'
!             podsetup 'seg_reg ds' $F800
!-------------------------------------------------------------------------------

                                    ! Note, even though ~BUSRQ and ~WAIT
                                    ! are Pod specific, "enable" is not.
                                    ! so these use the "built-in" syntax.
   podsetup 'enable ~BUSRQ' "off"
   podsetup 'enable ~WAIT' "off"
                                    ! These examples are all Z80QT
                                    ! Pod-specific.
   podsetup 'I_REG' $FF
end setup
```

Table A-1. Z80QT Pod Setup Parameters

POD SETUP	RANGE/KEY	DEFAULT	NOTES
I_REG nnnn	0-FF	00	*

* The I_REG setup allows users to set the value of the I register in the Z80 microprocessor. This is only useful for UUTs that use the contents of the I register (interrupt vector register) when it is echoed on the upper eight bits of the address bus during an interrupt acknowledge cycle. The special pod address for the I register is 20000h and it may take any value from 0 through FF.

A-6. List of Pod Sync Modes

A list of the Pod Sync Modes and the mnemonic for each is listed below:

NAME	MNEMONIC
Address Sync	ADDR
Data Sync	DATA

A-7. Pod Sync Calibration Data

Calibration is the process by which the internal delay lines in the I/O Module and Probe are adjusted to correctly align (in time) the clock and signals to be sampled. To calibrate an I/O Module or Probe to a Pod for a particular Pod sync mode, you are prompted to probe a signal on the UUT. The specified reference edge on that signal is found by adjusting the delay lines in the I/O Module or probe relative to the internal $\overline{\text{PODSYNC}}$ signal. The appropriate delay, labeled tcal (as shown in Figure A-1) may vary from one sync mode to another. If calibration is not performed, a default setting is used for the tcal value. When calibration is performed, the measured value for tcal replaces the default value.

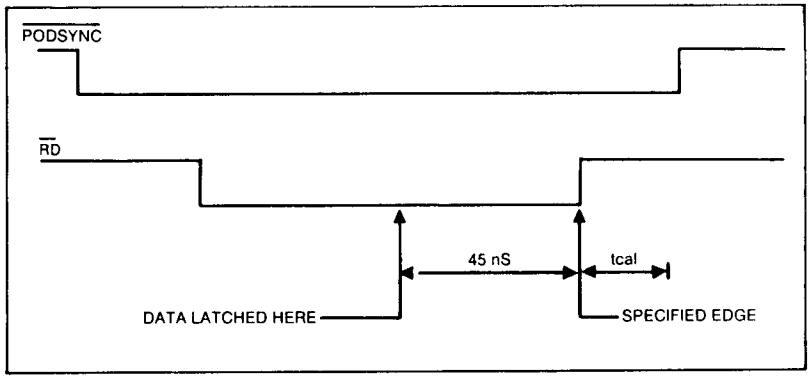


Figure A-1. 9000A-Z80QT Pod Address Sync Timing

Once the reference edge is found, an offset is applied to that edge to determine where in time the I/O Module or Probe latches data.

Figure A-1 contains an example of calibration for address sync that shows how the offset data listed in Table A-2 applies to real waveforms. The reference edge for address sync is the rising edge of $\overline{\text{RD}}$. The offset data shows that a valid address is best captured when sampled 45 ns before the rising edge of $\overline{\text{RD}}$. (A positive offset would indicate that the address should be latched after the reference edge.)

As a result of the calibration process, the offset value is set to that specified for the sync mode in use. If other offsets are required, the TL/1 "setoffset" statement can be used. See the "setoffset" statement and the "getoffset" statement in the 9100-Series TL/1 Reference Manual for exact syntax and details.

Table A-2. Z80QT Pod Sync Calibration Data

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	$\overline{RD}$	Rising	-45 ns
DATA	$\overline{RD}$	Rising	-45 ns

A-8. Available TL/1 Support Programs

The following list describes the available TL/1 support programs for the Z80QT Pod.

QWK_FILL

This program embodies all of the functions of the Z80QT pod Quick Fill and Verify test. For further information about the usage of the Quick Fill and Verify test, see Section 3-16. The major difference between the test that Section 3-16 describes and the test that is implemented by the QWK_FILL TL/1 program is that the TL/1 program only allows the Quick Fill and Verify test to be executed in the mainframe's current address space.

Arguments:

ADDR

This is the starting address of the Quick Fill test. The allowed range of values for ADDR is 0 through FFFF.

UPTO

This is the ending address of the Quick Fill test. The allowed range of values for UPTO is 0 through FFFF. UPTO's value must be greater than or equal to that of ADDR.

DATA

This is the data that is to be verified or written by the Quick Fill test. DATA may be any numeric value between 0 and FF.

ADDRSTEP

This is the address increment for the Quick Fill test. If ADDRSTEP has the value zero, the Quick Fill test will default to a one byte address increment. ADDRSTEP must not be greater than F.

FUNCTION

This determines whether the Quick Fill program performs a Quick Fill, a Quick Verify, or a Quick Fill and Verify. The values 1, 2, and 3 specify the test type, respectively. All other values are illegal.

Faults:

test_aborted:

reason "Illegal address in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick FILL program and should never be seen in normal operation.

reason "Illegal address increment". The value of the ADDRSTEP argument does not conform to the restrictions detailed above.

reason "Illegal data". The value of the DATA argument does not conform to the restrictions detailed above.

reason "Illegal data in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick FILL program and should never be seen in normal operation.

reason "Illegal function type". The value of the FUNCTION argument does not conform to the restrictions detailed above.

reason "Illegal space". The current address space is not legal for this test.

reason "Illegal start address". The value of the ADDR argument does not conform to the restrictions detailed above.

reason "Illegal stop address". The value of the UPTO argument does not conform to the restrictions detailed above.

reason "Illegal data". The value of the DATA argument does not conform to the restrictions detailed above.

reason "New command entered". A new command was entered during the execution of the Quick FILL test. This fault should never be raised because the program is in complete control of the mainframe while it is executing.

reason "Space not found". The current address space that the mainframe is in is not known by the program. This fault should never be raised in normal operation.

test_failed:

reason "Data does not match". The data written does not match the data read back during the verify cycle.

Slots: uut_address - The address at which the fault occurred.
data_expected - The data that was expected.
data_read - The data that was read.

Returns: Nothing.

QWK_RAM

This program embodies all of the functions of the Z80QT pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 3-14. The major difference between the test that Section 3-14 describes and the test that is implemented by the QWK_RAM TL/1 program is that the TL/1 program only allows the Quick RAM test to be executed in the mainframe's current address space.

Arguments:

ADDR

This is the starting address of the Quick RAM test. The allowed range of values for ADDR is 0 through FFFF.

UPTO

This is the ending address of the Quick RAM test. The allowed range of values for UPTO is 0 through FFFF. UPTO's value must be greater than or equal to that of ADDR.

ADDRSTEP

This is the address increment for the Quick RAM test. If ADDRSTEP has the value zero, the Quick RAM test will default to a one byte address increment. ADDRSTEP must be no more than F.

FUNCTION

This determines whether the Quick RAM program performs a Quick RAM test or a Quick Pattern Verify. The values 1 and 2 are the test type, respectively. All other values are illegal.

Faults:

test_aborted:

reason "Illegal address in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick RAM program and should never be seen in normal operation.

reason "Illegal address increment". The value of the ADDRSTEP argument does not conform to the restrictions detailed above.

reason "Illegal data in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick RAM program and should never be seen in normal operation.

reason "Illegal function type". The value of the FUNCTION argument does not conform to the restrictions detailed above.

reason "Illegal space". The current address space is not legal for this test.

reason "Illegal start address". The value of the ADDR argument does not conform to the restrictions detailed above.

reason "Illegal stop address". The value of the UPTO argument does not conform to the restrictions detailed above.

reason "New command entered". A new command was entered during the execution of the Quick RAM test. This fault should never be raised because the program is in complete control of the mainframe while it is executing.

reason "Space not found". The current address space that the mainframe is in is not known by the program. This fault should never be raised in normal operation.

test_failed:

reason "Pattern Verify Error". The Quick Pattern Verify has found an incorrect data pattern.

Slots: uut_address - The address at which the fault occurred
data_expected - The data that was expected
data_read - The data that was read
bad_data_mask - The hex mask of the bad data bits

See the 9100-Series TL/1 Reference Manual for a list of the RAM Test Fault Conditions.

Returns: Nothing.

QWK_RD

This program places the pod in Quick Looping read mode at the passed address. For further information about Quick Looping read mode, see Section 3-12.

Arguments:

ADDR

The address at which to perform the Quick Looping read. The allowed range of values for ADDR is 0 through FFFF.

Faults:

test_aborted:

reason "Illegal address". The ADDR argument does not conform to the specification detailed above.

reason "Illegal space". The current address space is not legal for this test.

reason "Space not found". The address space that the mainframe is in is not known by the program. This fault should never be raised in normal operation.

Returns:

The value that is found at the address specified by the argument during the first UUT access of the Quick Looping read mode.

QWK_ROM

This program embodies all of the functions of the Z80QT pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 3-15. The major difference between the test that Section 3-15 describes and the test that is implemented by the QWK_ROM TL/1 program is that the TL/1 program only allows the Quick ROM test to be executed in the mainframe's current address space.

Arguments:

ADDR

This is the starting address of the Quick ROM test. The allowed range of values for ADDR is 0 through FFFF.

UPTO

This is the ending address of the Quick ROM test. The allowed range of values for UPTO is 0 through FFFF. UPTO must be greater than ADDR.

ADDRSTEP

This is the address increment for the Quick ROM test. If ADDRSTEP has the value zero, the Quick ROM test will default to a one byte address increment. ADDRSTEP must be less than or equal to F (15 decimal).

Faults:

test_aborted:

reason "Illegal address in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick ROM program and should never be seen in normal operation.

reason "Illegal data". The value of the DATA argument does not conform to the restrictions detailed above.

reason "Illegal address increment". The value of the ADDRSTEP argument does not conform to the restrictions detailed above.

reason "Illegal data in invocation". This fault is included for consistency with the way 9000 series testers handle the Quick ROM program and should never be seen in normal operation.

reason "Illegal space". The current address space is not legal for this test.

reason "Illegal start address". The value of the ADDR argument does not conform to the restrictions detailed above.

reason "Illegal stop address". The value of the UPTO argument does not conform to the restrictions detailed above.

reason "New command entered". A new command was entered during the execution of the Quick ROM test. This fault should never be raised because the program is in complete control of the mainframe while it is executing.

reason "Space not found". The current address space that the mainframe is in is not known by the program. This fault should never be raised in normal operation.

test_failed:

reason "Inactive bits detected". Some set of bits has been determined to be inactive.

Slots: bit_mask - A mask in which the set bits have been determined to be inactive.

Returns:

A checksum of the area of memory that was tested. This checksum is unrelated to the signature that is generated by the regular ROM test.

QWK_WR

This program may be used to put the pod in Quick Looping write mode using the data and address specified by the arguments. For further information on Quick Looping write mode, see Section 3-12.

Arguments:

ADDR

The address at which to perform the Quick Looping write. The allowed range of values for ADDR is 0 through FFFF.

DATA

The data that is to be written to the address specified by the ADDR argument. Legal values for this argument are any number between 0 and FF.

Faults:

test_aborted:

reason "Illegal address". The value of ADDR does not conform to the restrictions detailed above.

reason "Illegal data". The value of DATA does not conform to the restrictions detailed above.

reason "Illegal space". The current address space is not legal for this test.

reason "Space not found". The current address space that the mainframe is in is not known by the program. This fault should never be raised in normal operation.

Returns: Nothing.

A-9. TL/1 Fault Conditions

The following list shows the TL/1 fault conditions that can result from Z80QT pod operation. (Handlers for most fault conditions are based on one of the mask types listed under the heading Bit Definitions of Fault Masks further on in this Appendix.)

- Fault Conditions Using the Address Mask

  pod_addr_tied

- Fault Conditions Using the Data Mask

  pod_data_incorrect
  pod_data_tied

- Fault Conditions Using the Control Mask

  pod_ctl_tied

- Fault Conditions Using the Status Mask

  pod_forcing_active
  pod_interrupt_active
  pod_timeout_enabled_line

- Fault Conditions With No Fault Mask

  pod_timeout_bad
  pod_timeout_no_clk
  pod_timeout_recovered
  pod_timeout_setup
  pod_uut_power

A-10. Bit Definitions of Fault Masks

Tables A-3 through A-6 describe the mapping from specific Z80QT pod signals to bit positions in the 64-bit fault masks generated by the built-in functions when they invoke TL/1 fault handlers. In each format example, unmapped positions have a "0" value to maintain the full mask length of 64 positions. Positions labeled "X" correspond to mapped signals. Mapped signals have a "1" value to indicate an active, faulty, or otherwise significant condition. A value of "0" represents the absence of a significant condition for that signal.

Table A-3. Address Signal Mapping to Fault Masks

Address signal fault mask format:

"0000000000000000000000000000000000000000000000000XXXXXXXXXXXXXXX"

Bit 63 — Bit 15 — Bit 0

Address signal fault mask bit assignments:

Mask Bit	Pod Signal	Z80 Pin No.
0	A0	30
1	A1	31
2	A2	32
3	A3	33
4	A4	34
5	A5	35
6	A6	36
7	A7	37
8	A8	38
9	A9	39
10	A10	40
11	A11	1
12	A12	2
13	A13	3
14	A14	4
15	A15	5
16 : 63	unused	

Table A-4. Data Signal Mapping to Fault Masks

Data signal fault mask format:

"0000000000000000000000000000000000000000000000000000000XXXXXXXX"

Bit 63 — Bit 7 — Bit 0

Data signal fault mask bit assignments:

Mask Bit	Pod Signal	Z80 Pin No.
0	D0	14
1	D1	15
2	D2	12
3	D3	8
4	D4	7
5	D5	9
6	D6	10
7	D7	13
8 : 63	unused	

Table A-5. Control Signal Mapping to Fault Masks

Control signal fault mask format:

"0000000000000000000000000000000000000000000000000000000XXXXXXXX"

Bit 63 — Bit 7 — Bit 0

Control signal fault mask bit assignments:

Mask Bit	Pod Signal	Z80 Pin No.
0	$\overline{\text{BUSAK}}$	23
1	$\overline{\text{HALT}}$	18
2	$\overline{\text{RFSH}}$	28
3	$\overline{\text{IORQ}}$	20
4	$\overline{\text{RD}}$	21
5	$\overline{\text{WR}}$	22
6	$\overline{\text{M1}}$	27
7	$\overline{\text{MREQ}}$	19
8 : 63	unused	

Table A-6. Status Signal Mapping to Fault Masks

Status signal fault mask format:

"00000000000000000000000000000000000000000000000XXXXXXXXXXXXXXX"

Bit 63 — Bit 7 — Bit 0

Status signal fault mask bit assignments:

Mask Bit	Pod Signal	Z80 Pin No.
0	WAIT	24
1	BUSRQ	25
2	NMI	17
3	INT	16
4	RESET	26
5	unused	
6	unused	
7	PWR FAIL	
8		
:	unused	
63		

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

```
program faults
!-------------------------------------------------------------------------------
!
! This program shows examples of raising each of the built-in primitive faults
! requiring specific mask arguments associated with the 9000A-Z80QT pod.
!
! When executed, this program stops as each fault is reported.  Press "CONT"
! to display successive faults.
!
!-------------------------------------------------------------------------------
    ! Raise built-in address drivability fault on A11.
    !
    fault pod_addr_tied mask "00000000000000000000000000000000000000000000000000
0000100000000000"
    !
    ! Raise built-in data drivability fault on D7.
    !
    fault pod_data_tied mask "00000000000000000000000000000000000000000000000000
0000000010000000"
    !
    ! Raise built-in control drivability fault on ~HALT.
    !
    fault pod_ctl_tied mask "00000000000000000000000000000000000000000000000000
0000000000000010"
    !
    ! Raise built-in active forcing line fault on ~WAIT
    !
    fault pod_forcing_active mask "00000000000000000000000000000000000000000000
0000000000000000001"
    !
    ! Raise built-in active interrupt fault on ~NMI
    !
    fault pod_interrupt_active mask "00000000000000000000000000000000000000000000
0000000000000000100"
    end faults
```