Supplemental Pod Information for 9100A/9105A Users

P/N 822866
July 1987 Rev 2 3/88

FLUKE®

MANUAL UPDATE PACKAGE

ISSUE NO: 1
DATE: 2/89

This Manual Update Package contains information necessary to ensure the accuracy of the following manual:

MANUAL

Title: Supplemental Pod Information for 9100A/9105A Users
Print Date: July 1987
Rev.- Date: 2, 3/88

PAGE EFFECTIVITY

Page No.	Print Date
1	2/89

---------- 1802 POD ----------

POD CONNECTION TO 9100 SERIES MAINFRAMES

Connect the pod cable with the 25-pin connector to the pod port on the side of the mainframe. The cable with the 9-pin connector is not used with 9100 Series products. Secure the connector using the slide locking collar.

All 9100 Series probe stimulus/response functions can be used for the UUTs with Vcc of 5V (the probe thresholds must be set to CMOS).

NOTE

The 9100 Series probe may be used as a measurement device for UUTs with Vcc greater than 5V if UUT logic low is below 0.8V. Again, the probe thresholds must be set to CMOS. If the probe is used in this manner, be aware that the thresholds of the probe do not match thresholds of typical CMOS devices when Vcc is above 5V. The output functions of the probe cannot be used for UUTs with Vcc greater than 5V.

TABLE OF CONTENTS

INTRODUCTION

Pods Supported

This manual provides information to support use of the Interface Pods listed below with 9100 series testers. The manual is a supplement to the Instruction Manual for each Pod.

1802	6502	6800	6802	6809	68000
8080	8085	8048	8051	8086	8088
80186	80188	9900	Z80	Z8000	

Multiple Databases per Pod

Many Fluke microprocessor interface pods support more than one microprocessor type. These pods fall into two types. The first type enters a mode depending upon the type of microprocessor installed in the pod and/or the configuration of switches on the pod. The second type of pod is general enough in its operation that it can emulate several processor types without knowing the specific type on the UUT.

The first type of pod reports to the mainframe which mode it is in and the mainframe automatically loads a database appropriate to that mode.

The mainframe also automatically loads a database for the second type of pod, but this may not be the database that enables the pod to most closely emulate the processor type. A keystroke method is available to choose a specific database. To do this, press the SETUP MENU key on the front panel of the 9100 series tester. Then select the POD NAME softkey. You can enter an alphanumeric value for the name of the database to be loaded and used with the pod that is installed. Further information about valid database names for each pod may be found in the sections for pods that support this feature.

OPERATOR'S INFORMATION

Reading the Database Version Number

Pod databases are disk files in the 9100 series Mainframe that contain information to determine the pod's interface to the rest of the system. To read the version number of the database from the front panel, press the SETUP, ->, and SOFT KEYS keys, the POD_NAME softkey, then the ENTER key. After a short disk operation, the front panel will display PODFILE = xxxxxx.# where xxxxxx is the pod database name and # is the revision level of that database.

Understanding 9100 Pod Self Test Return Codes

9100 series mainframe pod test algorithms are improved over those of 9000 series mainframes. Hence, the error codes returned by the 9100 series mainframes have different meanings. Table 1 details the steps that the 9100 takes in testing pods and their corresponding error codes.

NOTES FOR TABLE 1 (on next page):

A = UUT power sensing circuit fault.
B = Control line(s) cannot be driven.
C = Address line(s) cannot be driven.
D = Wrong data read.
E = Data line(s) cannot be driven.
F = Forcing or interrupt line buffer(s) or associated logic faulty.

maxdata = FF for 8-bit interface pods.
FFFF for 16-bit interface pods.

N = the number of enableable lines in the pod.

S = the number of address spaces in the pod.

Table 1. 9100 Pod Selftest Error Codes

ERROR CODE (DECIMAL)	ATTEMPTED ACTION	POSSIBLE FAULTS WHEN ACTION FAILS
0000	Readspecial addr 0FF00FF0. Reset and initialize pod.	A, B, C, D
0001	Writespecial addr 0FF00FF0, data F00F0FF0 AND maxdata	A, B, C, E
0002	Test control lines.	B
0003	Disable all enableable lines.	F
0004	Sync on ADDR.	A, F
0005	Readspecial addr 0000F00F.	A, B, C, D
0006	Writespecial addr 00000001, data FFFFFFFE AND maxdata	A, B, C, E
0007	Writespecial addr 00000002, data FFFFFFFD AND maxdata	A, B, C, E
0008	Writespecial addr 00000004, data FFFFFFFB AND maxdata	A, B, C, E
0009	Writespecial addr 00000008, data FFFFFFF7 AND maxdata	A, B, C, E
0010	Writespecial addr 00000010, data FFFFFFEF AND maxdata	A, B, C, E
0011	Writespecial addr 00000020, data FFFFFFDF AND maxdata	A, B, C, E
0012	Writespecial addr 00000040, data FFFFFFBF AND maxdata	A, B, C, E
0013	Writespecial addr 00000080, data FFFFFF7F AND maxdata	A, B, C, E
|	|	|
|	|	|
V	V	V
0034	Writespecial addr 10000000, data EFFFFFFF AND maxdata	A, B, C, E
0035	Writespecial addr 20000000, data DFFFFFFF AND maxdata	A, B, C, E
0036	Writespecial addr 40000000, data BFFFFFFF AND maxdata	A, B, C, E
0037	Writespecial addr 80000000, data 7FFFFFFF AND maxdata	A, B, C, E
0038	Enable first pod enableable line.	F
0039	Enable second pod enableable line.	F
|	|	|
|	|	|
V	V	V
0037+N	Enable last pod enableable line (N defined in NOTES).	F
0038+N	Write maxdata to low address of first address space. (N defined in NOTES on previous page)	A, B, C, E
0038+N +1	Write maxdata to high address of first address space. (N defined in NOTES on previous page)	A, B, C, E
|	|	|
|	|	|
V	V	V
0038+N+2S	Write maxdata to high address of last address space. (N and S defined in NOTES on previous page)	A, B, C, E
2001	Pod is not in selftest socket.	
2002	Unexpected powerfail.	
2003	No powerfail when expected.	
2004	Pod not reporting ABORT when ABORT line toggled.	
2005	Pod reporting ABORT unexpectedly.	
2006	Pod drops dead unexpectedly.	
2007	Pod gives error setting enableable lines.	
2008	Pod gives error setting fault mask.	

POD CALIBRATION

Calibration is the process by which the internal delay lines in the I/O module and Probe are adjusted to correctly line up, in time, the clock and data signals. (Data, in this context, refers to the signal to be sampled, be it address, data, etc.) To calibrate an I/O Module or Probe to a pod, for a particular pod sync mode, the user is prompted to probe a signal on the UUT. A particular edge on that signal is found by adjusting the delay lines in the I/O module or Probe relative to the internal PODSYNC signal. Once the edge is found, an offset is applied to that edge to determine just where in time the I/O module or Probe will latch data.

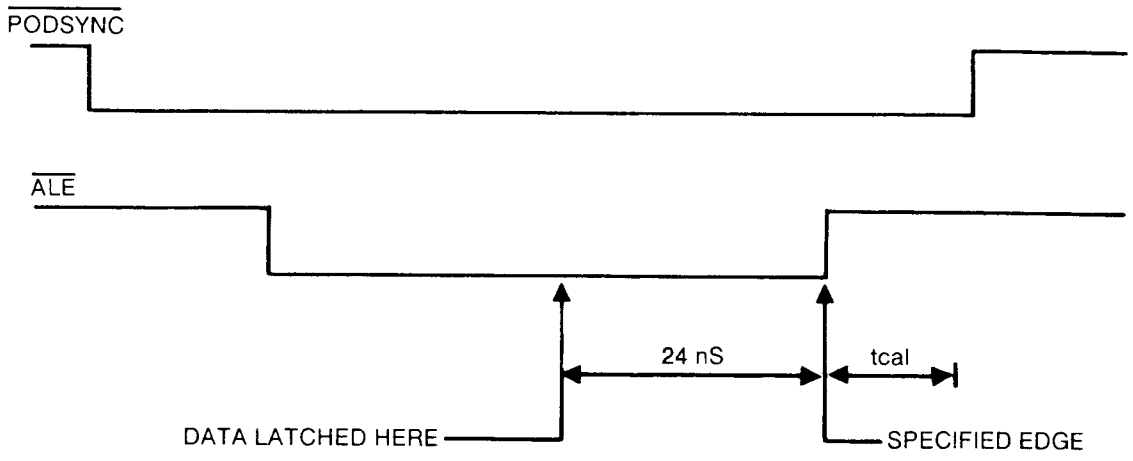
This manual lists the calibration data for each pod and sync mode. Following is an example for the imaginary xyz pod showing how the data is listed in the manual, and how this data would apply to real waveforms.

EXAMPLE:

In the imaginary xyz pod, Address sync is to be calibrated to the rising edge of the ~ALE line. For this example, valid address is best captured when sampled 24ns before the rising edge of ~ALE. The data would appear in the manual like this:

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	~ALE	RISING	-24ns

The waveforms would look like this:



NOTE: In this manual, any active-low signal name starts with '~'.

If calibration is not performed, a default setting is used for tcal, (the location of the ~ALE signal relative to ~PODSYNC). The -24ns offset is applied to this setting. When calibration is performed, tcal is actually measured, and the default is replaced with the measured value. If the TL/1 function "getoffset" were executed after calibration, an offset of -24ns should be returned, (or a number near -24, reflecting the granularity of the hardware delay lines). If other offsets are desired, the TL/1 function "setoffset" can be used. See the description of setoffset and getoffset in the TL/1 documentation for exact syntax and details.

PROGRAMMER'S INFORMATION

Fault Masks

Faults raised during testing may be processed by fault handlers written in TL/1, or displayed on the front panel of the 9100 series mainframe. System fault names and arguments associated with those faults are listed in Appendix G of the TL/1 Reference Manual. Many of the faults listed there have an argument named mask, mask_tied, mask_low, or mask_high. The format of these masks is a character string of exactly 64 ones (1) and zeros (0). "X" can be used for "don't care" bits. The rightmost position in the character string represents the least significant bit (LSB) position of the data item with the fault. The leftmost position of the string represents bit 63 of the data item with the fault.

The format of TL/1 fault masks follows directly from the model of a microprocessor used by the 9100 system. Processor signal lines are divided into address, data, status, and control groups. The format of each mask type is as follows:

Address, Data:	Bit positions map starting from MSB to LSB of mask. Bits beyond the processor bus width are unused.
Status, Control:	Bit positions map directly from the pod underside decal or from the Status and Control Line Bit Assignment table in the pod manual. Other bits are unused.

Exceptions to this general fault mask structure are covered in the sections for the 8048, 8051, 80186, and 80188 pods.

Because the mainframe generates different fault masks (Data, Address, Control, or Status) depending on the type of fault, the meaning associated with a particular position changes from type to type. For instance, in the 68000 pod, bit 1 can mean either ~BR fault (for status faults), ~VMA fault (for control faults), data bit 1 fault (for data faults), or address bit 1 fault (for address faults), depending on what type of fault occurred.

Table 2 lists the TL/1 fault masks for the 68000 pod.

Table 2. Bit Positions for 68000 Pod Fault Masks

BIT POSITION	ADDRESS MASK	DATA MASK	STATUS MASK	CONTROL MASK
0	A0	D0	~HALT	~HALT
1	A1	D1	~BR	~VMA
2	A2	D2	~BGACK	~BG
3	A3	D3	~BERR	unused
4	A4	D4	~RESET	~RESET
5	A5	D5	~VPA	FC0
6	A6	D6	~DTACK	FC1
7	A7	D7	PWR FAIL	FC2
8	A8	D8	IPL0	~LDS
9	A9	D9	IPL1	~UDS
10	A10	D10	IPL2	R/~W
11	A11	D11	intr vector	~AS
12	A12	D12	note 1	E
13	A13	D13	note 2	unused
14	A14	D14	unused	.
15	A15	D15	.	.
16	A16	unused	.	.
17	A17	.	.	.
18	A18	.	.	.
19	A19	.	.	.
20	A20	.	.	.
21	A21	.	.	.
22	A22	.	.	.
23	A23	unused	unused	unused
24				
:	unused	unused	unused	unused
63				

NOTES:
1. Neither ~DTACK nor ~VPA asserted.
2. Both ~DTACK and ~VPA asserted.

When testing within a fault handler, often only a particular fault is of interest. To do this, the fault mask can be compared against a string of ones, zeros, and Xs. As an example, suppose that only bit 3 is of interest. The comparison might be done something like this:

```
testmask = "XXXXX ... XXX1XXX"
if (mask = testmask) then ...
```

where the one is in the fourth position from the right (bit 3). The Xs cause any other fault to be temporarily ignored in the comparison. Other individual faults can be tested by changing the testmask string.

Examples of Fault Masks

The following examples show how fault masks are used to handle specific types of faults.

EXAMPLE : Forcing Line Fault Handlers

A pod_forcing_active fault raised during the read operation in program EXAMPLE1 below would cause the handler for pod_forcing_active to activate. The argument mask passed to the fault handler contains the string "00000000000000000000000000000000000000000000000000000100". This string means that the fault raised is due to a problem with bit 2 of the status word for this pod (since the status word reports forcing line faults). EXAMPLE1 tests this particular bit position, and if the fault is present, the fault handler disables further reporting of the pod_forcing_active fault by using the podsetup statement to turn off forcing line fault reports. (See the heading "Pod-Specific Setup Information" further on in this section for more information on using the podsetup function with pod-specific setup information.)

```
program EXAMPLE1

  handle pod_forcing_active ( mask )

    declare string mask

    ! Turn off reporting of forcing line faults if one is reported.

    if (mid str mask, from 62, length 1) = "1" then
      podsetup 'report forcing' "off"
    end if
  end pod_forcing_active

  read addr 0

end EXAMPLE1
```

EXAMPLE : Enableable Line Fault Handling

Each section in this manual contains information about the bit positions of the pod's enableable lines in the enableable line mask. Handling enableable line faults is exactly like handling other faults that use fault masks to report the bit position of the faulty line. For example, in the 6800 pod, there are three enableable lines; TSC in bit position 3, DBE in bit position 4, and ~HALT in bit position 5 of the enableable line mask. The mask for a pod_timeout_enabled_line fault for this pod would be of the format:

"00000000000000000000000000000000000000000000000000000000XYZ000"

where the X would contain a 1 if TSC caused a pod timeout and 0 otherwise. Y would contain a 1 if DBE caused a pod timeout, etc. A fault handler could test for which enableable line caused the timeout (as was done in EXAMPLE1).

Raising Built-in Faults from TL/1

TL/1 programs may raise system built-in faults. Information about system faults can be found in appendices G and H of the TL/1 Reference Manual. Raising built-in faults from a TL/1 program often requires that the fault mask format be passed as an argument to the fault statement. Care must be taken to ensure the mask is **exactly 64 characters long**. Program EXAMPLE2 shows how to raise a bus_data_low_tied fault on data line D4. Assuming no handlers for this fault exist in the calling chain for EXAMPLE2, the fault would report on the 9100 series front panel just as if the front panel BUSTEST found D4 tied low. Other built-in faults may be raised in the same manner.

```
program EXAMPLE2

declare string fift6_0s

  fift6_0s = "0000000000000000000000000000000000000000000000000000000000000000"
    ! The program somehow decides that D4 is tied low.  A fault is
    ! raised to indicate that fault.
    !
    ! Raise the fault:

  fault bus_data_low_tied mask fift6_0s + "00010000"

    ! If no handlers for this fault are in the calling chain for
    ! EXAMPLE2, the 9100 series front panel will display:
    !    "data line D4 pin xx stuck low"
    ! The pin number xx in the fault message is pod-dependent.

end EXAMPLE2
```

Pod-Specific Setup Information

Some pods have setup information that is only used for that pod. The TL/1 manual describes the podsetup command and gives examples of the podsetup command that can be used with any pod.

There are differences between the way the podsetup command works with the general setups and pod-specific setups. The following examples using the podsetup command are for the Z8000 pod. See the Z8000 section of this manual for a list of the setups that are available for that pod.

NOTE

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

```
program SETUP

!-------------------------------------------------------------------
!
!   This program is a example of how to use the podsetup function
!   with pod-specific setup arguments (example pod-specific setup
!   information is from the Z8000 pod database).
!
!   IMPORTANT NOTES:
!       All character strings are case insensitive.
!
!       Standard (not pod-specific) setups use syntax with single
!       quoted arguments and double quoted values, eg.:
!               podsetup 'report intr' "off"
!
!       Pod specific setups either use a single single-quoted argument
!       or a single-quoted argument followed by a non-quoted value:
!               podsetup 'intr_ack on'
!               podsetup 'seg_reg ds' $F800
!
!-------------------------------------------------------------------
                                ! Note, even though enableable status
                                ! lines ~BUSRQ, ~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 Z8000
                                ! pod-specific.
  podsetup 'trnsp_rd high_ad' $100F
  podsetup '~mo_sig state active'
  podsetup 'refresh rate' $15
  podsetup 'refresh status active'
                                ! Note upper case works too.
  podsetup 'TRNSP_RD HIGH_AD' $100F
  podsetup '~MO_SIG STATE ACTIVE'
  podsetup 'REFRESH RATE' $15
  podsetup 'REFRESH STATUS ACTIVE'

end SETUP
```

---------- 1802 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

SPACE
MEMORY
I/O

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
DMA_TRAP key	ON/OFF	ON	See Section 4-21 and Tbl 4-2
QLINE key	ON/OFF	OFF	See Table 4-2

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

CLR_MBIO This program clears Multi-Bank I/O mode operation. For further information, see Section 4-19 and Table 4-2 in the 1802 pod manual.

Arguments : None.

Faults : None.

Returns : Nothing.

DMA_IND This program returns the data from the most recent DMA IN operation, resets the DMA IN flag, and re-enables DMA operations. For more information, see Section 4-21 and Table 4-2 in the 1802 pod manual.

Arguments : None.

Faults : None.

Returns : The data from the most recent DMA IN operation that was trapped by the pod.

DMA_OUTD This program returns the data from the most recent DMA OUT operation, resets the DMA OUT flag, and re-enables DMA operations. For more information, see Section 4-21 and Table 4-2 in the 1802 pod manual.

Arguments : None.

Faults : None.

Returns : The data from the most recent DMA OUT operation that was trapped by the pod.

QWK_IO_R This program returns the data present at the passed I/O port number and places the pod in Quick Looping Read mode at that I/O port. For further information, see Section 4-15 through 4-20 and Table 4-2 of the 1802 pod manual.

Arguments : PORT - The port number at which the Quick Looping Read is to be performed. PORT may take values from 1 through 7.

Faults : test_aborted
reason "Illegal port number".
The value of the PORT argument does not conform to the restrictions detailed above.

Returns : The data found at the passed port during the first cycle of the Quick Looping Read.

QWK_IO_W This program repeatedly writes the passed data to the passed I/O port number. For further information, see Section 4-15 through Section 4-20 and Table 4-2 in the 1802 pod manual.

Arguments : PORT - The port number at which the Quick Looping Read is to be performed. PORT may take values from 1 through 7.

DATA - The data that is to be written to the above I/O port. Any number between 0 and FF is a legal value for this argument.

Faults : test_aborted
reason "Illegal port number".
The value of the PORT 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.

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 4-15 of the 1802 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Any number between 0 and FFFF is a legal value for this argument.

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

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

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 4-15 of the 1802 pod manual.

Arguments : ADDR - The address at which to perform the quick-looping write. Any number between 0 and FFFF is a legal value for this argument.

DATA - The data that is to be written to the address specified by the ADDR argument. Any number between 0 and FF is a legal value for this argument.

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.

Returns : Nothing.

RD_DMAR This program returns the current DMA register contents (16 bits). For further information, see Section 4-21 and Table 4-2 in the 1802 pod manual.

Arguments : None.

Faults : None.

Returns : The 16 bit-wide contents of the DMA register.

RD_TRV This program returns the current value of the I/O transfer vector (16 bits). For further information, see Section 4-16 through Section 4-20 and Table 4-2 in the 1802 pod manual.

Arguments : None.

Faults : None.

Returns : The 16 bit-wide value of the I/O transfer vector.

SET_DMAR This program sets the DMA register contents to the passed ADDR and writes the passed DATA to that address. For further information, see Section 4-21 and Table 4-2 in the 1802 pod manual.

Arguments : ADDR - This is the value to be loaded in the DMA register and the address to which the passed data will be written. This argument will take any value from 0 to FFFF.

DATA - This is the data to be written to the above address. Valid 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.

Returns : Nothing.

SET_MBIO This program sets up Multi-Bank I/O mode. After use of this program, every I/O operation will be preceded by a write of the passed DATA to the passed PORT. For further information, see Section 4-19 and Table 4-2 in the 1802 pod manual.

Arguments : PORT - This is the port that the bank selection data will be written to before each I/O operation. Any number between 1 and 7 is a legal value for this argument.

DATA - The data to be written to the above port before each I/O operation. Any number between 0 and FF is a legal value for this argument.

Faults : test_aborted

reason "Illegal port number".
The value of PORT does not conform to the restrictions detailed above.

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

Returns : Nothing.

SET_TRV This program sets the I/O transfer vector to the value of the passed argument. For further information, see Section 4-16 through 4-20 and Table 4-2 in the 1802 pod manual.

Arguments : VECTOR - The value that is to be the new I/O transfer vector. Any number between 0 and FFFF is a legal value for this argument.

Faults : test_aborted
reason "Illegal vector".
The value of VECTOR does not conform to the restrictions detailed above.

Returns : Nothing.

WR_DMA_V This program writes the passed data to the address the DMA register points to. For further information, see Section 4-21 and Table 4-2 in the 1802 pod manual.

Arguments : DATA - The value to be written to the address the DMA register points to. Any number between 0 and FF is a legal value for this argument.

Faults : test_aborted
reason "Illegal data".
The value of DATA does not conform to the restrictions placed above.

Returns : Nothing.

SYNC MODES

NAME	MNEMONIC	CODE
DMA Address Sync	DMA_ADDR	0
DMA Data Sync	DMA_DATA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	TPA	FALLING	-45ns
DATA	TPB	FALLING	-45ns
DMA_ADDR	TPA	FALLING	-45ns
DMA_DATA	TPB	FALLING	-45ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
~WAIT	bit 4	~WAIT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 6502 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

Only the following pod address space option is available.

SPACE
MEMORY

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

POD SYNC MODES

The 6502 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	CLK2	FALLING	-60ns
DATA	CLK2	FALLING	-10ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
READY	bit 5	RDY

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 6800 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

Only the following pod address space option is available.

SPACE
MEMORY

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

POD SYNC MODES

The 6800 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	PH2	FALLING	-25ns
DATA	PH2	FALLING	-25ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
THREE STATE CONTROL	bit 3	TSC
DATA BUS ENABLE	bit 4	DBE
~HALT	bit 5	~HALT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

6802 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS AVAILABLE

Only the following pod address space option is available.

SPACE
MEMORY

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

POD SYNC MODES

The 6802 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	E	FALLING	-30ns
DATA	E	FALLING	-30ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
MEMORY READY	bit 4	MR
~HALT	bit 5	~HALT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 6809 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

The 6809 pod has two modes, the 6809 mode and the 6809E mode. There is a database provided to support each of the two operational modes of this pod. The 9100 automatically detects which type of UUT is being used and loads the appropriate database for that UUT. In the following description, any differences between the databases are noted.

ADDRESS SPACE OPTIONS AVAILABLE

Only the following pod address space option is available.

SPACE
MEMORY

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

QWK_RD This program may be used to put the pod in Quick Looping read mode at the address specified by the argument. For further information about Quick Looping read mode, see Section 4-4 of the 6809 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Any number between 0 and FFFF is a legal value for this argument.

Faults : test_aborted
reason "Illegal address".
The ADDR argument has a value greater than FFFF.

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_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, refer to Section 4-4 of the 6809 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. Any number between 0 and FFFF is a legal value for this argument.

DATA - The data that is to be written to the address specified by the ADDR argument. Any number between 0 and FF is a legal value for this argument.

Faults : test_aborted
reason "Illegal address".
The ADDR argument has a value greater than FFFF.

reason "Illegal data".
The DATA argument has a value greater than FF.

Returns : Nothing.

POD SYNC MODES

The 6809 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

The 6809 and 6809E modes both have the same calibration data.

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	E	FALLING	-15ns
DATA	E	FALLING	-15ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

6809 Mode :

NAME	BIT POSITION	MNEMONIC
~HALT	bit 1	~HALT
~DMA	bit 4	~DMA
MEMORY READY	bit 5	MR

6809E Mode :

NAME	BIT POSITION	MNEMONIC
THREE STATE CONTROL	bit 0	TSC
~HALT	bit 1	~HALT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 68000 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

The 68000 pod operates with the DIP, PGA, PLCC, and Quad Pack versions of the microprocessor. In order to support the package variations, three databases are required to support the pod. Like other pods, the 68000 pod assumes a particular package type is being used. Because of this, the user must enter the name of the required database into the 9100 series mainframe if the database for a package type other other than the default is desired.

Use the following databases with the corresponding package types.

PACKAGE TYPE	DATABASE TO USE
DIP	68000 (default)
PGA	68000P
LCC/PLCC	68000L
Quad Pack	68000L

To load the database for the PGA, LCC/PLCC, or Quad Pack package styles, you must press the SETUP MENU key, then the -> key, then the SOFT KEYS key. Then the POD NAME softkey must be pressed, followed by the -> key. Now you may type the name of the desired database (68000, 68000L, or 68000P) and press the ENTER key. After a short disk operation the new database will be loaded.

The 68000 database comes up when the 68000 pod is plugged into the 9100 series mainframe because the pod reports the name 68000. If you wish to change the default startup database for the 68000 pod, you may copy the desired library over the 68000 library (be sure **NOT** to do this to the Distribution disks). Since the 9100 looks in the 68000 library when starting up the 68000 pod, whatever is present in that library is used as the startup database. Copying over the default removes the need to go through the steps in the previous paragraph every time the mainframe is reset.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

SPACE	SIZE
SUPPROG	WORD
SUPPROG	BYTE
USRPROG	WORD
USRPROG	BYTE
SUPDATA	WORD
SUPDATA	BYTE
USRDATA	WORD
USRDATA	BYTE

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
POD_CTRL DTACK_D nnn	0-3FF	A	See Section 4-32
ERR_MASK CTRL_DR nnnn	0-FFFF	FFFF	See Section 4-44
ERR_MASK FORCING nnnn	0-FFFF	FFF	See Section 4-45
IPL_MASK VALUE n	0-7	0	See Section 4-30
DFLT_FC STANDBY n	0-7	6	See Section 4-36
DFLT_FC STANDARD n	0-7	6	See Section 4-34
DFLT_FC DEFAULT n	0-7	6	See Section 4-38
DFLT_FC SB_AD nnnn	0-FFFF	FFFF	See Section 4-35
DFLT_FC DEF_ADDR nnnn	0-FFFF	FFFF	See Section 4-37
STCK_PTR SUP_LOW nnnn	0-FFFF	4000	See Section 4-46
STCK_PTR SUP_HIGH nnnn	0-FFFF	0000	See Section 4-47
STCK_PTR USR_LOW nnnn	0-FFFF	4000	See Section 4-48
STCK_PTR USR_HIGH nnnn	0-FFFF	0000	See Section 4-49

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

FRC_INT This program forces the execution of an interrupt acknowledge routine at the passed Interrupt Priority Level.

Arguments : IPL - The interrupt priority level of the interrupt acknowledge cycle that is to be simulated. IPL may have any value from 1 to 7.

Faults : test_aborted
reason "Illegal argument".
The value of IPL does not conform to the requirements detailed above.

Returns : Nothing.

FRC_RINT This program forces continuous execution of the an interrupt acknowledge routine at the passed Interrupt Priority Level.

Arguments : IPL - The interrupt priority level of the interrupt acknowledge cycle that is to be simulated. IPL may have any value from 1-7.

Faults : test_aborted
reason "Illegal argument".
The value of IPL does not conform to the requirements detailed above.

Returns : Nothing.

QWK_RAM This program embodies all of the functions of the 68000 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4-22 of the 68000 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick RAM test. UPTO may be any numeric value between 0 and FFFFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than 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 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 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
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 4-21 of the 68000 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Any number between 0 and FFFFFF is a legal value for this argument. ADDR's value must be even if the mainframe is in a WORD address space.

Faults : test_aborted

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

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 68000 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4-23 of the 68000 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick ROM test. UPTO may be any numeric value between 0 and FFFFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than 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 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 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:
bad_data_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 checksum 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 4-21 of the 68000 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. Any number between 0 and FFFFFF is a legal value for this argument. ADDR's value must be even if the mainframe is in a WORD address space.

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 if the mainframe is in a BYTE address space or any number between 0 and FFFF in a WORD address space.

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 "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.

RD_NOARM This program returns the most recent interrupt type information from the pod without re-enabling interrupts. For further information, refer to Section 4-28 of the 68000 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle. For help interpreting the return value, see Section 4-28 in the 68000 pod manual.

RD_REARM This program returns the most recent interrupt type information from the pod and re-enables interrupts. For further information, consult Section 4-29 of the 68000 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle. For help interpreting the return value, see Section 4-28 in the 68000 pod manual.

UUT_RST Execution of this program causes the pod to execute a RESET instruction, forcing the RESET line low for 124 clock cycles.

Arguments : None.

Faults : None.

Returns : Nothing.

POD SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	~AS	RISING	-40ns
DATA	~AS	RISING	-15ns
INTA	~AS	RISING	-15ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
~HALT	bit 0	~HALT
~BUS REQUEST/~BUS GRANT ACK	bit 1	~BR/~BG
INTR	bit 3	INTR

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

8080 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

SPACE
MEMORY
I/O

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

SYNC MODES

The 8080 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes names are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	DBIN	FALLING	-90ns
DATA	DBIN	FALLING	-90ns

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	DBIN	FALLING	-90ns
DATA	DBIN	FALLING	-90ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
READY	bit 4	READY
HOLD	bit 5	HOLD

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

8085 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

SPACE
MEMORY
I/O

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

POD SYNC MODES

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	-20ns
DATA	~RD	RISING	-40ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
HOLD	bit 1	HOLD

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

8048 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

The 8048 pod requires multiple databases to function properly with the many different microcomputers that the pod supports. The four pod-reported names described in the pod manual each correspond to a separate database/program set in 9100 series mainframes. The pod-reported names and their 9100 series equivalents are:

POD-REPORTED NAME	9100 SERIES DATABASE NAME
'35/48	3548
'39/49	3949
'40/50	4050
8041	8041

9100 series mainframes are able to use the pod-reported name to load the appropriate database. Important differences among the databases are noted in the Section below.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

3548, 3949, and 4050 Databases:

SPACE	RAMTYPE	PORTTYPE
PROGRAM		
DATA	INT_MPU	
DATA	EXT_UUT	
POD_RAM		
I/O		PORT
I/O		EXP_PORT

8041 Database:

SPACE	PORTTYPE
INT_RAM	
POD_RAM	
I/O	PORT
I/O	EXP_PORT

POD DATA SIGNAL DEFINITIONS FOR FAULT MAPPING

The following describes the non-standard mapping of specific 8048 pod data signals to fault masks for the purpose of invoking built-in 9100 fault handlers or raising built-in 9100 fault messages. In the format examples which follow bits labeled "0" are not used in the mapping but must be "0" regardless. Those labeled "X" correspond to mapped signals and are set to "0" to specify "no fault" or to "1" when a fault is to be reported.

Address Signal Mapping to Fault Masks (8041 Mode Only)

Address signal fault mask format:

```
"00000000000000000000000000000000000000000000000000XXXX00000000"
 |                                                    |  |       |
 +-bit 63                                             |  +-bit8  +-bit 0
                                                      +-bit 11
```

Address signal fault mask bit assignments:

MASK BIT	POD SIGNAL	8041 PIN NO.
0 : 7	unused	
8	P20	21
9	P21	22
10	P22	23
11	P23	24
12 : 63	unused	

Data Signal Mapping to Fault Masks (8048 and 8041 Modes)

Data signal fault mask format:

```
"00000000000000000000000000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXX"
 |                                            |                       |
 +-bit 63                                     +-bit 23                +-bit 0
```

Data signal fault mask bit assignments:

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.
0	DB0	12
1	DB1	13
2	DB2	14
3	DB3	15
4	DB4	16
5	DB5	17
6	DB6	18
7	DB7	19
8	P10	27
9	P11	28
10	P12	29
11	P13	30
12	P14	31
13	P15	32
14	P16	33
15	P17	34
16	P20	21
17	P21	22
18	P22	23
19	P23	24
20	P24	35
21	P25	36
22	P26	37
23	P27	38
24 : 63	unused	

POD-SPECIFIC SETUP INFORMATION

3548, 3949, and 4050 Databases:

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
TO_LINE key	INPUT/CLK_OUT	INPUT	See Section 4-17

8041 Database:

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

QWK_RD This program may be used to put the pod in Quick Looping read mode at the address specified by the argument. For further information about Quick Looping read mode, see the 8048 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are:

3548, 3949, 4050 Modes:
- PROGRAM Space - 0 through FFF.
- DATA EXT_UUT Space - 0 through FF.
- I/O PORT - 0 through 2.
- I/O EXP_PORT Space - 4 through 7.

8041 Mode:
- I/O PORT - 0 through 2.
- I/O EXP_PORT Space - 4 through 7.

Faults : test_aborted

reason "Illegal address".
The ADDR argument does not conform to the restrictions placed 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 : The value that is found at the address specified by the argument during the first UUT access of the Quick Looping read mode.

QWL_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, refer to the 8048 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are:

3548, 3949, 4050 Modes:
DATA EXT_UUT Space - 0 through FF.
I/O PORT - 0 through 2.
I/O EXP_PORT Space - 4 through 7.
8041 Mode:
I/O PORT - 0 through 2.
I/O EXP_PORT Space - 4 through 7.

DATA - The data that is to be written to the address specified by the ADDR argument. Any number between 0 and FF is a legal value for this argument (0 - F for I/O EXP_PORT space).

Faults : test_aborted

reason "Illegal address".
The ADDR argument does not conform to the restrictions placed above.

reason "Illegal data".
The DATA argument does not conform to the restrictions placed 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.

SYNC MODES

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

All 8048 pod modes have the same calibration data.

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	-45ns
DATA	~RD*	RISING	-45ns

* PROG signal is used in 8041 mode.

ENABLEABLE LINES

NONE

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

```
pod_data_incorrect
pod_data_tied
```

Fault Conditions Using the Address Mask

```
pod_addr_tied
```

Fault Conditions Using the Status Mask

```
pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line
```

Fault Conditions Using the Control Mask

```
pod_control_tied
```

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

8051 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

The 8051 pod requires multiple databases to function properly with the many different microcomputers that the pod supports. The three pod-reported names described in the pod manual each correspond to a separate database/program set in 9100 series mainframes. The pod-reported names and their 9100 series equivalents are:

POD-REPORTED NAME	9100 SERIES DATABASE NAME
8031	8031
8051	8051
8051X	8051X

9100 series mainframes are able to use the pod-reported name to load the appropriate database. Important differences among the databases are noted in the Section below.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

SPACE	BIT_TYPE	BIT_REG
EXT_PRG		
EXTDAT16		
EXT_DAT8		
INT_DAT		
SFR_REG		
BIT	FLAG	
BIT	REGISTER	P0
BIT	REGISTER	P1
BIT	REGISTER	P2
BIT	REGISTER	P3
BIT	REGISTER	TCON
BIT	REGISTER	SMOD
BIT	REGISTER	IE
BIT	REGISTER	IP
BIT	REGISTER	PSW

POD ADDRESS SPACE OPTIONS AVAILABLE (CONT)

SPACE	BIT_TYPE	BIT_REG
BIT	REGISTER	ACC
BIT	REGISTER	B
INT_PRG		

POD DATA SIGNAL DEFINITIONS FOR FAULT MAPPING

The following describes the non-standard mapping of specific 8051 pod data signals to fault masks for the purpose of invoking built-in 9100 fault handlers or raising built-in 9100 fault messages. In the format example which follows, bits labeled "0" are not used in the mapping but must be "0" regardless. Those labeled "X" correspond to mapped signals and are set to "0" to specify "no fault" or to "1" when a fault is to be reported.

DATA SIGNAL MAPPING TO FAULT MASKS

Data signal fault mask format:

```
"00000000000000000000000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
 |                      |                                        |
 +-bit 63               +-bit 39                                 +-bit 0
```

Data signal fault mask bit assignments:

MASK BIT	POD SIGNAL	8051 PIN NO.
0	AD0	39
1	AD1	38
2	AD2	37
3	AD3	36
4	AD4	35
5	AD5	34
6	AD6	33
7	AD7	32
8	P1.0	1
9	P1.1	2
10	P1.2	3
11	P1.3	4
12	P1.4	5
13	P1.5	6
14	P1.6	7

MASK BIT	POD SIGNAL	8051 PIN NO.
15	P1.7	8
16	P2.0	21
17	P2.1	22
18	P2.2	23
19	P2.3	24
20	P2.4	25
21	P2.5	26
22	P2.6	27
23	P2.7	28
24	P3.0	10
25	P3.1	11
26	P3.2	12
27	P3.3	13
28	P3.4	14
29	P3.5	15
30	P3.6	16
31	P3.7	17
32	P0.0	39
33	P0.1	38
34	P0.2	37
35	P0.3	36
36	P0.4	35
37	P0.5	34
38	P0.6	33
39	P0.7	32
40 : 63	unused	

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
REFRESH key	ON/OFF	OFF	See Section 4-38
ERRMASK CONTROL nn	0-FF	FF	See Section 4-27
ERRMASK FORCING nn	0-FF	FF	See Section 4-28
SW_MASK P3.2 key	SWITCH,OVERRIDE	SWITCH	See Section 4-39
SW_MASK P3.3 key	SWITCH,OVERRIDE	SWITCH	See Section 4-39
SW_DATA P3.2 key	I/O,INT0	I/O	See Section 4-39
SW_DATA P3.3 key	I/O,INT1	I/O	See Section 4-39

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

QWK_FILL This program embodies all of the functions of the 8051 pod Quick Fill and Verify test. For further information about the usage of the Quick Fill and Verify test, see Section 4-45 of the 8051 pod manual. The major difference between the test that the pod manual 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 address-space dependent. The following are allowed values:
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.

UPTO - This is the ending address of the Quick Fill test. The allowed range of values for UPTO is address-space dependent. The following are allowed values:
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.
UPTO's value must be greater than 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 RAM Fill, a Quick Pattern Verify, or both a Quick RAM Fill and a Quick Pattern 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 8051 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4-43 of the 8051 pod manual. The major difference between the test that the pod manual 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 address-space dependent. The following are allowed values:
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.

UPTO - This is the ending address of the Quick RAM test. The allowed range of values for UPTO is address-space dependent. The following are allowed values:
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.
UPTO's value must be greater than 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

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 4-17 of the 8051 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. The allowed range of values for ADDR is address-space dependent. The following are allowed values:
EXT_PRG Space - 0 through FFFF.
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.

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 8051 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4-44 of the 8051 pod manual. The major difference between the test that the pod manual 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 address-space dependent. The following are allowed values:
EXT_PRG Space - 0 through FFFF.
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.
INT_PRG Space - 0 through FFFF.

UPTO - This is the ending address of the Quick ROM test. The allowed range of values for UPTO is address-space dependent. The following are allowed values:

EXT_PRG Space - 0 through FFFF.
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.
INT_DAT Space - 0 through FF.
INT_PRG Space - 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:
bad_data_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 4-17 of the 8051 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. The allowed range of values for ADDR is address-space dependent. The following are allowed values:
EXTDAT16 Space - 0 through FFFF.
EXT_DAT8 Space - 0 through FF.

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.

RD_PORT This program returns the value on the specified input port. For further information, see Section 4 of the 8051 pod manual.

Arguments : PORT - The port number from which to read. Any number between 0 and 3 is a legal value for this argument.

Faults : test_aborted
reason "Illegal port".
PORT does not conform to the restrictions placed above.

Returns : The data found at the specified port.

TESTPORT This program performs a bus test on the output lines of the specified port. The bustest is not performed on the lines specified in the input mask.

Arguments : PORT - The port number on which the bus test is to be performed must be 0 to 3.

INPUT_MASK - Bits set to 1 will not be tested. Only bits 0 through 7 are significant.

Faults : test_aborted
reason "Illegal mask".
The value of the INPUT_MASK argument does not conform to the restrictions detailed above.

reason "Illegal port".
PORT does not conform to the requirements specified above.

pod_data_tied
The test raises the built-in pod_data_tied fault if a line is found not to be drivable.

Returns : Nothing.

WR_PORT This program writes the passed data to the specified port, masking out bits that are set in the input mask. For further information, see Section 4 of the 8051 pod manual.

Arguments : PORT - The port number from which to read must be 0 to 3.

DATA - The data to be written to the above port.

INPUT_MASK - Bits set to 1 will not be written regardless of whether they are supposed to be written as part of the data. Only bits 0 through 7 are significant.

Faults : test_aborted

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

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

reason "Illegal port".
PORT does not conform to the requirements specified above.

Returns : Nothing.

SYNC MODES

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

All 8051 pod modes have the same calibration data.

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	-45ns
DATA	~RD	RISING	-45ns

ENABLEABLE LINES

NONE

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 8086 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

There are two databases for this pod. They correspond to the two operating modes of the 8086 microprocessor, Minimum Mode and Maximum Mode. The pod reports either the name '8086' or the name '8086MX' depending on the state of the MN/~MX signal (processor pin 33). The mainframe automatically configures itself for the appropriate database. Differences between the two modes are noted in the following information.

Pod selftest uses the '8086' database. When changing between pod selftest and operation in a Maximum Mode UUT, a display message will indicate that the Mainframe needs to load the correct pod database. Press the RESET key to continue.

POD ADDRESS SPACE OPTIONS

The pod address space options are the same for both the 8086 database and the 8086MX database.

SPACE	SIZE
DATA	WORD
DATA	BYTE
CODE	WORD
CODE	BYTE
EXTRA	WORD
EXTRA	BYTE
STACK	WORD
STACK	BYTE
I/O	WORD
I/O	BYTE

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

FRC_INT This program forces the execution of an interrupt acknowledge routine. For further information, see Section 4-38 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : Nothing.

FRC_RINT This program forces continuous execution of the an interrupt acknowledge routine by executing a Quick Looping write. For further information about the use of this program, see Sections 4-25 and 4-38 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : Nothing.

QWK_ARM This program places the pod in Quick Looping read mode, continuously re-arming interrupts by clearing the Interrupt Vector Status bit. For further information about this program, see Sections 4-25 and 4-38 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type captured on the bus during the most recent interrupt acknowledge cycle, if any. Otherwise, the program returns a nonsense value.

QWK_RAM This program embodies all of the functions of the 8086 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4-26 of the 8086 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick RAM test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than 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 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 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
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 4-25 of the 8086 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value must be even if the mainframe is in a WORD address space.

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

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 8086 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4-27 of the 8086 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick ROM test. UPTO may be any numeric value between 0 and FFFFF unless then mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than 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 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 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:
bad_data_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 checksum 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 4-25 of the 8086 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value must be even if the mainframe is in a WORD address space.

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 if the mainframe is in a BYTE address space or any number between 0 and FFFF in a WORD address space.

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 "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.

RD_CSCD This program returns the cascade address stored in the pod from the most recent execution of the interrupt acknowledge routine. For further information, refer to Section 4-37 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : The cascade address stored in the pod from the most recent execution of the interrupt acknowledge routine.

RD_NOARM This program returns the most recent interrupt type information from the pod and neither re-enables interrupts nor clears the INT VECT status bit (status bit 11). For further information, refer to Section 4-37 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle.

RD_REARM This program returns the most recent interrupt type information from the pod and re-enables interrupts and clears the INT VECT status bit (status bit 11). For further information, consult Section 4-37 of the 8086 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle.

RUN_UUT This program allows easy access to the RUNUUT mode on the 8086 pod. It allows specification of the starting register contents and execution address. For further information, consult Sections 4-28, 4-29, 4-30, and 4-31 of the 8086 pod manual.

Arguments : OFFSET - The offset past the contents of the CS register (shifted left four bits) at which to start the RUNUUT operation. Legal values for this argument are any numeric value between 0 and FFFF (CS shifted left four bits plus the value of OFFSET must not exceed FFFFF). If this argument has an odd value, the RUNUUT operation will be started in CODE BYTE space. Otherwise, CODE WORD space is used.

CS - The value the CS register is to take on at the beginning of the RUNUUT operation. CS must be less than FFFF.

DS - The value the DS register is to take on at the beginning of the RUNUUT operation. DS must be less than FFFF.

ES - The value the ES register is to take on at the beginning of the RUNUUT operation. ES must be less than FFFF.

SS - The value the SS register is to take on at the beginning of the RUNUUT operation. SS must be less than FFFF.

Faults : test_aborted
reason "Illegal address".
The value of OFFSET is greater than FFFF or CS shifted left four bits plus the value of OFFSET is greater than FFFFF.

reason "Illegal register value".
The value of some register is greater than FFFF.

Returns : Nothing.

POD SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

8086 mode :

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	+13ns
DATA	~RD	RISING	-30ns
INTA	~RD	RISING	-30ns

8086MX mode :

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE*	FALLING	+8ns
DATA	~RD	RISING	-30ns
INTA	~RD	RISING	-30ns

* This signal is provided by the external 8288 bus controller. If your UUT doesn't use the 8288, Calibration should be done on the falling edge of of the signal used to latch the address.

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

8086 Mode :

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
HOLD	bit 1	HOLD
INTERRUPT	bit 3	INTR

8086MX Mode :

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
~RQO/~GTO OPERATION LATCH	bit 1	~RQ/~GTO
~RQ1/~GT1 OPERATION LATCH	bit 2	~RQ/~GT1
INTERRUPT	bit 3	INTR

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 8088 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

There are two databases for this pod. They correspond to the two operating modes of the 8088 microprocessor, Minimum Mode and Maximum Mode. The pod reports either the name '8088' or the name '8088MX' depending on the state of the MN/~MX signal (processor pin 33). The mainframe automatically configures itself with the appropriate database. Differences between the two modes are noted in the following information.

Pod selftest uses the '8088' database. When changing between pod selftest and operation in a Maximum Mode UUT, a display message will indicate that the Mainframe needs to load the correct pod database. Press the RESET key to continue.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

SPACE	SEGMENT
MEMORY	DATA
MEMORY	CODE
MEMORY	EXTRA
MEMORY	STACK
I/O	

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

FRC_INT This program forces the execution of an interrupt acknowledge routine. For further information, see Section 4-38 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : Nothing.

FRC_RINT This program forces continuous execution of the an interrupt acknowledge routine by doing a Quick Looping write. For further information about the use of this program, see Sections 4-25 and 4-38 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : Nothing.

QWK_ARM This program places the pod in Quick Looping read mode, continuously re-arming interrupts by clearing the Interrupt Vector Status bit. For further information about this program, see Sections 4-25 and 4-38 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type captured on the bus during the most recent interrupt acknowledge cycle, if any. Otherwise, the program returns a nonsense value.

QWK_RAM This program embodies all of the functions of the 8088 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4-26 of the 8088 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

UPTO - This is the ending address of the Quick RAM test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than 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 may not be greater than F(hex).

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 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

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 4-25 of the 8088 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

Faults : test_aborted

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

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 8088 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4-27 of the 8088 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

UPTO - This is the ending address of the Quick ROM test. UPTO may be any numeric value between 0 and FFFFF unless then mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of 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 not be greater than F(hex).

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 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 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:
bad_data_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 4-25 of the 8088 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

DATA - The data that is to be written to the address specified by the ADDR argument. Any number between 0 and FF is a legal value for this argument.

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 "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.

RD_CSCD This program returns the cascade address stored in the pod from the most recent execution of the interrupt acknowledge routine. For further information, refer to Section 4-37 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : The cascade address stored in the pod from the most recent execution of the interrupt acknowledge routine.

RD_NOARM This program returns the most recent interrupt type information from the pod and neither re-enables interrupts nor clears the INT VECT status bit (status bit 11). For further information, refer to Section 4-37 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle.

wRD_REARM- This program returns the most recent interrupt type information from the pod and re-enables interrupts and clears the INT VECT status bit (status bit 11). For further information, consult Section 4-37 of the 8088 pod manual.

Arguments : None.

Faults : None.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle.

RUN_UUT This program allows easy access to the RUNUUT mode on the 8088 pod. It allows specification of the starting register contents and execution address. For further information, consult Sections 4-28, 4-29, 4-30, and 4-31 of the 8088 pod manual.

Arguments : OFFSET - The offset past the contents of the CS register (shifted left four bits) at which to start the RUNUUT operation. Legal values for this argument are any numeric value between 0 and FFFF (CS shifted left four bits plus the value of OFFSET must not exceed FFFFF).

CS - The value the CS register is to take on at the beginning of the RUNUUT operation. CS must be less than FFFF.

DS - The value the DS register is to take on at the beginning of the RUNUUT operation. DS must be less than FFFF.

ES - The value the ES register is to take on at the beginning of the RUNUUT operation. ES must be less than FFFF.

SS - The value the SS register is to take on at the beginning of the RUNUUT operation. SS must be less than FFFF.

Faults : test_aborted
reason "Illegal address".
The value of OFFSET is greater than FFFF or CS shifted left four bits plus the value of OFFSET is greater than FFFFF.

reason "Illegal register value".
The value of some register is greater than FFFF.

Returns : Nothing.

POD SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

8088 mode :

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	+15ns
DATA	~RD	RISING	-30ns
INTA	~RD	RISING	-30ns

8088MX mode :

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE *	FALLING	+20ns
DATA	~RD	RISING	-30ns
INTA	~RD	RISING	-30ns

* This signal is provided by the external 8288 bus controller. If your UUT doesn't use the 8288, Calibration should be done on the falling edge of of the signal used to latch the address.

ENABLE/DISABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

8088 Mode :

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
HOLD	bit 1	HOLD
INTERRUPT	bit 3	INTR

8088MX Mode :

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
~RQ0/~GT0 OPERATION LATCH	bit 1	~RQ/~GT0
~RQ1/~GT1 OPERATION LATCH	bit 2	~RQ/~GT1
INTERRUPT	bit 3	INTR

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 9900 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

SPACE
MEMORY
I/O

POD-SPECIFIC SETUP INFORMATION

NONE

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

POD SYNC MODES

The 9900 pod always runs in a single sync mode that is sufficient for both data and address sync. However, two sync modes are provided to allow for independent calibration of the ~PODSYNC signal to the appropriate UUT signal.

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	DBIN	FALLING	-45ns
DATA	DBIN	FALLING	-45ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
READY	bit 0	READY
~HOLD	bit 1	~HOLD

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

Z80 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

SPACE
MEMORY
I/O

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
I_REG nnnn	0-FF	0800	None. See text below.

The I_REG setup allows users to set the value of the I register in the Z80 microprocessor. This is only useful for UUTs which use the contents of the I register as 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.

AVAILABLE TL/1 SUPPORT PROGRAMS

NONE

SYNC MODES

NAME	MNEMONIC	CODE
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	~RD	RISING	-45ns
DATA	~RD	RISING	-45ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
~BUS REQUEST	bit 4	~BUSRQ
~WAIT	bit 5	~WAIT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- Z8000 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

MULTIPLE DATABASES

The Z8000 pod operates with the Z8001, Z8002, Z8003, and Z8004 microprocessors. In order to support the package variations and differences in addressable space, two databases are required to support the pod. Like other pods, the Z8000 pod assumes a particular processor type is being used. Because of this, the user must enter the name of the required database into the 9100 series mainframe if a database other than the default is desired.

Use the following databases for the corresponding processor types.

PROCESSOR	DATABASE TO USE
Z8001	Z8001
Z8002	Z8002 (default)
Z8003	Z8001
Z8004	Z8002 (default)

The two databases for this pod are called Z8001 and Z8002. The Z8001 database provides support for the Z8001 and Z8003 processors. The Z8002 database is the default and supports the Z8002 and Z8004 processors. If you wish to use the 9100 series mainframe with the Z8001 database, you must press the SETUP MENU key, then the -> key, then the SOFT KEYS key. Then the POD NAME softkey must be pressed, followed by the -> key. Now you may type the name of the desired database (Z8001 or Z8002) and press the ENTER key. After a short disk operation the new database will be loaded.

The Z8002 database comes up when the Z8000 pod is plugged into the 9100 series mainframe because the pod reports the name Z8000. You may have noticed that there are three libraries for the Z8000 pod on Master Userdisk 2. The library named Z8000 is a direct copy of the Z8002 library. If you wish to change the default startup database for the Z8000 pod, you may copy the Z8001 library over the Z8000 library. (Be sure **NOT** to do this to the Distribution disks.) Since the 9100 looks in the Z8000 library when starting up the Z8000 pod, whatever is present in that library is used as the startup database. Copying over the default removes the need to go through the steps in the previous paragraph every time the mainframe is reset.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

MODE	SPACE	SIZE
SYSTEM	DATA	WORD
SYSTEM	DATA	BYTE
SYSTEM	STD_I/O	WORD
SYSTEM	STD_I/O	BYTE
SYSTEM	SPC_I/O	WORD
SYSTEM	SPC_I/O	BYTE
SYSTEM	STACK	WORD
SYSTEM	PRGM(C)	WORD
SYSTEM	PRGM(D)	WORD
NORMAL	DATA	WORD
NORMAL	DATA	BYTE
NORMAL	STACK	WORD
NORMAL	PRGM(C)	WORD
NORMAL	PRGM(D)	WORD

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
TRNSP_RD HIGH_AD nnnn	0-FFFF	0800	See Section 4-33
TRNSP_RD OFFSET nnnn	0-FFFF	0000	See Section 4-33
~MO key	ON/OFF	OFF	See Section 4-34
REFRESH FUNCTION key	ON/OFF	ON	See Section 4-35
REFRESH RATE nn	0-3F	F	See Section 4-36
FAST_MD key	ON/OFF	ON	See Section 4-37
INT_FLG key	ON/OFF	OFF	See Section 4-38
ERR_MASK CTL_DRV nnnn	0-FFFF	FFFF	See Section 4-40
ERR_MASK FORC_REP nnnn	0-FFFF	FFFF	See Section 4-40
FCW VALUE nnnn	0-FFFF	0	See Section 4-32

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

QWK_RAM This program embodies all of the functions of the Z8000 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4-24 of the Z8000 pod manual. The major difference between the test that the pod manual 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 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. ADDR must be even if the pod in is a WORD address space.

UPTO - This is the ending address of the Quick RAM test. The allowed range of values for UPTO is 0 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. UPTO must be even if the pod is in a WORD address space. UPTO's value must be greater than 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 two byte address increment. ADDRSTEP must be less than F and must be even if the pod is in a WORD address space.

Faults : test_aborted

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

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

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

reason "Space not 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

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 4-23 of the Z8000 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. The allowed range of values for ADDR is 0 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. ADDR must be even if the pod is in a WORD address space.

Faults : test_aborted

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

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 Z8000 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4-25 of the Z8000 pod manual. The major difference between the test that the pod manual 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 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. ADDR must be even if the pod is in a WORD address space.

UPTO - This is the ending address of the Quick ROM test. The allowed range of values for UPTO is 0 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. ADDR must be even if the pod is in a WORD address space. 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 two byte address increment. ADDRSTEP must be less than or equal to F (15 decimal).

Faults : test_aborted

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

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

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

reason "Space not 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:
bad_data_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 checksum 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 4-23 of the Z8000 pod manual.

Arguments : ADDR - The address at which to perform the quick-looping write. The allowed range of values for ADDR is 0 to FFFF for the Z8002 database and 0 through 7FFFFF for the Z8001 database. ADDR must be even if the pod is in a WORD address space.

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 in a BYTE address space and 0 through FFFF in a WORD address space.

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 "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.

RD_NMI This program returns the data that was present on the bus during the last Non-Maskable Interrupt.

Arguments : None.

Faults : None.

Returns : The data read from the bus during the last Non-Maskable Interrupt.

RD_NVI This program returns the data that was present on the bus during the last Non-Vectored Interrupt.

Arguments : None.

Faults : None.

Returns : The data read from the bus during the last Non-Vectored Interrupt.

RD_VI This program returns the data that was present on the bus during the last Vectored Interrupt.

Arguments : None.

Faults : None.

Returns : The data read from the bus during the last Vectored Interrupt.

SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	~AS	RISING	+10ns
DATA	~DS	RISING	-45ns
INTA	~DS	RISING	-45ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
~BUS REQUEST	bit 4	~BUSREQ
~WAIT	bit 5	~WAIT

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

---------- 80186 POD ----------

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS

The following pod address space options are available.

MODE	SPACE	SIZE
NORMAL	MEMORY	WORD
NORMAL	MEMORY	BYTE
NORMAL	I/O	WORD
NORMAL	I/O	BYTE
DMA	MEMORY	WORD
DMA	MEMORY	BYTE
DMA	I/O	WORD
DMA	I/O	BYTE

POD CONTROL SIGNALS

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	POD MANUAL REFERENCE
0	~LOCK	48	See Table 3-1
1	RESET	57	See Table 3-1
2	HLDA	51	See Table 3-1
3	ALE/QS0	61	See Table 3-1
4	~WR/QS1	63	See Table 3-1
5	~RD	62	See Table 3-1
6	~DEN	39	See Table 3-1
7	DT/~R	40	See Table 3-1
8	~S0	52	See Table 3-1
9	~S1	53	See Table 3-1
10	~S2	54	See Table 3-1
11	A19/S6	65	See Table 3-1
12	~BHE	64	See Table 3-1
13	TMR_OUT	(expanding)	See Section 3-14
14	INTA_ER	(expanding)	See Section 3-13
15	CHIP_SEL	(expanding)	See Section 3-15
16	TMR_OUT0	22	See Table 3-1
17	TMR_OUT1	23	See Table 3-1

POD CONTROL SIGNALS (CONT.)

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	POD MANUAL REFERENCE
18	~INTA0	42	See Table 3-1
19	~INTA1	41	See Table 3-1
20	~UCS	34	See Table 3-1
21	~LCS	33	See Table 3-1
22	~MCS0	38	See Table 3-1
23	~MCS1	37	See Table 3-1
24	~MCS2	36	See Table 3-1
25	~MCS3	35	See Table 3-1
26	~PCS0	25	See Table 3-1
27	~PCS1	27	See Table 3-1
28	~PCS2	28	See Table 3-1
29	~PCS3	29	See Table 3-1
30	~PCS4	30	See Table 3-1
31	~PCS5	31	See Table 3-1
32	~PCS6	32	See Table 3-1
33 : 63		unused	

POD STATUS SIGNALS

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	TYPE	POD MANUAL REFERENCE
0	ARDY	55	F,E	See Table 3-1
1	SRDY	49	F,E	See Table 3-1
2	HOLD	50	F,E	See Table 3-1
3	~RES	24	F	See Table 3-1
4	DRQ0	18	F	See Table 3-1
5	DRQ1	19	F	See Table 3-1
6	TMR_IN0	20		See Table 3-1
7	TMR_IN1	21		See Table 3-1
8	~TEST	47		See Table 3-1
9	NMI	46	I	See Table 3-1
10	~QSMD			See Section 3-10
11	ACT_INTR	(expanding)		See Section 3-9
12	INTR_V0			See Section 3-8
13	INTR_V1			See Section 3-8

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	TYPE	POD MANUAL REFERENCE
14	unused			
15	PWR_FAIL			See Section 3-7
16	INT0	45	I	See Table 3-1
17	INT1	44	I	See Table 3-1
18	INT2	42	I	See Table 3-1
19	INT3	41	I	See Table 3-1
20 : 63		unused		

TYPES: E - Enableable
F - Forcing
I - Interrupt

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
POD_CTRL STDBY_AD nnn	0-3FF	FF	See Section 4B-14
POD_CTRL RESET key	ON/OFF	OFF	See Section 4B-15
ERR_MSK ACT_FRC nn	0-3F	3F	See Section 4B-21
ERR_MSK ACT_INT nn	0-1F	1F	See Section 4B-22
ERR_MSK CTRL_DR nnnn	0-FFFF	FFFF	See Section 4B-20
ERR_MSK SEG_DR n	0-F	F	See Section 4B-23
ERR_MSK ADDR_DR nnnn	0-FFFF	FFFF	See Section 4B-24
ERR_MSK DATA_DR nnnn	0-FFFF	FFFF	See Section 4B-25
ERR_MSK INTA_TO n	0-F	F	See Section 4B-26
ERR_MSK CHIP_SEL nnnn	0-1FFF	1FFF	See Section 4B-27

All of the following setups are used only as initialization values for the Peripheral Control Block functions. Refer to Appendix H of the 80186 pod manual for more information.

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
CS_REG MPCS nnnn	0-FFFF	A0FB	See Section 4B-4 to 4B-9
CS_REG MMCS nnnn	0-FFFF	81FB	See Section 4B-4 to 4B-9
CS_REG PACS nnnn	0-FFFF	403B	See Section 4B-4 to 4B-9
CS_REG LMCS nnnn	0-FFFF	3FFB	See Section 4B-4 to 4B-9
CS_REG UMCS nnnn	0-FFFF	C03B	See Section 4B-4 to 4B-9
TIMER0 M/C_WD nnnn	0-FFFF	0	See Appendix H
TIMER0 MAX_CNTB nnnn	0-FFFF	0	See Appendix H
TIMER0 MAX_CNTA nnnn	0-FFFF	0	See Appendix H
TIMER0 COUNT nnnn	0-FFFF	0	See Appendix H
TIMER1 M/C_WD nnnn	0-FFFF	0	See Appendix H
TIMER1 MAX_CNTB nnnn	0-FFFF	0	See Appendix H
TIMER1 MAX_CNTA nnnn	0-FFFF	0	See Appendix H
TIMER1 COUNT nnnn	0-FFFF	0	See Appendix H
TIMER2 M/C_WD nnnn	0-FFFF	0	See Appendix H
TIMER2 MAX_CNTA nnnn	0-FFFF	0	See Appendix H
TIMER2 COUNT nnnn	0-FFFF	0	See Appendix H

All of the following setups are used only as as initialization values for the RUNUUT function. Refer to Appendix H of the 80186 pod manual for more information.

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
SEG_REG EXTRA nnnn	0-FFFF	0	See Appendix H
SEG_REG STACK nnnn	0-FFFF	0	See Appendix H
SEG_REG CODE nnnn	0-FFFF	FFFF	See Appendix H
SEG_REG DATA nnnn	0-FFFF	0	See Appendix H
DMA_CH0 CTRL_WD nnnn	0-FFFF	0	See Appendix H
DMA_CH0 TRNS_CNT nnnn	0-FFFF	0	See Appendix H
DMA_CH0 DP_UPPR n	0-F	0	See Appendix H
DMA_CH0 DP_LWR nnnn	0-FFFF	0	See Appendix H
DMA_CH0 SP_UPPR n	0-F	0	See Appendix H
DMA_CH0 SP_LWR nnnn	0-FFFF	0	See Appendix H
DMA_CH1 CTRL_WD nnnn	0-FFFF	0	See Appendix H
DMA_CH1 TRNS_CNT nnnn	0-FFFF	0	See Appendix H
DMA_CH1 DP_UPPR n	0-F	0	See Appendix H
DMA_CH1 DP_LWR nnnn	0-FFFF	0	See Appendix H
DMA_CH1 SP_UPPR n	0-F	0	See Appendix H
DMA_CH1 SP_LWR nnnn	0-FFFF	0	See Appendix H
INT_CNTR INT3 nnnn	0-FFFF	F	See Appendix H

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
INT_CNTR INT2 nnnn	0-FFFF	F	See Appendix H
INT_CNTR INT1 nnnn	0-FFFF	F	See Appendix H
INT_CNTR INT0 nnnn	0-FFFF	F	See Appendix H
INT_CNTR DMA1 nnnn	0-FFFF	F	See Appendix H
INT_CNTR DMA0 nnnn	0-FFFF	F	See Appendix H
INT_CNTR TMR_CTL nnnn	0-FFFF	F	See Appendix H
INT_CNTR INT_STAT nnnn	0-FFFF	8000	See Appendix H
INT_CNTR INT_REQ nnnn	0-FFFF	0	See Appendix H
INT_CNTR IN_SERV nnnn	0-FFFF	0	See Appendix H
INT_CNTR PRI_MSK nnnn	0-FFFF	7	See Appendix H
INT_CNTR MASK nnnn	0-FFFF	FD	See Appendix H
INT_CNTR EOI nnnn	0-FFFF	0	See Appendix H
INT_CNTR INT_VECT nnnn	0-FFFF	0	See Appendix H
REL_REG VALUE nnnn	0-FFFF	FF	See Appendix H

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

CONFIG This program provides for easy interrupt mode configuration. See Sections 4A-20 through 4A-37 of the 80186 pod manual for further information.

Arguments : RMX_MODE - Sets the pod in RMX mode if this argument is non-zero. If RMX_MODE is non-zero, all other arguments to this function MUST be zero.

PAIR0 - A non-zero value configures INT0 and INT2/~INTA0 as an interrupt acknowledge pair. Otherwise, both lines are configured as inputs. This argument must be zero if RMX_MODE is set.

LVL_T_0 - If PAIR0 is set then a non-zero value will configure interrupt acknowledge pair 0 as level-triggered interrupts. Otherwise, PAIR0 is configured as an edge-triggered interrupt pair. If PAIR0 is zero, LVL_T_0 must be zero. Also, if RMX_MODE is set then LVL_T_0 must be zero.

PAIR1 - A non-zero value configures INT1 and INT3/~INTA1 as an interrupt acknowledge pair. Otherwise, both lines are configured as inputs. This argument must be zero if RMX_MODE is set.

LVL_T_1 - If PAIR1 is set then a non-zero value will configure interrupt acknowledge pair 1 as level-triggered interrupts. Otherwise, PAIR1 is configured as an edge-triggered interrupt pair. If PAIR1 is zero, LVL_T_1 must be zero. Also, if RMX_MODE is set then LVL_T_1 must be zero.

PRIORITY - If both PAIR0 and PAIR1 are set then PRIORITY determines which interrupt pair has priority. If PRIORITY is zero then pair 0 has priority. If PRIORITY is one then pair 1 has priority. PRIORITY must be set to zero if RMX_MODE is on. PRIORITY must also be set to zero if either PAIR0 or PAIR1 is off.

Faults : test_aborted
reason "Incompatible argument values".
The arguments do not conform to the specifications detailed above.

Returns : Nothing.

FRC_INT This program forces the execution of an interrupt acknowledge routine on either channel 0 or channel 1. For further information, see Section 4A-30 in the 80186 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted
reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : Nothing.

GET_PCB This program sets up the pod for PCB recovery (see 80186 manual appendix I), places the pod in RUNUUT mode, waits for the passed number of seconds, then pulls the pod out of RUNUUT mode. The program then copies the recovered PCB information into the 9100 setup, where it may be saved by using 9100 front panel commands.

Arguments : SECONDS - This is the number of seconds that the UUT will be left in RUNUUT mode before it is interrupted for PCB recovery.

Faults : test_failed

reason "NMI routine failed".
The pod-resident PCB capture routine failed.

reason "Pod RAM corrupted".
The pod RAM that is used to store PCB information is corrupt.

reason "Couldn't find PCB".
The PCB wasn't found by the pod-resident recovery routine.

Returns : Nothing. (The recovered information is left in the) (pod-specific setups on the 9100.)

QWK_FILL This program embodies all of the functions of the 80186 pod Quick Fill and Verify test. For further information about the usage of the Quick Fill and Verify test, see Section 4A-5 of the 80186 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick Fill test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than F (15 decimal).

FUNCTION - This determines whether the Quick Fill program performs a Quick RAM Fill, a Quick Pattern Verify, or both a Quick RAM Fill and a Quick Pattern 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 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 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 FILL test. This fault should never be raised because the program is in complete control of the mainframe while it is executing.

reason "Pod timeout occurred".
The pod timed out while performing this test. This indicates a drastic UUT fault which should be corrected by using tests with more diagnostic capability.

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.
bad_data_mask - A mask in which the set bits were read differently than they were written.

Returns : Nothing.

QWK_RAM This program embodies all of the functions of the 80186 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4A-4 of the 80186 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick RAM test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than F (15 decimal).

FUNCTION - This determines whether the Quick RAM program performs a Quick RAM test, a Quick Pattern Verify, or both a Quick RAM test and a Quick Pattern 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 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 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 "Pod timeout occurred".
The pod timed out while performing this test. This indicates a drastic UUT fault which should be corrected by using tests with more diagnostic capability.

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

Returns : Nothing.

QWK_RAMP This program embodies all of the functions of the 80186 pod Quick Ramp test. For further information about the usage of the Quick Ramp test, see Section 4A-7 of the 80186 pod manual. The major difference between the test that the pod manual describes and the test that is implemented by the QWK_RAMP TL/1 program is that the TL/1 program only allows the Quick Ramp test to be executed in the mainframe's current address space.

Arguments : ADDR - The address at which to perform the Quick Ramp. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value must be even if the mainframe is in a WORD address space.

DATA - The data value at which the ramping operation is to start. Legal values for this argument are any number between 0 and FF if the mainframe is in a BYTE address space or any number between 0 and FFFF in a WORD address space.

Faults : test_aborted

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

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

reason "Pod timeout occurred".
The pod timed out while performing this test. This indicates a drastic UUT fault which should be corrected by using tests with more diagnostic capability.

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 "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.

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 4A-12 of the 80186 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value must be even if the mainframe is in a WORD address space.

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

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 80186 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4A-6 of the 80186 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value may not be odd if the mainframe is in a WORD address space.

UPTO - This is the ending address of the Quick ROM test. UPTO may be any numeric value between 0 and FFFFF unless then mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR and may not be odd if the mainframe is in a WORD address space.

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 two byte address increment. ADDRSTEP must be even if the mainframe is in a WORD address space and may not be greater than 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 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 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:
bad_data_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 checksum 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 4A-12 of the 80186 pod manual.

Arguments : ADDR - The address at which to perform the quick-looping write. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF. ADDR's value must be even if the mainframe is in a WORD address space.

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 if the mainframe is in a BYTE address space or any number between 0 and FFFF in a WORD address space.

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 "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.

FRC_INT This program forces the execution of an interrupt acknowledge routine on either channel 0 or channel 1. For further information, see Section 4A-30 in the 80186 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : Nothing.

RD_CSCD This program returns the passed channel's cascade address as stored in the pod during the most recent execution of the interrupt acknowledge routine for that channel. For further information, refer to Section 4A-29 of the 80186 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel 1 invalid in iRMX mode".
Interrupt Acknowledge pair 1 does not exist in iRMX mode.

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The cascade address stored in the pod for the passed channel number from the most recent execution of the interrupt acknowledge routine for that channel.

RD_NOARM This program returns the most recent interrupt type information from the pod for the passed channel and neither re-enables interrupts nor clears the INT VECT status bit (status bit 11) for that channel. For further information, refer to Section 4A-28 of the 80186 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted
reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle for the passed channel.

RD_REARM This program returns the most recent interrupt type information from the pod for the passed channel and re-enables interrupts and clears the INT VECT status bit (status bit 11) for that channel. For further information, refer to Section 4A-28 of the 80186 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted
reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle for the passed channel.

SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	0 ns
DATA	~DEN	RISING	-30 ns
INTA	~DEN	RISING	-50 ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
EXTRDY*	bit 0	EXTRDY*
HOLD	bit 1	HOLD

* The 80186 manual lists lines SRDY and ARDY as separately enableable lines. However, they are both enabled/disabled by bit 0 (EXTRDY) in the enableable line mask.

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power

80188 POD

NOTE

Read the Introduction section for important information needed to operate and program this pod with the 9100 series mainframes.

POD ADDRESS SPACE OPTIONS AVAILABLE

The following pod address space options are available.

MODE	SPACE
NORMAL	MEMORY
NORMAL	I/O
DMA	MEMORY
DMA	I/O

POD CONTROL SIGNALS

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	POD MANUAL REFERENCE
0	~LOCK	48	See Table 3-1
1	RESET	57	See Table 3-1
2	HLDA	51	See Table 3-1
3	ALE/QS0	61	See Table 3-1
4	~WR/QS1	63	See Table 3-1
5	~RD	62	See Table 3-1
6	~DEN	39	See Table 3-1
7	DT/~R	40	See Table 3-1
8	~S0	52	See Table 3-1
9	~S1	53	See Table 3-1
10	~S2	54	See Table 3-1
11	A19/S6	65	See Table 3-1
12	S7	64	See Table 3-1
13	TMR_OUT	(expanding)	See Section 3-14
14	INTA_ER	(expanding)	See Section 3-13
15	CHIP_SEL	(expanding)	See Section 3-15
16	TMR_OUT0	22	See Table 3-1
17	TMR_OUT1	23	See Table 3-1
18	~INTA0	42	See Table 3-1
19	~INTA1	41	See Table 3-1
20	~UCS	34	See Table 3-1
21	~LCS	33	See Table 3-1

POD CONTROL SIGNALS (CONT.)

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	POD MANUAL REFERENCE
22	~MCS0	38	See Table 3-1
23	~MCS1	37	See Table 3-1
24	~MCS2	36	See Table 3-1
25	~MCS3	35	See Table 3-1
26	~PCS0	25	See Table 3-1
27	~PCS1	27	See Table 3-1
28	~PCS2	28	See Table 3-1
29	~PCS3	29	See Table 3-1
30	~PCS4	30	See Table 3-1
31	~PCS5	31	See Table 3-1
32	~PCS6	32	See Table 3-1
33 : 63		unused	

POD STATUS SIGNALS

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	TYPE	POD MANUAL REFERENCE
0	ARDY	55	F,E	See Table 3-1
1	SRDY	49	F,E	See Table 3-1
2	HOLD	50	F,E	See Table 3-1
3	~RES	24	F	See Table 3-1
4	DRQ0	18	F	See Table 3-1
5	DRQ1	19	F	See Table 3-1
6	TMR_IN0	20		See Table 3-1
7	TMR_IN1	21		See Table 3-1
8	~TEST	47		See Table 3-1
9	NMI	46	I	See Table 3-1
10	~QSMD			See Section 3-10
11	ACT_INTR	(expanding)		See Section 3-9
12	INTR_V0			See Section 3-8

MASK BIT	POD SIGNAL	PROCESSOR PIN NO.	TYPE	POD MANUAL REFERENCE
13	INTR_V1			See Section 3-8
14	unused			
15	PWR_FAIL			See Section 3-7
16	INT0	45	I	See Table 3-1
17	INT1	44	I	See Table 3-1
18	INT2	42	I	See Table 3-1
19	INT3	41	I	See Table 3-1
20 : 63		unused		

TYPES: E - Enableable
F - Forcing
I - Interrupt

POD-SPECIFIC SETUP INFORMATION

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
POD_CTRL STBY_LO nn	0-FF	FF	See Section 4B-14
POD_CTRL STBY_HI n	0-3	0	See Section 4B-14
POD_CTRL RESET key	ON/OFF	OFF	See Section 4B-15
ERR_MSK ACT_FRC nn	0-3F	3F	See Section 4B-21
ERR_MSK ACT_INT nn	0-1F	1F	See Section 4B-22
ERR_MSK CTL_DRL nn	0-FF	FF	See Section 4B-20
ERR_MSK CTL_DRH nn	0-FF	FF	See Section 4B-20
ERR_MSK SEG_DR n	0-F	F	See Section 4B-23
ERR_MSK ADDR_DRL nn	0-FF	FF	See Section 4B-24
ERR_MSK ADDR_DRH nn	0-FF	FF	See Section 4B-24
ERR_MSK DATA_DR nn	0-FF	FF	See Section 4B-25
ERR_MSK INTA_TO n	0-F	F	See Section 4B-26
ERR_MSK CS_LO nn	0-FF	FF	See Section 4B-27
ERR_MSK CS_HI nn	0-1F	1F	See Section 4B-27

All of the following setups are used only as initialization values for the Peripheral Control Block functions. Refer to Appendix H of the 80188 pod manual for more information.

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
CS_REG MPCS_LO nn	0-FF	FB	See Section 4B-4 to 4B-9
CS_REG MPCS_HI nn	0-FF	A0	See Section 4B-4 to 4B-9
CS_REG MMCS_LO nn	0-FF	FB	See Section 4B-4 to 4B-9
CS_REG MMCS_HI nn	0-FF	81	See Section 4B-4 to 4B-9
CS_REG PACS_LO nn	0-FF	3B	See Section 4B-4 to 4B-9
CS_REG PACS_HI nn	0-FF	40	See Section 4B-4 to 4B-9
CS_REG LMCS_LO nn	0-FF	FB	See Section 4B-4 to 4B-9
CS_REG LMCS_HI nn	0-FF	3F	See Section 4B-4 to 4B-9
CS_REG UMCS_LO nn	0-FF	3B	See Section 4B-4 to 4B-9
CS_REG UMCS_HI nn	0-FF	C0	See Section 4B-4 to 4B-9
TIMER0 M/C_WDLO nn	0-FF	0	See Appendix H
TIMER0 M/C_WDHI nn	0-FF	0	See Appendix H
TIMER0 CNT_BLO nn	0-FF	0	See Appendix H
TIMER0 CNT_BHI nn	0-FF	0	See Appendix H
TIMER0 CNT_ALO nn	0-FF	0	See Appendix H
TIMER0 CNT_AHI nn	0-FF	0	See Appendix H
TIMER0 CNT_LO nn	0-FF	0	See Appendix H
TIMER0 CNT_HI nn	0-FF	0	See Appendix H
TIMER1 M/C_WDLO nn	0-FF	0	See Appendix H
TIMER1 M/C_WDHI nn	0-FF	0	See Appendix H
TIMER1 CNT_BLO nn	0-FF	0	See Appendix H
TIMER1 CNT_BHI nn	0-FF	0	See Appendix H
TIMER1 CNT_ALO nn	0-FF	0	See Appendix H
TIMER1 CNT_AHI nn	0-FF	0	See Appendix H
TIMER1 CNT_LO nn	0-FF	0	See Appendix H
TIMER1 CNT_HI nn	0-FF	0	See Appendix H
TIMER2 M/C_WDLO nn	0-FF	0	See Appendix H
TIMER2 M/C_WDHI nn	0-FF	0	See Appendix H
TIMER2 CNT_ALO nn	0-FF	0	See Appendix H
TIMER2 CNT_AHI nn	0-FF	0	See Appendix H
TIMER2 CNT_LO nn	0-FF	0	See Appendix H
TIMER2 CNT_HI nn	0-FF	0	See Appendix H

All of the following setups are used only as initialization value for the RUNUUT function. Refer to Appendix H of the 80188 pod manual for more information.

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
SEG_REG ES_LO nn	0-FF	0	See Appendix H
SEG_REG ES_HI nn	0-FF	0	See Appendix H

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
[illegible]_REG SS_LO nn	0-FF	0	See Appendix H
[illegible]_REG SS_HI nn	0-FF	0	See Appendix H
[illegible]_REG CS_LO nn	0-FF	FF	See Appendix H
[illegible]_REG CS_HI nn	0-FF	FF	See Appendix H
[illegible]_REG DS_LO nn	0-FF	0	See Appendix H
[illegible]_REG DS_HI nn	0-FF	0	See Appendix H
DMA_CH0 CTL_WDL nn	0-FF	0	See Appendix H
DMA_CH0 CTL_WDH nn	0-FF	0	See Appendix H
DMA_CH0 XFR_CT_L nn	0-FF	0	See Appendix H
DMA_CH0 XFR_CT_H nn	0-FF	0	See Appendix H
DMA_CH0 DP_LO nn	0-FF	0	See Appendix H
DMA_CH0 DP_MED nn	0-FF	0	See Appendix H
DMA_CH0 DP_HI n	0-F	0	See Appendix H
DMA_CH0 SP_LO nn	0-FF	0	See Appendix H
DMA_CH0 SP_MED nn	0-FF	0	See Appendix H
DMA_CH0 SP_HI n	0-F	0	See Appendix H
DMA_CH1 CTL_WDL nn	0-FF	0	See Appendix H
DMA_CH1 CTL_WDH nn	0-FF	0	See Appendix H
DMA_CH1 XFR_CT_L nn	0-FF	0	See Appendix H
DMA_CH1 XFR_CT_H nn	0-FF	0	See Appendix H
DMA_CH1 DP_LO nn	0-FF	0	See Appendix H
DMA_CH1 DP_MED nn	0-FF	0	See Appendix H
DMA_CH1 DP_HI n	0-F	0	See Appendix H
DMA_CH1 SP_LO nn	0-FF	0	See Appendix H
DMA_CH1 SP_MED nn	0-FF	0	See Appendix H
DMA_CH1 SP_HI n	0-F	0	See Appendix H
INT_REG INT3_LO nn	0-FF	F	See Appendix H
INT_REG INT3_HI nn	0-FF	0	See Appendix H
INT_REG INT2_LO nn	0-FF	F	See Appendix H
INT_REG INT2_HI nn	0-FF	0	See Appendix H
INT_REG INT1_LO nn	0-FF	F	See Appendix H
INT_REG INT1_HI nn	0-FF	0	See Appendix H
INT_REG INT0_LO nn	0-FF	F	See Appendix H
INT_REG INT0_HI nn	0-FF	0	See Appendix H
INT_REG DMA1_LO nn	0-FF	F	See Appendix H
INT_REG DMA1_HI nn	0-FF	0	See Appendix H
INT_REG DMA0_LO nn	0-FF	F	See Appendix H
INT_REG DMA0_HI nn	0-FF	0	See Appendix H
INT_REG TMR_LO nn	0-FF	F	See Appendix H
INT_REG TMR_HI nn	0-FF	0	See Appendix H
INT_REG STAT_LO nn	0-FF	00	See Appendix H
INT_REG STAT_HI nn	0-FF	80	See Appendix H
INT_REG REQ_LO nn	0-FF	0	See Appendix H
INT_REG REQ_HI nn	0-FF	0	See Appendix H
INT_REG INSRV_LO nn	0-FF	0	See Appendix H
INT_REG INSRV_HI nn	0-FF	0	See Appendix H
INT_REG PRMSK_L nn	0-FF	7	See Appendix H
INT_REG PRMSK_H nn	0-FF	0	See Appendix H

POD SETUP	RANGE/KEY	DEFAULT	POD MANUAL REFERENCE
INT_REG MASK_LO nn	0-FF	FD	See Appendix H
INT_REG MASK_HI nn	0-FF	0	See Appendix H
INT_REG EOI_LO nn	0-FF	0	See Appendix H
INT_REG EOI_HI nn	0-FF	0	See Appendix H
INT_REG VECT_LO nn	0-FF	0	See Appendix H
INT_REG VECT_HI nn	0-FF	0	See Appendix H
REL_REG LOW_BYTE nn	0-FF	FF	See Appendix H
REL_REG HI_BYTE nn	0-FF	0	See Appendix H

AVAILABLE TL/1 SUPPORT PROGRAMS

NOTE

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

CONFIG This program provides for easy interrupt mode configuration. See Sections 4A-20 through 4A-37 of the 80188 pod manual for further information.

Arguments : RMX_MODE - Sets the pod in RMX mode if this argument is non-zero. If RMX_MODE is non-zero, all other arguments to this function MUST be zero.

PAIR0 - A non-zero value configures INT0 and INT2/~INTA0 as an interrupt acknowledge pair. Otherwise, both lines are configured as inputs. This argument must be zero if RMX_MODE is set.

LVL_T_0 - If PAIR0 is set then a non-zero value will configure interrupt acknowledge pair 0 as level-triggered interrupts. Otherwise, PAIR0 is configured as an edge-triggered interrupt pair. If PAIR0 is zero, LVL_T_0 must be zero. Also, if RMX_MODE is set then LVL_T_0 must be zero.

PAIR1 - A non-zero value configures INT1 and INT3/~INTA1 as an interrupt acknowledge pair. Otherwise, both lines are configured as inputs. This argument must be zero if RMX_MODE is set.

LVL_T_1 - If PAIR1 is set then a non-zero value will configure interrupt acknowledge pair 1 as level-triggered interrupts. Otherwise, PAIR1 is configured as an edge-triggered interrupt pair. If PAIR1 is zero, LVL_T_1 must be zero. Also, if RMX_MODE is set then LVL_T_1 must be zero.

PRIORITY - If both PAIR0 and PAIR1 are set then PRIORITY determines which interrupt pair has priority. If PRIORITY is zero then pair 0 has priority. If PRIORITY is one then pair 1 has priority. PRIORITY must be set to zero if RMX_MODE is on. PRIORITY must also be set to zero if either PAIR0 or PAIR1 is off.

Faults : test_aborted
reason "Incompatible argument values".
The arguments do not conform to the specifications detailed above.

Returns : Nothing.

FRC_INT This program forces the execution of an interrupt acknowledge routine on either channel 0 or channel 1. For further information, see Section 4A-28 in the 80188 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted
reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : Nothing.

GET_PCB This program sets up the pod for PCB recovery (see 80188 manual appendix I), places the pod in RUNUUT mode, waits for the passed number of seconds, then pulls the pod out of RUNUUT mode. The program then copies the recovered PCB information into the 9100 setup, where it may be saved by using 9100 front panel commands.

Arguments : SECONDS - This is the number of seconds that the UUT will be left in RUNUUT mode before it is interrupted for PCB recovery.

Faults : test_failed

reason "NMI routine failed".
The pod-resident PCB capture routine failed.

reason "Pod RAM corrupted".
The pod RAM that is used to store PCB information is corrupt.

reason "Couldn't find PCB".
The PCB wasn't found by the pod-resident recovery routine.

Returns : Nothing. (The recovered information is left in the) (pod-specific setups on the 9100.)

QWK_FILL This program embodies all of the functions of the 80188 pod Quick Fill and Verify test. For further information about the usage of the Quick Fill and Verify test, see Section 4A-5 of the 80188 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

UPTO - This is the ending address of the Quick Fill test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of ADDR.

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 may not be greater than F.

FUNCTION - This determines whether the Quick Fill program performs a Quick RAM Fill, a Quick Pattern Verify, or both a Quick RAM Fill and a Quick Pattern 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 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 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 "Pod timeout occurred".
The pod timed out while performing this test. This indicates a drastic UUT fault which should be corrected by using tests with more diagnostic capability.

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.
bad_data_mask - A mask in which the set bits were read differently than they were written.

Returns : Nothing.

QWK_RAM This program embodies all of the functions of the 80188 pod Quick RAM test. For further information about the usage of the Quick RAM test, see Section 4A-4 of the 80188 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

UPTO - This is the ending address of the Quick RAM test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than 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 may not be be greater than F.

FUNCTION - This determines whether the Quick RAM program performs a Quick RAM test, a Quick Pattern Verify, or both a Quick RAM test and a Quick Pattern 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 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 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 "Pod timeout occurred".
The pod timed out while performing this test. This indicates a drastic UUT fault which should be corrected by using tests with more diagnostic capability.

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

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 4A-10 of the 80188 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping read. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

Faults : test_aborted

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

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 80188 pod Quick ROM test. For further information about the usage of the Quick ROM test, see Section 4A-6 of the 80188 pod manual. The major difference between the test that the pod manual 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. ADDR may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to FFFF.

UPTO - This is the ending address of the Quick ROM test. UPTO may be any numeric value between 0 and FFFFF unless the mainframe is in an I/O space, where UPTO must be less than or equal to FFFF. UPTO's value must be greater than that of 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 may not be greater than F.

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 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 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:
bad_data_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 checksum 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 4A-10 of the 80188 pod manual.

Arguments : ADDR - The address at which to perform the Quick Looping write. Legal values for this argument are any number between 0 and FFFFF unless the mainframe is in an I/O space, where ADDR must be less than or equal to 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 "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.

FRC_INT This program forces the execution of an interrupt acknowledge routine on either channel 0 or channel 1. For further information, see Section 4A-28 in the 80188 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : Nothing.

RD_CSCD This program returns the passed channel's cascade address as stored in the pod during the most recent execution of the interrupt acknowledge routine for that channel. For further information, refer to Section 4A-27 of the 80188 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel 1 invalid in iRMX mode".
Interrupt Acknowledge pair 1 does not exist in iRMX mode.

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The cascade address stored in the pod for the passed channel number from the most recent execution of the interrupt acknowledge routine for that channel.

RD_NOARM This program returns the most recent interrupt type information from the pod for the passed channel and neither re-enables interrupts nor clears the INT VECT status bit (status bit 11) for that channel. For further information, refer to Section 4A-26 of the 80188 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle for the passed channel.

RD_REARM This program returns the most recent interrupt type information from the pod for the passed channel and re-enables interrupts and clears the INT VECT status bit (status bit 11) for that channel. For further information, refer to Section 4A-26 of the 80188 pod manual.

Arguments : CHANNEL - The channel the interrupt is to be forced upon. Allowed values are 0 and 1. The channel must be configured as an interrupt acknowledge pair.

Faults : test_aborted

reason "Channel not configured as an INT/INTA pair".
The selected channel is not configured as an interrupt acknowledge pair. See the CONFIG program for information on configuring interrupts.

reason "Illegal channel number".
The CHANNEL argument is greater than 1.

Returns : The interrupt type information that was stored during the last interrupt acknowledge cycle for the passed channel.

POD SYNC MODES

NAME	MNEMONIC	CODE
Interrupt Sync	INTA	1
Address Sync	ADDR	A
Data Sync	DATA	D

POD SYNC CALIBRATION DATA

SYNC MODE	UUT SIGNAL	EDGE OF SIGNAL	OFFSET FROM EDGE
ADDR	ALE	FALLING	0 ns
DATA	~DEN	RISING	-30 ns
INTA	~DEN	RISING	-50 ns

ENABLEABLE LINES

The mnemonics below are used in the TL/1 podsetup 'enable' statement. The bit positions correspond to bit positions in the pod_timeout_enabled_line fault mask.

NAME	BIT POSITION	MNEMONIC
EXTRDY*	bit 0	EXTRDY*
HOLD	bit 1	HOLD

* The 80188 manual lists lines SRDY and ARDY as separately enableable lines. However, they are both enabled/disabled by bit 0 (EXTRDY) in the enableable line mask.

TL/1 FAULT CONDITIONS

This section lists the TL/1 fault conditions that can result from pod operation.

Handlers for most fault conditions are based on one of the 4 mask types: address, data, status, or control. See the Programmer's Information in the Introduction of this manual for information about the format of each fault mask.

Fault Conditions Using the Data Mask

pod_data_incorrect
pod_data_tied

Fault Conditions Using the Address Mask

pod_addr_tied

Fault Conditions Using the Status Mask

pod_forcing_active
pod_interrupt_active
pod_timeout_enabled_line

Fault Conditions Using the Control Mask

pod_control_tied

Fault Conditions With No Fault Mask

pod_timeout_bad_pwr
pod_timeout_no_clk
pod_timeout_recovered
pod_timeout_setup
pod_uut_power