DEBNA/DEBNK Installation Guide

Order Number: EK-DEBNX-IN-001

digital equipment corporation maynard, massachusetts

DEBNA/DEBNK Installation Guide

Order Number: EK-DEBNX-IN-001

digital equipment corporation maynard, massachusetts

First Printing, August 1987

The information in this manual is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

Digital Equipment Corporation makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of license to make, use, or sell equipment constructed in accordance with this description.

FCC NOTICE: The equipment described in this manual generates, uses, and may emit radio frequency energy. The DEBNA module has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

Contents

FIGURES

TABLES

Preface

Purpose of This Manual

This manual describes how to install the DEBNA and DEBNK modules. Both modules, referred to as DEBNx, provide an interface between the VAXBI bus and an Ethernet local area network. One form of this module—the DEBNK—also serves as an interface to the TK50 streaming tape drive.

This manual also describes diagnostics for these modules and gives advice for troubleshooting.

Intended Audience

This manual is for DIGITAL and customer personnel who install or replace the DEBNx module or option in the field.

Structure of This Document

This manual has four chapters and four appendixes, which are described below:

Chapter 1 briefly describes the DEBNx module: its functions and what you should have received.

Chapter 2 describes installation of the module.

Chapter 3 describes the module's self-test and how to interpret the results.

Chapter 4 describes the diagnostics that are available to test the functioning of the DEBNx module and how to run them. In addition to the ROM-based diagnostics, there are also software diagnostics for the DEBNx. The chapter also gives troubleshooting hints.

Appendix A gives environmental requirements for the DEBNx module.

Appendix B gives register information.

Appendix C describes booting with the DEBNx module.

Appendix D explains how to read the hardware Ethernet address.

Associated Documents

The DEBNx is one of a family of processors, memories, and adapters that use the 32-bit VAXBI bus. The *VAXBI Options Handbook* (EB-27271-46) provides a technical summary of all VAXBI modules.

Other related documentation includes:

DEBNA/DEBNK Technical Manual, EK-DEBNX-TM

Ethernet Installation Guide, EK-ETHER-IN

TK50-D, TK50-R Tape Drive Subsystem Owner's Manual, EK-LEP05-OM

Conventions Used in This Document

- All addresses are in hexadecimal (hex). All bit patterns are in binary notation. All other numbers are decimal unless otherwise indicated.
- Ranges are inclusive. For example, the range 0-4 includes the integers 0, 1, 2, 3, 4.
- Bits are enclosed in angle brackets (for example, <12>).
- Bit ranges are indicated by two bits in descending order separated by a colon; for example, <12:1>. Bit ranges are inclusive.
- K = kilo (1024); M = mega (1024**2); G = giga (1024**3).
- The term ''asserted'' indicates that a signal line is in the true state. The term ''deasserted'' indicates that a signal line is in the false state.

  ''Assertion'' is the transition from the false to the true state.

  ''Deassertion'' is the transition from the true to the false state.

1 Introduction

The term "DEBNx" is used to describe a family of intelligent I/O controllers for the VAXBI bus:

- DEBNA is an interface to an Ethernet local area network; it is compatible with the IEEE 802.3 Ethernet specification. DEBNA is the standard Ethernet interface for VAX 8xxx systems.
- DEBNK, in addition to an Ethernet controller identical to the DEBNA, has an onboard controller for a TK50 streaming tape drive. DEBNK is an option for OEM applications.

Both modules are designated T1034.[1] The modules can be distinguished by their VAXBI device type (BIIC register bb + 0), which is 410F for DEBNA and 410E for DEBNK. This manual uses the term "DEBNx" to refer to either or both. Specific TK50 information, such as microcode functions or diagnostics, of course pertains only to the DEBNK. Please see the *VAX Systems and Options Catalog* for specifications of which operating systems support the DEBNA and DEBNK options.

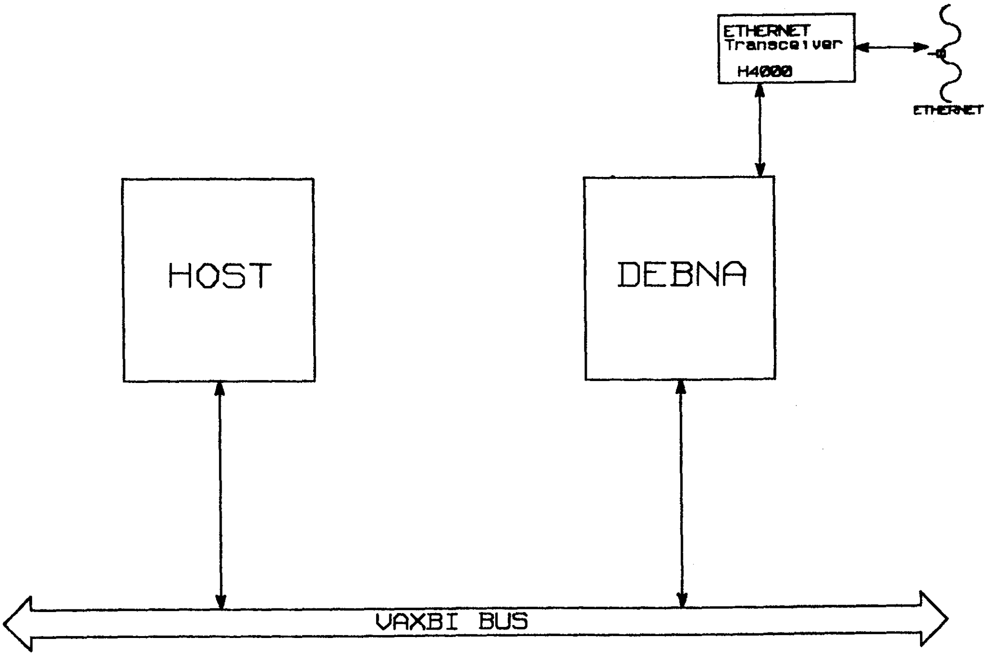
1.1 BASIC FUNCTIONS

The DEBNA supports one Ethernet port, which provides the physical and data link communication layers of the Ethernet protocol, as defined by the IEEE 802.3 Ethernet specification. The DEBNK supports one TK50 tape drive, which uses 1/2-inch cartridge tape.

With its own onboard MicroVAX processor, the DEBNx can control operations independently of the host processor. The details of Ethernet and tape transactions, including error correction and data transfer over the VAXBI bus, are thus transparent to the host processor (see Figure 1-1).

[1] An earlier version of DEBNA was DEBNT, designated T1032.

Figure 1-1 DEBNA in a VAXBI System



The Ethernet controller in the DEBNx lets the host processor communicate with other processors in an Ethernet local area network. The controller implements the complete Ethernet protocol and complies with the IEEE 802.3 Ethernet specification. In addition, the controller provides error detection and correction at the network management level.

The DEBNx has extensive on-board diagnostics. On power-up or reset the DEBNx tests itself and makes its status (pass or fail) available over the VAXBI bus. A set of LEDs on the printed circuit board also indicates the test results. In addition, a user may invoke other on-board diagnostics from the system console to test the DEBNx's logic and functionality more extensively.

The DEBNK's tape controller supervises tape operations (such as read, write, position, and rewind) and buffers data to and from the tape drive. The supported tape drive is the TK50, a 5.25-inch unit that drives a 0.5-inch cartridge tape at 75 inches per second. The tape controller also performs CRC error checking and ECC error checking and correction.

Figure 1-2 is a block diagram of the DEBNx module.

1.1.1 Ethernet/802.3 Controller

The Ethernet controller consists of the following chip set:

- A local area network controller for Ethernet (LANCE chip)
- A serial interface adapter (SIA chip)

The LANCE chip implements the data link layer of the Ethernet interface. The SIA chip implements the physical layer of the Ethernet protocol.

1.1.2 Tape Controller

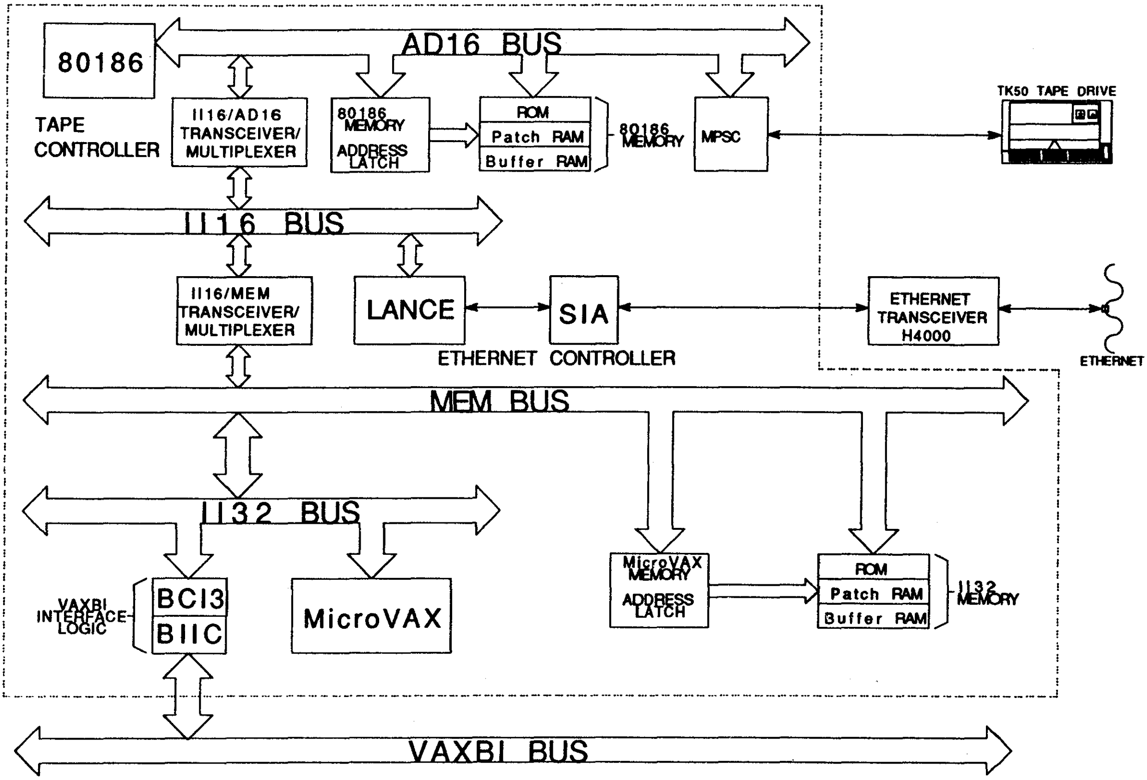
The tape controller consists of the following components:

- An 80186 microprocessor
- A multi-protocol serial controller (MPSC) chip
- 80186 memory and patch hardware

The 80186 is a 16-bit microprocessor that runs the tape controller's firmware.

The MPSC chip interfaces the 80186 processor to the tape drive. The MPSC performs parallel-to-serial and serial-to-parallel conversion, CRC generation, and CRC checking.

Figure 1-2 DEBNx Block Diagram



1.2 PHYSICAL DESCRIPTION

The DEBNA and DEBNK options both consist of a single board. The DEBNK has two cables—one to the tape drive and one to the Ethernet transceiver. The cables connect to the module on the VAXBI backplane. One cable is a 9-conductor, shielded cable with a male connector, and the other is a 26-conductor, flat-ribbon cable with a female connector.

Table 1-1 lists DEBNx cabinet kits, which must be ordered separately.

Three status-indicator lights on the module give a visual indication of whether the DEBNx module passes self-test. The green LED is for the TK50 tape drive.

Immediately after power-up or reset, the status-indicator lights are off. If the tape drive passes self-test, the controller's firmware lights the green Tape OK light. If all the executed tests pass (including the tape drive test), the firmware lights the yellow DEBNx OK lights.

Table 1-1 Cabinet Kits for DEBNx Options

System/Enclosure	Kit Number	Contents
VAX 8500, 8530, 8550, 8700, 8800	CK-DEBNA-LJ	5' internal Ethernet cable (17-01601-03) Ethernet I/O connector panel (70-22904-03) - includes 17-01601-03 Ethernet boot-enable jumper (17-01149-01) Ethernet loopback connector (12-22196-02)
VAX 8200, 8250, 8300, 8350	CK-DEBNA-LM	8' internal Ethernet cable (17-01601-02) Ethernet I/O connector panel (70-18799-00) Ethernet boot-enable jumper (as above) Ethernet loopback connector (as above)
	CK-DEBNK-LM	CK-DEBNA-LM, plus 8' internal TK50 cable and TK50 I/O connector panel (17-01550-01)
Expansion cabinet (H9652-EC/ED) for VAX 8500, 8530, 8550, 8700, 8800	CK-DEBNA-LN	15' internal Ethernet cable (17-01601-04) Ethernet I/O connector panel (70-18799-00) Ethernet boot-enable jumper (as above) Ethernet loopback connector (as above)

2 Installation

This chapter explains how to install the DEBNA and DEBNK options into a VAXBI computer system.

For complete instructions for installing a tape drive or Ethernet transceiver connected to a DEBNx, see the installation guides for the device and host computer system.

WARNINGS

POWER OFF—Shut off system power and disconnect the main system power cord before performing any procedure in this chapter.

STABILIZE THE CHASSIS—Make sure to extend the stabilizer leg(s) whenever you open the chassis.

WEAR ESD WRIST STRAP—You must wear an antistatic wrist strap that is connected to the processor cabinet whenever you work inside the chassis.

USE CONDUCTIVE CONTAINERS—Whenever you remove a circuit board from a VAXBI card cage, place it in a conductive container.

2.1 INSTALLATION

The following steps describe the installation process:

1 Power down the host computer system by:

  a Turning the POWER switch to the OFF position and

  b Setting the system circuit breaker in the rear to OFF.

2 Make sure you are wearing the ESD wrist strap that is attached to the system's chassis.

3 Open the chassis.

4 If necessary, install a transition header on the VAXBI backplane opposite the slot for the DEBNx. (The VAXBI transition header is part number 12-22246-01.) Use only the torque screwdriver (P/N 29-17381-00). Torque both screws to 5 +/- 1 inch-pounds.

5 Make sure there is a node ID plug in place. Note the number on the node ID plug; this number is now the installed module's VAXBI node ID.

6 Run the internal Ethernet cable from backplane segment E2 to the Ethernet I/O connector panel. See Figure 2-1 and Section 2.3.

7 If this is the first VAXBI Ethernet adapter in the system (for example, if you are replacing a DELUA connected through a DWBUA), skip the rest of this step and go to step 8.

When adding a second or third DEBNx, connect the pigtail connector from the internal Ethernet cable to a +15V two-prong connector from the power supply. Skip the next step and go to step 9.

NOTE

All the pigtail connectors from the power supply may already be used. (Some systems have one or two.) In this case, the external Ethernet cable from the I/O connector panel must not be connected directly to an H4000 transceiver. The cable may be connected either to a DECOM broadband transceiver or to a DELNI, which has its own power and which in turn may be connected to an H4000.

8 Connect the pigtail connector from the internal Ethernet cable to a +15V two-prong connector from the power supply.

NOTE

On a VAX 8200/8300 Model 821BA ONLY, you must first install the +15V connector to the power supply. The connector's part number is 17-00684-02.

9 If you wish the host to respond to MOP Boot messages over the Ethernet (trigger boot), install the boot-enable jumper on backplane segment E1. See Figure 2-1 and Section 2.4.

10 For a DEBNK, run the internal TK50 cable from backplane segment D2 to the TK50 I/O connector panel. See Figure 2-1 and Section 2.5.

11 Locate the card cage slot opposite the DEBNx transition header.

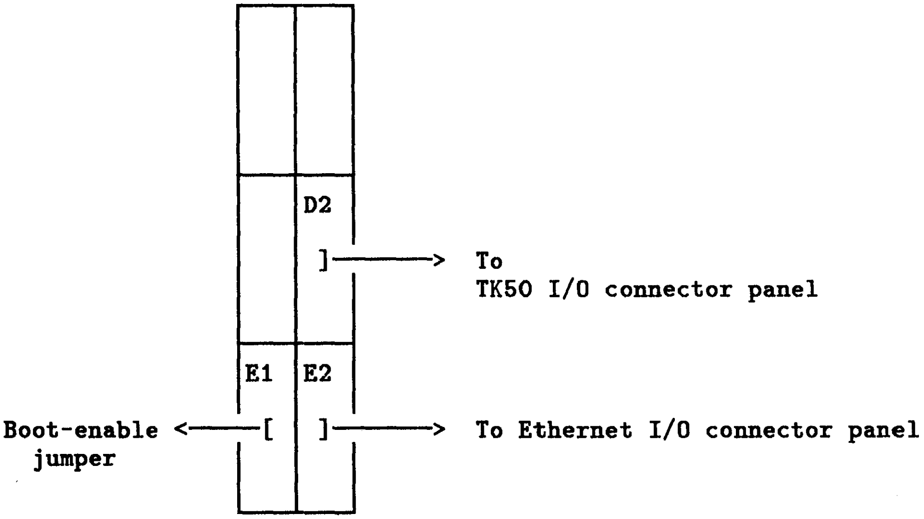
12 Lift the locking lever to open the slot.

13 Slide the DEBNx module into the card cage slot until it stops: this is a zero insertion force card cage.

14 Close the locking lever.

15 Verify installation as described in the following section.

Figure 2-1 Cabling at the DEBNx VAXBI Connector



2.2 INSTALLATION VERIFICATION PROCEDURE

Follow these steps to verify that the DEBNx is properly installed:

1 Turn the system power on and verify that the DEBNx passes self-test. (See Chaper 3.) If self-test fails, check to see that the module is properly seated in the card cage. If the module continues to fail self-test, swap in a different DEBNx module if available. You can also try installing the module in a different slot.

2 After the module passes self-test, boot the operating system.

3 Configure the network database and start DECnet traffic on the Ethernet.

4 If the system is unable to communicate over the Ethernet, verify that the network software is installed and configured properly. Check the hardware by running loopback tests with the ROM-based D1 diagnostics. You can run these diagnostics either from a VAX 8250/8350 system console or through the level 3 diagnostic EVDYC under the VAX Diagnostic Supervisor. See Chapter 4 for details.

5 For the DEBNK, copy a file to a tape and read it back. If the drive does not function properly, check the hardware by running the ROM-based D2 diagnostics. You can run these diagnostics either from a VAX 8250/8350 system console or through the level 3 diagnostic EVMDD under the VAX Diagnostic Supervisor.

2.3 INTERNAL ETHERNET CABLE

The internal Ethernet cable connects the VAXBI backplane to the I/O connector panel (see Table 1-1 for part numbers). The external cable is ordered separately; it runs from the outside of the connector panel to an Ethernet transceiver, such as an H4000 baseband transceiver, DECOM broadband transceiver, or DELNI local network interconnect.

On the backplane, the internal cable plugs into segment E2 opposite the DEBNx slot. (Viewed with the node ID plugs at the top, E2 is the right side of the segment farthest from the node ID plugs. See Figure 2-1.) The +15V pigtail connector supplies power to the H4000 transceiver; however, you should plug it in (if possible) regardless of the type of transceiver. Table 2-1 lists pinouts of the internal Ethernet cable connector at the VAXBI cage.

Table 2-1 Ethernet Cable Connector Pinouts

Pin	Signal	Type	Description
E01-E04	UNCONNECTED		
E05-E09	LOGIC GROUND		
E10 E11	ETHERNET COLLISION L ETHERNET COLLISION H	I I	Differential collision detect signals from the Ethernet bus.
E12 E13	ETHERNET RECEIVE L ETHERNET RECEIVE H	I I	Differential receive signals from the Ethernet bus.
E14 E15	ETHERNET TRANSMIT L ETHERNET TRANSMIT H	O O	Differential transmit signals to the Ethernet bus.
E16 E46	ETHERNET BOOT ENABLE	I	Enables remote rebooting over the Ethernet bus. The signal is asserted if there is a jumper connection between pins E16 and E46 in the DEBNx section of the VAXBI backplane. The signal is sent to the Status Register.

2.4 BOOT-ENABLE JUMPER

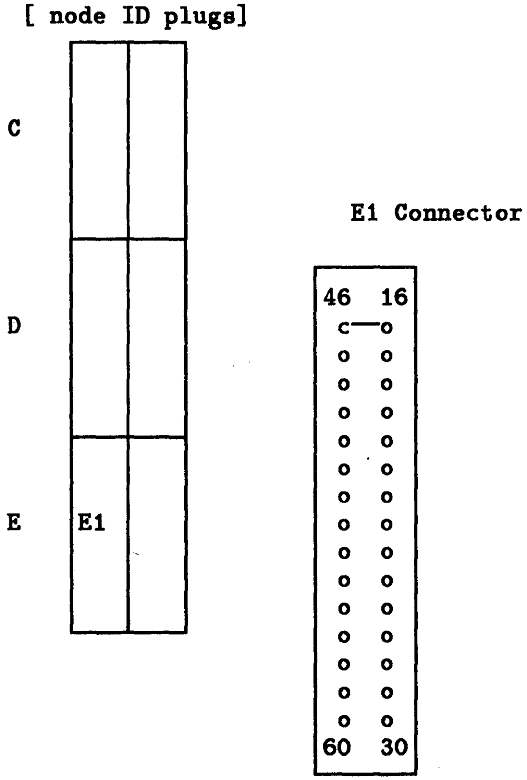
Install the boot-enable jumper if you want the system to accept an Ethernet trigger boot—a MOP Boot message sent by another node in the network. (Without the jumper, DEBNx will ignore Boot messages.)

The jumper is not required for you to boot the system over the Ethernet using a console command like:

```
>>>B ETA0
```

The jumper (P/N 17-01149-01) is a 30-pin connector that is installed on segment E1 of the VAXBI backplane, opposite the DEBNx slot. (Viewed with the node ID plugs at the top, E1 is the left side of the segment farthest from the node ID plugs.) The jumper connects pins E16 and E46, as shown in Figure 2-2.

Figure 2-2 Ethernet Boot-Enable Jumper



2.5 INTERNAL TK50 CABLE FOR THE DEBNK MODULE

The internal TK50 cable connects the VAXBI backplane to the I/O connector panel. The external cable is included with the TK50 desktop or rackmount unit (ordered separately).

On the backplane, the cable plugs into segment D2 opposite the DEBNK's slot (D2 is the right side of the middle I/O segment; see Figure 2-1). Table 2-3 lists the pinouts of the TK50 cable connector at the VAXBI cage.

Table 2-2 Tape Drive Cable Connector Pinouts

Pin	Signal	Type	Description
D01	DRIVE PRESENT		
D31	UNCONNECTED		
D02	DRIVE WRITE CLOCK H	I	Differential WRITE CLOCK signals.
D32	DRIVE WRITE CLOCK L	I	
D03	DRIVE STATUS L	I	Differential serial drive status data signal (low).
D33	GROUND		
D04	GROUND		
D03	DRIVE STATUS H	I	Differential serial drive status data signal (high).
D05	DRIVE READ DATA L	I	Differential read data signal.
D35	DRIVE READ DATA H	I	
D06	DRIVE READ CLOCK H	I	Differential READ CLOCK signal (high).
D36	GROUND		
D07	GROUND		
D37	DRIVE READ CLOCK L	I	Differential READ CLOCK signal (low).
D08	DRIVE ERASE ENABLE L	O	Differential signals from the Erase Enable Register.
D38	DRIVE ERASE ENABLE H	O	
D09	DRIVE WRITE GATE ENABLE L	O	Differential signal (low) from the Drive Write Gate Enable Register.
D39	GROUND		
D10	GROUND		
D40	DRIVE WRITE GATE ENABLE H	O	Differential signal (high) from the Drive Write Gate Enable Register.
D11	DRIVE DR COMMAND H	O	Differential serial drive command signals.
D41	DRIVE DR COMMAND L	O	
D12	DRIVE WRITE DATA L	O	Differential write data signal (low).
D42	GROUND		
D13	GROUND		
D43	DRIVE WRITE DATA H	O	Differential write data signal (high).

2.6 REMOVAL

To replace a DEBNx option, follow these steps:

1 Power down the host computer system by:

  a Turning the POWER switch to the OFF position and

  b Setting the system circuit breaker to OFF.

2 Make sure you are wearing an ESD wrist strap that is attached to the system's chassis.

3 Open the chassis.

4 On the module-installation side of the open chassis:

  a Locate the desired card cage slot.

  b Lift the locking lever to open the slot.

  c Slide the circuit board out of the card cage slot.

  d Put the board into a conductive container.

  e Close the locking lever.

5 On the cabling side of the open chassis:

  a Locate the same card cage slot.

  b If another DEBNx module will be installed, leave the cables and jumpers in place.

3 Power-Up Self-Test

The DEBNx's power-up self-test consists of ROM-based diagnostic routines that run automatically on power-up or reset. The power-up self-test verifies that the hardware at the node is operational and, for a DEBNK, that a tape drive is present.

Since the routines are contained in ROM, their execution requires no operating system. The self-tests are thus stand-alone programs independent of any software environment.

3.1 BIIC POWER-UP SELF-TEST

On power-up or reset, the BIIC performs its own self-test.

- If the BIIC passes its internal self-test, it sets the Broke, Initialization, and Self-Test Status (STS) bits in its VAXBI Control and Status Register (VAXBICSR). The set STS bit indicates that the BIIC has passed its internal self-test.
- If the BIIC fails its internal self-test, it disables its VAXBI bus drivers. This action prevents the DEBNx from writing corrupted data to the VAXBI bus and denies other nodes access to the DEBNx's registers. (The STS bit in the VAXBICSR remains clear, indicating to the DEBNx that the BIIC failed its self-test.)

3.2 HOW TO RUN SELF-TEST

There are various ways to run self-test for the DEBNx:

1. On system power-up—When the user powers up the host system, the DEBNx automatically runs power-up self-test. Front panel controls vary among host systems; see the owner's manual for the specific system.
2. On processor reset—When the user presses the reset or restart button on the host system's front panel, the host processor runs a system self-test, which causes each VAXBI node to run its own self-test.
3. From the console—A user can invoke the DEBNx's D0 tests from the system console of a VAX 8250/8350. (These are the same tests that run during power-up or reset self-test.)

In the following example, the user uses VAX console commands to contact the DEBNx, which is node 5:

```
<CTRL/P>
>>> Z 5

    T/R

RBD5> DO
--tests run--
--results appear--
RBD5> QUIT
<CTRL/P>
>>>
```

3.3 REPORTING SELF-TEST RESULTS

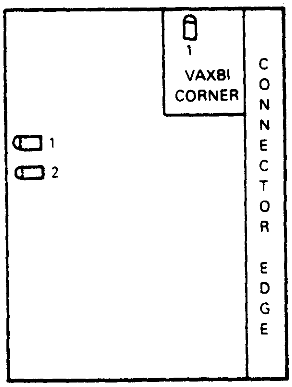
Test results (pass or fail) are indicated in three ways:

- The status of the node as a whole is made available over the VAXBI bus.
- LEDs—There are three status-indicator lights on the module:
  - 2 DEBNx OK LEDs (yellow)
  - 1 Tape OK LED (green; means tape is present)

  The location of the LEDs is shown in Figure 3-1.
- Registers
  - The VAXBICSR and the Port Status Register provide summary information on the state of the DEBNx during and after self-test.
  - If a self-test error occurs, the Port Error Register contains an error code, and the Port Data Register contains a copy of the Power-Up Diagnostic Register (PUDR), which provides additional information.

The TK50 tape drive executes its own power-up self-test from ROM inside the drive. If a tape is properly loaded in the drive, the drive's servo motors are exercised by the test, and the tape is rewound to the Beginning of Tape (BOT) mark after the test finishes. The drive does not read or write the tape during the test. The outcome of the test is indicated by a red LED on the drive. If the drive fails the test, the red LED flashes rapidly.

Figure 3-1 Self-Test LED Locations



3.3.1 Self-Test Results on VAXBI Bus

On power-up, the BIIC sets the Broke bit in the VAXBICSR, and the DEBNx asserts BI BAD. If the node as a whole passes power-up self-test, the DEBNx clears the Broke bit and deasserts BI BAD. If the node does not pass power-up self-test, the Broke bit remains set and BI BAD remains asserted.

3.3.2 Self-Test Results in LEDs

The yellow LEDs (DEBNx OK LEDs) show the status of the module after the node self-test. The green LED (Tape OK LED) on the DEBNK module shows the presence of the TK50 cable and drive.

At power-up or reset, all the LEDs are off. If the DEBNx passes all the executed tests, the DEBNx's firmware lights the DEBNx OK LEDs; otherwise, these LEDs remain off. If the DEBNK firmware determines that a tape drive is present (not necessarily powered on, but at least cabled), it lights the Tape OK LED.

3.3.3 Self-Test Results in VAXBI Registers

- **VAXBICSR and PS** —The VAXBICSR and the Port Status (PS) Register provide a comprehensive picture of the DEBNx's functional level during and after self-test. The following bits in these two registers indicate increasing levels of functionality.
  - The Self-Test Status (STS) bit in the VAXBICSR indicates whether the BIIC passed its self-test.
  - The Adapter Can Communicate (ACC) bit in the PS Register indicates whether the DEBNx is minimally functional. (1 = yes; 0 = no). Minimum functionality is defined as follows: the host and DEBNx are able to access the DEBNx's port registers; the ports are able to enter the MAINTENANCE state; the port must be able to execute the supported BI VAX Port (BVP) maintenance instructions.
  - The Broke bit in the VAXBICSR indicates whether the DEBNx's node as a whole is functional (1 = pass; 0 = fail).
- **PUDR** —The Power-Up Diagnostic Register (PUDR) shows the results of the power-up self-test in detail, including Ethernet results and, on the DEBNK, tape results. There are three copies of the (PUDR).
- **PS** —The Self-Test Done bit in the PS Register indicates whether the self-test completed execution (1 = test completed; 0 = test in progress).
- **PE and PD** - If self-test passes, both the Port Error (PE) and Port Data (PD) registers contain zero. If self-test fails, the PD contains the PUDR and the PE contains an error code (see Table 3-1).

NOTE

The host should not examine the PE and PD registers until the Self-Test Done (STD) bit in the PS is set by the DEBNx.

Table 3-1 indicates how the VAXBICSR and the PS Register can be used to determine the state of the DEBNx during and after self-test. The table also indicates when the PE and PD registers should be read for additional error information.

Table 3-1 State of DEBNx During and After Self-Test

VAXBICSR		PS Register			
STS	**Broke**	**STD**	**ACC**	**State of DEBNx**	**Additional Information**
1	1	0	0	The BIIC passed its self-test; self-test is still in progress.	None
1	1	1	0	The DEBNx failed self-test and is not minimally functional.	PD Register contains the PUDR. PE Register contains FF00 0003.
1	1	0	1	The DEBNx is minimally functional; self-test is still in progress.	PD Register contains a running count during self-test.
1	1	1	1	The DEBNx is minimally functional, but a noncritical component failed self-test.	PD Register contains the PUDR. PE Register contains FF00 0004.
1	0	1	1	The DEBNx node is fully functional; self-test has passed.	PD and PE registers contain zero.

3.4 INTERPRETING TEST RESULTS

For the VAXBI node as a whole to pass self-test, all the executed subtests must pass. The node may thus be usable to a large degree even though, as a whole, it did not pass self-test. For example, if the Ethernet controller in the DEBNK does not pass power-up self-test, BI BAD remains asserted and the Broke bit remains set. However, the DEBNK could possibly be used for tape transactions.

After most self-test failures, system software can examine the VAXBICSR, PS, PE, and PD registers to determine which components are usable. Based on this information, the system can decide that the node is still usable for certain tasks. If the BIIC chip fails its self-test, however, it disables its VAXBI drivers. This puts the entire node offline. In this case, the BVP registers are inaccessible.

If the PD Register indicates that all the components have failed, the problem is probably the BIIC, MicroVAX, or MicroVAX ROM. If one of these components fails, the self-test routine stops, and the MicroVAX enters a wait state.

If the DEBNK's PD Register indicates that the tape drive failed power-up self-test, the problem may be one of the following:

- The tape drive is, in fact, malfunctioning.
- The tape drive cable is defective or is not connected properly.

In any case, the user should look at the tape drive's LED to determine whether the drive passed its own self-test. If the LED is flashing rapidly, the drive failed the test.

If the DEBNK's PD Register indicates that the tape controller chips failed self-test, the problem could be any of the following:

- The MicroVAX's testing of communication with the 80186 failed.
- The 80186's testing of communication with the MicroVAX failed.
- Any of the tested components in the tape controller failed.

Finally, self-test could fail because of a systemwide fault. For example, a faulty power supply or missing VAXBI bus terminators could be the problem. Make sure that system power is OK and check for other possible systemwide faults.

3.5 TESTED COMPONENTS

Self-test tests the following components on the DEBNx module:

- MicroVAX
- MicroVAX ROM
- MicroVAX RAM
- MicroVAX patch hardware
- BIIC and BCI3 chips
- LANCE chip
- MicroVAX-80186 communication

On the DEBNK, the self-test also tests the tape controller chips, including:

- 80186 microprocessor
- Multi-protocol serial controller (MPSC)
- 80186 gap-detection hardware
- 80186 patch registers

- 80186 patch PAL
- 80186 buffer and patch RAM
- 80186 ROM
- 80186 tape control registers

3.6 UNTESTED COMPONENTS AND FUNCTIONS

The following components and functions are not tested:

MicroVAX:

- A complete instruction set (only a small subset of the instruction set is tested)

Ethernet controller functions:

- More (multiple retries of packet transmission)
- One (one retry of a packet transmission)
- Babble error
- Time domain reflectometry
- Late collision
- Loss of carrier
- Memory error

In the BCI3/BIIC logic:

- BIIC nodespace locking
- BIIC nodespace Write Sense bits
- BCI3 datamove operations with quadwords and octawords
- BCI3 Nonlocal Memory Reference (NLMR) line

In the DEBNK's tape controller:

- A complete 80186 instruction set (only a small subset of the instruction set is tested)
- Module transceivers out to the drivers

4 Diagnostics and Troubleshooting

This chapter describes diagnostics that test the DEBNx module. Some are ROM-based diagnostics, which are resident on the module; others are software diagnostics, which are loaded and run under the VAX Diagnostic Supervisor (VDS). Customers must purchase software diagnostics under a separate license.

The ROM-based diagnostics (RBDs) fall into three categories:

- D0 — DEBNx Self-Test
- D1 — NI (Network Interconnect) Tests
- D2 — Tape Drive Tests

The D0 diagnostic verifies the functionality of the DEBNx module. The D1 diagnostic verifies the Ethernet link from the Ethernet controller to the Ethernet transceiver. The D2 diagnostic verifies the integrity of the tape drive and media (only for the DEBNK module).

NOTE

> **The ROM-based diagnostics are not supported by the T1032, module type 0.**

On a VAX 8250/8350, all the ROM-based diagnostics can be invoked from the operator's console. On all VAX 8xxx systems, the D1 and D2 diagnostics can be invoked through the VAX Diagnostic Supervisor (VDS) using the level 3 interface programs and the standard VDS operating procedures.

Section 4.4 provides detailed information on the status and error reports generated by the diagnostics.

Section 4.5 lists the software diagnostics for the T1032 and T1034 modules, which can be run under VDS.

4.1 ROM-BASED DIAGNOSTICS

The tests in each category are described in detail in the following sections. Individual tests, or test groups, can be invoked with commands from the operator's console or, for the D1 or D2 tests, executed through the VAX Diagnostic Supervisor (VDS) (see Section 4.5). If a test-fault occurs, deductive reasoning, along with a comprehensive understanding of the test routine, is required to determine which hardware element is malfunctioning.

4.1.1 D0—DEBNx Self-Test

The DEBNx self-test, which is functionally equivalent to the DEBNx's power-up self-test, performs a basic confidence check of the module. Table 4-1 describes the tests in the DEBNx self-test diagnostic. Unless otherwise specified in the table, the test execution parameters and method of fault reporting can be selected with the appropriate run-time switches. If a test number is not specified with the D0 execute command, all tests are executed by default.

When any D0 test is selected, an internal self-check of the BIIC chip is first executed to verify the VAXBI bus communications logic. After a 250-millisecond delay, which allows the BIIC chip to complete its internal self-check, the Self-Test Status (STS) bit in the VAXBI Control and Status Register is checked. If the BIIC chip passed its self-check, the STS bit is set. If the BIIC failed the self-check, the STS bit is clear and the VAXBI bus registers are disabled. When this is the case, further communication with the DEBNx module is inhibited.

Table 4-1 D0—DEBNx Self-Test

Diagnostic/Test	Name	Description
D0/T1	ROM CRC	Calculates a longword CRC character for each ROM section. Compares calculated CRC to corresponding stored CRC.
D0/T2	RAM	First verifies byte mask logic. Then verifies non-stack area of RAM. Finally, moves stack into verified RAM and verifies untested stack area. (Also verifies data and address lines as a result of RAM tests.)
D0/T3	Patch	Disables patch function and toggles each patch register bit. Enables patch function and executes single patch operation.
D0/T4	MicroVAX Critical Path	Checks critical timing and principal micro-instructions.
D0/T5	IRQ Lines	Steps through all IPLs (interrupt priority levels) to verify that no IRQ line is stuck asserted.
D0/T6	Interval Timer	Verifies interval timer operation.
D0/T7	BI Corner	Performs seven subtests to verify BIIC and BCI3 chips. Exits to next test if test fails.
D0/T8	NI LANCE Chip	Verifies LANCE chip and CRC logic in the Ethernet controller. (SIA chip is not tested.)
D0/T9*	ROM CRC	Calculates a longword CRC character for each ROM section. Compares calculated CRC to corresponding stored CRC.
D0/T10	RAM	First verifies byte mask logic. Then verifies non-stack area of RAM. Finally, moves stack into verified RAM and verifies untested stack area. (Also verifies data and address lines as a result of RAM tests.)
D0/T11	Patch	Disables patch function and toggles each patch register bit. Enables patch function and executes single patch operation.
D0/T12	CPU/Timer	Verifies basic functionality of both the 80186 CPU and the programmable timers.
D0/T13	Miscellaneous Registers	Enables and disables the Write Drive Gate, Drive Erase, and Patch Enable Registers.
D0/T14	Gap Detect	Verifies gap detection logic in tape controller.
D0/T15	MPSC	Verifies multi-protocol serial controller (MPSC) chip.

*Tests 9-16 are 80186 tests.

Table 4-1 (Cont.) D0—DEBNx Self-Test

Diagnostic/Test	Name	Description
D0/T16	CPU Communications	Verifies communication and interrupt operations between the 80186 and MicroVAX CPUs.
D0/T17	Drive Present	Checks that drive present bit is set in Tape Status Register.

4.1.2 D1—Network Interconnect Diagnostic

The Network Interconnect (NI) diagnostic consists of the two test routines described in Table 4-2. These tests verify that the Ethernet controller, cable, link, and transceiver are operational. If a test number is not specified when this diagnostic is selected, test 2 is executed by default. Tests 1 and 2 should not be invoked at the same time because their run-time environments are different.

Table 4-2 D1—Network Interconnect Tests

Diagnostic/Test	Name	Description
D1/T1	External Loopback (with connector)	Transmits data out through the SIA chip and verifies that the same data is received back through the loopback connector.
D1/T2	External Loopback (on live Ethernet)	Transmits data out through the SIA chip and verifies that the same data is received back from the Ethernet bus.

Typically, test 2 is run first to verify the entire Ethernet system. This test verifies that the Ethernet link from the Ethernet controller (LANCE/SIA subsystem), through the transceiver cable, to the transceiver is functioning properly. If this test passes, further Ethernet testing is not required.

NOTE

For test 2 to execute properly, the DEBNx module must be connected to an operational Ethernet bus, a DELNI in loopback mode, or an H4080 transceiver. The test may fail if there is excessive traffic on the bus.

Test 2 is automatically retried if any of the following errors is detected:

- Loss of carrier
- Retry error
- CRC error
- Framing error
- Babble error
- Missed packet error
- Overflow error
- Buffer error

Test 2 fails if any of the following occurs:

- 32 retries
- Transmission error
- Descriptor ring error
- CRC error
- Memory error
- Underflow error

If test 2 fails, test 1 can be run to isolate the faulty field-replaceable unit (FRU). Follow these steps:

1. Disconnect the external Ethernet transceiver cable (BNE3) at the transceiver end.
2. Install a loopback connector on the cable.
3. Run test 1 and observe one of the following:
   - If test 1 passes, the transceiver is bad. Replace the transceiver, reconnect the cable to the new transceiver and run test 2 to verify proper operation. No further action is required.
   - If test 1 fails, one of the following is bad: transceiver cable, internal Ethernet cable, backplane, or DEBNx module. Go to the next step.
4. Disconnect the external transceiver cable at the I/O connector panel and install a loopback connector in its place.

5 Rerun test 1 and observe one of the following:

- If test 1 passes, the transceiver cable is bad. Replace the cable and run test 2 to verify proper operation. No further action is required.
- If test 1 fails, one of the following is bad: internal Ethernet cable, backplane, or DEBNx module. Replace the internal Ethernet cable. Go to the next step.

6 Rerun test 1 and observe one of the following:

- If test 1 passes, the removed cable is bad. Run test 2 to verify proper operation. No further action is required.
- If test 1 fails, either the DEBNx module or the backplane is bad. If the module passes self-test, it is probably good, but replace it and go to the next step.

7 Rerun test 1 and observe one of the following:

- If test 1 passes, the removed DEBNx module is bad. Run test 2 to verify proper operation. No further action is required.
- If test 1 fails, the backplane is bad. Install the DEBNx module in a different slot. Run test 2 to verify proper operation. Consider replacing the card cage.

4.1.3 D2—Tape Drive Diagnostic

The tape drive diagnostic consists of the tests described in Table 4-3. Tests 1-6 detect hard (nonintermittent) errors. Tests 7-9 detect hard or soft media errors. This diagnostic should be run when: the DEBNx tests fail to isolate a malfunction, the tape errors are intermittent, or the media integrity is questionable. Since some of the tape tests destroy user data, a confirm (/C) run-time switch is required to execute these tests. Use a scratch or blank tape for the destructive tests.

If no test numbers are specified, the diagnostic runs tests 1-6.

Table 4-3 D2—Tape Drive Tests

Diagnostic/Test	Name	Description
D2/T1	Initialization	Verifies that drive can be initialized to a valid state by resetting the drive and then checking the drive status, hardware revision, and software revision levels.
D2/T2	Online and Calibration	Rewinds tape to BOT and verifies that drive read/write heads can be calibrated.
D2/T3	Tape Motion	Verifies that drive can move tape rapidly forward and backward.
D2/T4	Write (destructive)[1]	Verifies that drive can write tape by first rewinding to BOT and then writing a test pattern on track 0. Test fails with Initialization error if tape cartridge is write-protected or drive is powered down.
D2/T5	Read	(Pre-written tape required.) Verifies that drive can read tape by rewinding to BOT and then reading track 0. Test fails if read error occurs or tape does not contain fixed-length records.
D2/T6	Write/Read (destructive)[1]	Verifies that drive can write, reposition, and then read tape. Read data is compared byte-for-byte to write data.
D2/T7	Thrashing Write (destructive)[1]	Verifies drive functionality by writing individual test pattern records to tracks 0 and 1. After each record is written, tape is stopped and repositioned before another record is written. Requires approximately 15 minutes to complete.
D2/T8	Thrashing Read	(Pre-written tape required.) Verifies drive functionality by reading individ- ual records to LEOT (logical end of tape) from tracks 0 and 1. After each record, tape is stopped and repositioned before another record is read. Requires approximately 25 minutes to complete.

[1] The command for these tests requires the /C (confirm) switch.

Table 4-3 (Cont.) D2—Tape Drive Tests

Diagnostic/Test	Name	Description
D2/T9	Thrashing Write/Read (destructive)[1]	Verifies drive functionality by writing and reading individual test pattern records from tracks 0 and 1. After each record, tape is stopped and repositioned before another record is written or read. Requires approximately 40 minutes to complete test if no parameter is given. A test of the entire tape (parameter of 1) takes about 8 hours.

[1] The command for these tests requires the /C (confirm) switch.

4.2 HOW TO RUN ROM DIAGNOSTICS FROM THE CONSOLE

The ROM-based diagnostics are stand-alone programs that can be controlled and executed from the operator's console of a VAX 8250 or 8350. The D1 and D2 tests, along with the other diagnostics listed in Section 4.5, can also be executed through the VAX Diagnostic Supervisor (VDS).

To run the RBDs from the operator's VAXBI console, do the following:

1 Enter console mode by typing the following characters at the system prompt:

```
>>>Z n
    T/R
RBDn>
```

where *n* is the node ID of the node under test (0-F hex)

2 Select the desired diagnostic test(s) by entering the appropriate Execute command along with the applicable run-time switches.

3 Terminate the diagnostics by entering the QUIT command or one of the control characters described below.

Four control characters affect the operation of the ROM-based diagnostics. Generally, these characters are used to abort test routines, reset the DEBNx, or exit from the console mode as described in Table 4-4. To enter a control character, press and hold down the CTRL key and then simultaneously press the desired character key once.

CTRL/C and CTRL/Y abort test execution and return program control to the command-entry level prompt (RBDn>). If either character is entered when a test is not executing, the character is echoed on the screen and the command-entry prompt is reissued.

CTRL/U functions only at the command-entry level to abort the current command line and reissue the command-entry prompt. This character should be used when a mistake is made while typing a command.

CTRL/Z serves two functions. If a test routine is executing when this character is entered, the routine is aborted and program control is returned to the command-entry level. If a test is not executing when this character is entered, the DEBNx is reset thereby executing a power-up self-test.

CTRL/P terminates the console mode and returns program control to the system-level prompt regardless of the test execution status. If a test routine is executing when this character is entered, the routine continues; however, program control returns to the system level.

Table 4-4 Console Mode Control Characters

Character	When an RBD Is Executing	When the Parser Is Executing
CTRL/C	Aborts test routine, rewinds to BOT, executes cleanup code, and returns to parser and reissues the RBDn> prompt.	Echoes "^C", reissues RBDn> prompt.
CTRL/U	Ignored.	Echoes "^U", aborts current command line, and reissues RBDn> prompt.
CTRL/Y	Aborts test routine, and returns to the diagnostic prompt. Does not execute cleanup code.	Echoes "^C", reissues RBDn> prompt.
CTRL/Z	Aborts test routine, executes cleanup code, and returns to command-entry level.	Resets the DEBNx and executes power-up self-test; same as QUIT command.
CTRL/P	Terminates console mode and returns to system-level prompt (>>>)	Terminates console mode and returns to system-level prompt (>>>). CTRL/Z or QUIT must be used before CTRL/P to put adapter in a known state.

4.3 ROM DIAGNOSTIC CONSOLE COMMANDS

The ROM diagnostic console commands are as follows:

- D0, D1, D2—the Execute commands
- Deposit
- Examine
- Quit

Each command is described in detail in the following sections. Examples are also included. In all cases, when an illegal or invalid command is entered, the keyboard alarm sounds, a question mark is displayed, and the command-entry prompt is redisplayed as shown.

```
RBDn>Invalid Command
 ?
RBDn>
```

4.3.1 Execute Commands (D0, D1, D2)

The Execute commands invoke a designated test, or group of tests, within a specified diagnostic. The command format is:

```
Dn[/Sw1/Sw2/...:Sw9  pc]
```

Dn

Diagnostic number (n = 0, 1, or 2)

Sw

Switch (see Table 4-5)

pc

Parameter code. The only parameter is 1, which is supported only by test 9 of D2. A pc of 1 means test the entire tape; no parameter means test 2 tracks of the tape.

NOTE

The /C (confirm) switch is required for all data-destructive tests. If this switch is not included when data-destructive commands are entered, the just-entered command line is echoed on the screen. When this occurs, type /C to execute the command or just press RETURN to abort the command.

Table 4-5 Execute Command Switches

Switch	Name	Default	Description
/BE	Bell on Error	No bell	Sounds keyboard alarm each time an error is detected.
/C	Confirm	No data destroyed	Confirms data-destructive test.
/DS	Disable Status	Status reports	Disables status reports.
/HE	Halt on Error	Continue on error	Diagnostic halts on test that detects first error and returns to command-entry prompt.
/IE	Inhibit Errors	Error reports	Disables error reports.
/IS	Inhibit Summary	Summary reports	Disables summary reports.
/LE	Loop on Error	Continue on error	Diagnostic loops on test that detects first error, even if error is intermittent. Type CTRL/C or CTRL/Y to terminate and return to command-entry prompt (RBDn> displayed).
/P=n	Pass Count	One pass	Specifies total number of diagnostic passes. (One pass equals one iteration of all selected tests.) A pass count of 0 selects an infinite number of passes. Type CTRL/C or CTRL/Y to terminate and return to command-entry prompt.
/QV	Quick Verify	Normal test	Executes Quick Verify version of specified test, when applicable; otherwise, is ignored.
/T=n[:m]	Test Number[s]	Unique to each diagnostic	Specifies test or group of tests to be executed. If group is specified, tests are executed in ascending numerical order.
/TR	Enable Trace	No trace	Enables trace reports.

EXAMPLES OF SELF-TEST EXECUTE COMMAND (D0)

1

```
D0
```

Executes one pass of all tests.

2

```
D0/P=0/T=4:6/BE
```

Executes tests 4 through 6 in loop-forever mode with bell-on-error active and trace reports disabled by default. Summary, status, and error reports enabled by default.

```
3   D0/P=1/TR/HE
```

Executes one pass of all tests, by default, in halt-on-error mode with trace reports enabled. Summary, status, and error reports enabled by default.

```
4   D0/T=8/LE/BE
```

Executes one pass, by default, of test 8 in loop-on-error mode with bell-on-error active and trace reports disabled by default. Summary, status, and error reports enabled by default. (If test 8 fails initially, diagnostic loops forever, even though test may subsequently pass. To stop, use CTRL/C.

EXAMPLES OF NI EXERCISER EXECUTE COMMAND (D1)

```
1   D1
```

Executes one pass of D1 test 2 (which is default test executed).

```
2   D1/T=1/P=0
```

Executes test 1 for infinite passes. To stop, use CTRL/C.

EXAMPLES OF TK50 TAPE EXERCISER EXECUTE COMMAND (D2)

```
1   D2/TR/T=9/C 1
```

Runs long version of test 9 of tape exerciser with trace enabled for one pass. The /C (confirm) switch is required for data-destructive tests.

```
2   D2/T=1:9/C/P=2/DS
```

Runs all tape exerciser tests for two passes. Test 9 runs in short mode since no parameter was given. Status reports are disabled.

4.3.2 Deposit and Examine Commands

The Deposit and Examine commands are generally used with each other to deposit data into VAXBI registers and memory locations or to read the contents of VAXBI registers and memory locations. Both commands can be used in the following formats:

D* = Deposit into current location

D+ = Deposit into next location

D- = Deposit into previous location

D/L = Deposit longword

D/W = Deposit word

D/B = Deposit byte

E* = Examine current location

E+ = Examine next location

E- = Examine previous location

E/L = Examine longword

E/W = Examine word

E/B = Examine byte

EXAMPLES OF EXAMINE AND DEPOSIT COMMANDS

1

```
RBD5>E/L 1FFD0000
1FFD0000 0D25410D
```

Examine 1FFD0000; it contains 0D25410D.

2

```
RBD5>D* 0
RBD5>D+ 0
RBD5>D+ 0
RBD5>D+ 0
```

Deposit 00000000 into 1FFD0000, 1FFD0004, 1FFD0008, and 1FFD000C.

3

```
RBD5>E*
1FFD0000 00000000
```

Examine 1FFD0000; it contains 00000000.

4

```
RBD5>D/W* A5A5
```

Deposit word A5A5 into 1FFD0000.

```
5   RBD5>E/L
    1FFD0004 0000A5A5
```

Examine longword at 1FFD0004; it contains 0000A5A5.

```
6   RBD5>E+
    1FFD0008 00000000
```

Examine longword at 1FFD0008; it contains 00000000.

```
7   RBDn>D/B* A5A5
    ?
```

Try to use a "byte" command to deposit an illegal byte value into the current location.

```
8   RBDn>D/B* A5
```

Deposit byte A5 into 1FFD0008.

```
9   RBDn>E/W*
```

Examine word at 1FFD0008; it contains 000000A5.

NOTE

When the *, +, or - delimiter is used without a length designator, the previous designator is used. Hence, the E* in step 3, examines a longword because the previous D* command designated a longword operation.

4.3.3 Quit

The QUIT command sets the Node Reset bit in the VAXBI Control and Status Register, thereby initializing the DEBNx and thus executing a power-up self-test. CTRL/Z has the same function. The command format is:

```
QU[IT]
```

4.4 TEST STATUS AND ERROR REPORTING

ROM-based diagnostics have five types of status and error reports:

1. Error reports
2. Status reports
3. Trace headers and messages
4. Diagnostic completion messages
5. Summary reports

The following sections describe each report in detail.

4.4.1 Error Reports

Three types of errors are reported:

- System fatal errors
- Device hard errors
- Device soft errors

A system fatal error is an error that prevents the diagnostic from running to completion. Such errors include unexpected machine checks during diagnostic initialization, powerfail interrupts, and certain exceptions. A system fatal error causes the diagnostic to abort.

A device hard error is an error that prevents the tested device from being able to perform the current operation under test. A device hard error is thus an unrecoverable error. After reporting a device hard error, the diagnostic performs the action specified in the invoking command: halt on error, loop on error, or continue on error (default).

A device soft error is an intermittent hardware error that probably will go away if the test is repeated. A device soft error is thus a recoverable error. After reporting a device soft error, the diagnostic performs the action specified in the invoking command: halt on error, loop on error, continue on error (default).

Error reports for D1 and D2 tests have three lines; error reports for D0 tests have three lines plus the contents of the PUDR.

Figure 4-1 shows a sample error report. Table 4-6 lists the fields in each line in Figure 4-1. Table 4-7 defines the fields. Table 4-8 defines the error codes for line 3, field 1.

Figure 4-1 Sample Error Report

```
;      F         4      410F 000000002
;      SE        NI        00        T02
;      69   A5A5A5A5 00A5A5A5 000000000 00000010 1FF81020
```

```
Line 1:  F = fail
         4 = VAXBI node number
         410F = DEBNA module
         00000002 = pass count (hex)

Line 2:  SE = soft device error
         NI = network interconnect (Ethernet link)
         00 = unit number
         T02 = test number

Line 3:  69 = error code (see Table 4-8)
         A5A5A5A5 = expected data
         00A5A5A5 = received data
         00000000 = SCB offset (not applicable)
         00000010 = RAM location of received data
                    (expressed as hex offset)
         1FF81020 = ROM PC value at which error was
                    detected
```

Table 4-6 Error Report Fields

Line	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
1	Status	Node No.	Device Type	Passes		
2	Err Type	ASCII L	Unit No.	Test		
3	Err Code	Expected	Received	SCB	Address	PC

Table 4-7 Error Report Field Definitions

Mnemonic	Name	Description
LINE 1		
--	Status	The report type. For error reports, P = passed, F = failed.
NODE NO.	Node Number	The node under test. The value may range from 0-F (hex).
--	Device Type	The type of module under test. 410F (hex) = DEBNA; 410E (hex) = DEBNK
PASS	Pass Count	The execution pass at which· the error was detected (hex.)
LINE 2		

ERR TYPE	Error Type	The error type:
		FE = system fatal error
		HE = hard device error
		SE = soft device error

ASCII L	ASCII Logic	An ASCII code that indicates the failing logic:
		ROM = ROM
		UVAX = MicroVAX
		RAM = RAM
		BCI = BCI3
		80186 = 80186 processor
		NI = Ethernet interface
		TKCTL = tape controller
		TK = TK50 tape drive

UNIT	Unit	Number The unit under test (always 0).
TEST	Test. Number	The test that detected the error. A value of zero indicates that the initialization code for the test failed.

Table 4-7 (Cont.) Error Report Field Definitions

Mnemonic	Name	Description
LINE 3		
ERR CODE	Error Code	The nature of the detected error. See Table 4-8 for a list of error codes.
EXPCT	Expected Data	The expected data for certain types of data comparison errors.
RCVD	Received Data	The incorrect data that was received for a data comparison.
SCB	System Control Block Offset	The system control block offset through which an interrupt was expected or received.
ADD	Address	The memory location or register address at which a data comparison error or register operation error was detected.
PC	Program Counter	The PC (program counter) value in ROM at which the error was detected.

Table 4-8 Error Codes

Error Code	Description
ERRORS FROM D1 TESTS	
55	No packet transmitted.
56	The packet transmitted has an incorrect destination or source address.
57	No packet received.
58	The packet received has an incorrect destination or source address.
59	No LANCE interrupt was received after the LANCE was started.
60	The STOP bit was not set, or it was not the only bit set in LANCE CSR0 when the LANCE was stopped.
61	Initialization of the LANCE was not successful.
62	The low byte of LANCE CSR0 was not as expected after packet transmission.
63	The MISS bit was set in LANCE CSR0 indicating a missed packet.
64	The BABL bit was set in LANCE CSR0 indicating a transmitter timeout.
65	32 soft errors have occurred, and another soft error condition has arisen.
66	The UFLO bit was set in LANCE transmit descriptor 3 indicating a truncated transmitted packet.
67	32 retries have occurred, and another retry condition has arisen.
68	The MERR bit is set in LANCE CSR0 indicating a memory error.
69	The CERR bit is set in LANCE CSR0 indicating loss of heartbeat.
99	8 retries due to LANCE bugs have already occurred, and another retry condition has arisen.

Table 4-8 (Cont.) Error Codes

Error Code	Description
ERRORS FROM D2 TESTS	
70	Command was aborted.
71	Controller error was detected.
72	Read/write data error detected.
73	Drive error was detected.
74	Invalid command detected.
75	Unit is offline.
76	Position lost error.
77	Record data truncation error.
78	Serious exception detected.
79	Hardware write-protect error.
80	Illegal interrupt from 80186.
81	Interrupt message from 80186 bad.
82	No cartridge detected in drive.
83	Tape not loaded into drive.
84	Unexpected interrupt exception.
85	Controller initialization error.
86	MicroVAX-80186 communication error.
87	Controller timeout error.
88	Data overrun error detected.
89	LEOT encountered unexpectedly.
90	BOT encountered unexpectedly.
91	Unexpected TapeM encountered.
92	No command echo received within required time.
93	Drive motion test error: drive unable to perform the motion test, which consists of several drive motion commands.

Table 4-8 (Cont.) Error Codes

Error Code	Description
94	Tape load error: drive unable to successfully unload and reload a tape.
95	Unsolicited byte from drive: drive sent an illegal, unsolicited byte. Certain bytes are allowed at any time.
96	Test timed out while waiting for a response from the drive.
97	80186 did not request a new buffer.

4.4.2 Status Reports

Status reports are printed during long-running tests to indicate that the diagnostic is still running and to show the status of the diagnostic. A status report consists of two lines (see Figure 4-2). Table 4-9 lists the fields in each line. Table 4-10 defines the fields.

Figure 4-2 Sample Status Report

```
;              S        F      410E 00000200
;             XX       TK50      00       T09
```

```
Line 1:  S = status report
         F = node F (hex)
         410E = DEBNK module
         00000200 = pass 200

Line 2:  XX = undefined field
         TK50 = tape drive
         00 = unit number
         T09 = test 9
```

Table 4-9 Status Report Fields

Line	Field 1	Field 2	Field 4	Field 4
1	Status	Node No.	Device Type	Passes
2	Err Type	ASCII L	Unit No.	Test

Table 4-10 Status Report Field Definitions

Mnemonic	Name	Description
Line 1		
--	Status	Report type. S = status report.
NODE NO.	Node Number	Node under test. The value may range from 0-F (hex).
--	Device Type	The type of module under test. 410E (hex) = DEBNK module
PASS	Pass Count	Number of passes completed so far (hex).
Line 2		
--	XX	Undefined field.
ASCII L	ASCII Logic	An ASCII code that indicates the logic currently being tested: TK50 = TK50 tape drive
UNIT	Unit Number	The unit under test. Always 0.
TEST	Test Number	The currently executing test.

4.4.3 Trace Headers and Messages

A trace header is printed at the start of diagnostic execution. If trace mode is enabled (with the /TR switch), a trace header is printed at the start of diagnostic execution, and a trace message is printed at the start of each test's execution.

A trace header is an ASCII mnemonic that indicates what logic is currently being tested along with the revision level of the code.

A trace message consists of a ''T'' followed by the number of the currently executing test. For example, ''T02'' indicates that test 2 is currently executing. A trace message is printed for each pass of the test.

Figure 4-3 shows a sample trace header and message.

Figure 4-3 Sample Trace Header and Message

```
;          TK_50   1.00
;
;          T01   T02   T03   T04   T01   T02   T03   T04
```

```
TK_50        = the diagnostic being executed
1.00         = the diagnostic revision
T01 - T04    = tests 1-4, run for two passes each
```

4.4.4 Diagnostic Completion Messages

A diagnostic completion message (see Figure 4-4) is printed immediately after a diagnostic completes execution and before a summary error report.

A diagnostic completion message consists of one line with the fields shown in Table 4-11. Figure 4-5 illustrates a sample diagnostic completion message. The fields are defined in Table 4-12.

Figure 4-4 Sample Diagnostic Completion Message

```
 ;            P         6       410F 00000020
```

```
P = pass
6 = node 6
410F = DEBNA module
00000020 = 20 passes (hex)
```

Table 4-11 Diagnostic Completion Fields

Field 1	Field 2	Field 3	Field 4
Status	Node No.	Device Type	Pass Count

Table 4-12 Diagnostic Completion Message Field Descriptions

Mnemonic	Name	Heading
--	Status	Indicates whether the diagnostic passed or failed. P = pass. F = fail.
NODE NO.	Node Number	The node under test. The value may range from 0-F (hex).
--	Device Type	The type of module under test. 410E = DEBNK 410F = DEBNA
PASS	Pass	The number of passes executed (hex).

4.4.5 Summary Reports

If not disabled (with the /IS switch), a summary report (see Figure 4-5) is printed directly below the diagnostic completion message. A summary report is printed for each pass of the diagnostic. In addition, the user may invoke a summary report by entering the Summary command.

A summary report consists of one line with seven fields. Table 4-13 shows the summary report fields, and Table 4-14 defines the fields.

Figure 4-5 Sample Summary Report

```
;00000000 00000100 000000A0 0000100F 00010000 00010000 00002400
```

```
00000000 = unit 0 (bit mask)
00000100 = 256 hard errors (hex)
000000A0 = 160 soft errors (hex)
0000100F = 4,111 retries (hex)
00010000 = 65,536 reads (hex)
00010000 = 65,536 writes (hex)
00002400 = 9,216 seeks (hex)
```

Table 4-13 Summary Report Fields

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7
UUT MASK	HARD ERR	SOFT ERR	RETRIES	READS	WRITES	SEEKS

Table 4-14 Summary Report Field Definitions

Mnemonic	Name	Heading
UUT MASK	Unit Under Test Mask	An 8-bit mask that indicates which units were tested. The bits represent units 7-0, proceeding from left to right. In the DEBNx, there is only one Ethernet unit (unit 0) and one tape control unit (unit 0).
HARD ERR	Hard Errors	The total number of hard errors detected (hex).
SOFT ERR	Soft Errors	The total number of soft errors detected (hex).
RETRIES	Retries	The total number of retries attempted (hex).
READS	Reads	The total number of read operations performed (hex).
WRITES	Writes	The total number of write operations performed (hex).
SEEKS	Seeks	The total number of track seeks performed (hex). Defined only for certain tape diagnostics.

4.5 SOFTWARE DIAGNOSTICS

Table 4-15 lists the diagnostics that support the DEBNx module. The EVDYC and EVMDD diagnostics can be used to interface with the D1 and D2 ROM-based diagnostics using standard operating procedures under the VAX Diagnostic Supervisor (VDS).

Table 4-15 Down-Line Loaded Diagnostic Programs

Program	Diagnostic Level	Description
EVDYC	3	Provides user interface to D1 RBDs.
EVMDD	3	Provides user interface to D2 RBDs.
EVDYD	2R	Tests the functioning of the NI port.
EVDWC	2R	Tests the installation of the host Ethernet node and all other nodes on the local Ethernet that support MOP protocol.
EVMDA	2R	Tests the TK50 tape drive.

The level 3 diagnostics in Table 4-15, EVDYC and EVMDD (the D1 and D2 tests) are run as follows:

1 Boot the VAX Diagnostic Supervisor.

2 Enter the following commands at the DS> prompt:

```
DS> ATTACH DEBNK HUB ETA n        ! n is the node under test
DS> ATTACH LANCE ETA ETAO
DS> SELECT ETAO
DS> RUN EVDYC
```

The level 2R diagnostics are run as follows:

1 Boot VMS.

2 Run the VAX Diagnostic Supervisor.

3 At the DS> prompt enter the commands as shown in the example above.

4.5.1 EVDYC Program

The EVDYC program provides a way for VDS users to run the DEBNx's D1 ROM-based diagnostic tests. D1 is the Network Interconnect diagnostic.

4.5.2 EVMDD Program

The EVMDD program provides a way for VDS users to run the DEBNK's D2 ROM-based diagnostic tests. D2 is the tape drive diagnostic.

The EVMDD program divides the D2 tests into the following five sections:

- DEFAULT—Tests 1 through 6. These tests destroy tape data.
- ALL—All the D2 tests.
- NONDESTRUCT—Tests 1 through 3, which check the tape drive's ability to move the tape and calibrate itself.
- DATA—Tests 4-6, which write the tape, read the tape, and then compare tape writes and reads. These tests destroy tape data.
- THRASHING—Tests 7-9, which perform thrashing writes, thrashing reads, and thrashing write/reads. These tests destroy tape data.

The user can specify the sections above with the /SECTION qualifier.

4.5.3 EVDYD Program

The EVDYD program makes sure that the DEBNx's NI port performs all the functions that the VAX/VMS Ethernet port driver, ETDRIVER, may request. Customers for ETDRIVER include DECnet.

EVDYD has seven tests:

1. Transmit CRC test
2. Receive CRC test
3. Promiscuous Address test
4. Enable All Multicast test
5. Station test
6. No Receive Buffer Available test
7. Stress test

Test 7 uses either internal or external loopback facilities:

- Internal loopback—If you do nothing, test 7 will use the LANCE chip's internal loopback.
- External loopback—You must
    1. Install a loopback connector on the I/O connector panel or on the Ethernet transceiver cable.
    2. Select external loopback by setting the VAX Diagnostic Supervisor event flag 3:

       ```
       DS> SET EVENT FLAG 3
       ```

    3. Answer "yes" to the prompt

       ```
       Connector in place?
       ```

4.5.4 EVDWC Program

The EVDWC program tests the installation of the host Etheret node and checks the connectivity of all other nodes on the local Ethernet.

The functions of the EVDWC program are as follows:

- Monitors DECnet ID messages on the Ethernet
- Builds a node table for the local Ethernet
- Conducts loopback testing, monitors network traffic, and permits users to view contents of network packets
- Provides online help for users

There are limits on this program's use of the available channels on the NI section of the DEBNx.

4.5.5 EVMDA Program

The EVMDA program tests the TK50 tape drive; it exercises the drive, makes sure it can read and write data, checks the read and written data for validity, and keeps statistics.

A Environmental Requirements

Operating Environment	
Temperature	5 degrees C to 50 degrees C (41 degrees F to 122 degrees F)
Humidity	10% to 95% with maximum wet bulb of 32 degrees C (89.6 degrees F) and minimum dew point of 2 degrees C (36 degrees F) noncondensing
Altitude	To 2.4 km (8,000 ft)

Storage Environment	
Temperature	-40 degrees C to 66 degrees C (-40 degrees F to 151 degrees F)
Humidity	To 95% noncondensing
Altitude	To 9.1 km (30,000 ft)

B Registers

The DEBNx VAXBI registers are divided into two groups—BIIC and BCI3 registers. The BIIC registers include the standard VAXBI registers (see the *VAXBI Options Handbook*) and the tape port registers. The BCI3 registers include the Receive Console Data (RXCD) Register and the NI (Network Interconnect) registers.

Table B-1 gives a brief description of how some of these registers are used in the DEBNx module. Each register is then described in detail.

Table B-1 VAXBI Registers as Used in the DEBNx Module

Register	Use
DTYPE	Identifies device as DEBNA or DEBNK and gives module type.
GPR 0 (BCI3)	RXCD Register.
GPRs 1-4 (BCI3)	BI VAX Port Registers (NI).
GPRs 0-3 (BIIC)	BI VAX Port Registers (tape).
SADR, EADR	Always set to zero.
RXCD	Invokes ROM-based diagnostics and reports results.
PUDR	Gives the results of self-test.

B.1 DEVICE REGISTER (DTYPE)

The Device Register contains information that identifies the option (see Figure B-1).

Figure B-1 Device Register

	31 24	23 16	15 0
bb+0	MODULE TYPE	REVISION LEVEL	DEVICE TYPE

Bits <31:24> give the module type:

- 0 or 1 = T1032 board
- 2 = T1034 board

Bits <23:16> give the revision level:

- Hexadecimal number 0–FF

Bits <15:0> give the device type:

- 410F for the DEBNA
- 410E for the DEBNK

B.2 RECEIVE CONSOLE DATA (RXCD) REGISTER

A user at the system console can write the Receive Console Data (RXCD) Register (Figure B-2) to invoke the DEBNx's self-test or complementary diagnostics. The RXCD Register is also used to send test results back to the console. The DEBNx's firmware does not support any other functions for the RXCD Register.

The RXCD can be defined in two ways, depending on whether the host implements an RXCD. If the host has an RXCD, the RXCD Register in the DEBNx has only one set of fields. If the host does not have an RXCD, the RXCD Register in the DEBNx has two identical sets of fields. The higher-order set is used as the host's RXCD. Table B-2 defines the RXCD bits.

Figure B-2 Receive Console Data (RXCD) Register

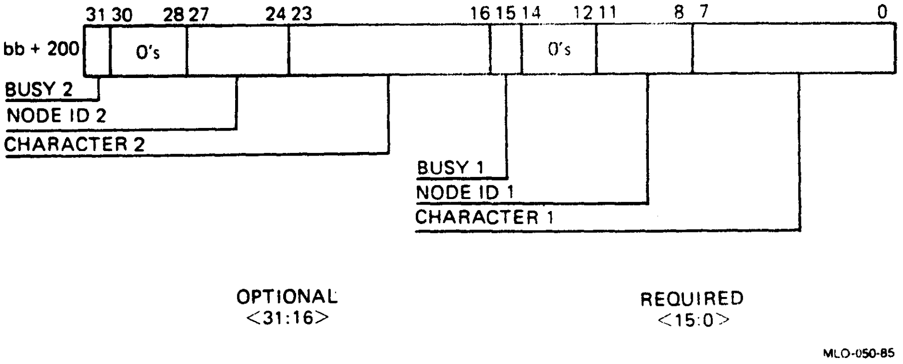


Table B-2 Receive Console Data (RXCD) Register Bit Definitions

Bit	Mnemonic	Name	Type	Description
31	--	Busy 2	R/W	When set, indicates that the CHAR2 field contains a character that has not yet been read by the host. The host clears the bit after reading the CHAR2 field. The DEBNx cannot write the CHAR2 field until the bit is clear.
30:28	--	RESERVED and zeros	RO	These bits are RESERVED and are always read as zeros.
27:24	--	Node ID 2	R/W	Written by the DEBNx to indicate that data in the CHAR2 field is from the DEBNx.
23:16	CHAR2	Character 2	R/W	Contains the console command character being sent from the DEBNx to the host.
15	--	Busy 1	R/W	When set, indicates that the CHAR1 field contains a character that has not yet been read by the DEBNx. The RXCD Register clears this bit after reading the CHAR1 field. The sending node cannot write another character to the CHAR1 field until the bit is cleared.
14:12	--	RESERVED and zeros	RO	These bits are RESERVED and are always read as zeros.
11:8	--	Node ID 1		Written by the sending node. Indicates the node that has sent the data in the CHAR1 field.
7:0	CHAR1	Character 1		Written by the sending node with a console command character or a console message character.

B.3 BI VAX PORT REGISTERS

The BIIC general purpose registers are used as port registers for the tape port in the DEBNK, and BCI3 GPRs 1-4 are used as port registers for the NI (Ethernet) port in the DEBNA and DEBNK. The use of the port registers is specified by the BI VAX Port (BVP) architecture, which defines the interface between the BVP port drivers resident in the host and the BVP ports resident in the DEBNx. The tape port driver and tape port in the DEBNK transfer data between the host and the DEBNK's tape controller. The NI port driver and NI port physically transfer data between the host and the DEBNx's NI port. Since both sets of port registers are identical, the following description pertains to both sets.

The port registers, collectively called the Port Register Block (PRB), include the following (Figure B-3):

- Port Control (PC) Register
- Port Status (PS) Register
- Port Error (PE) Register
- Port Data (PD) Register

In general, the port driver writes the PC register with commands for the port, and the port writes the PS, PE, and PD registers with status information for the port driver. In addition, the DEBNx uses the PE and PD registers to report self-test results to the host.

B.3.1 Self-Test Results in Port Error Register

Immediately after power-up self-test, the Port Error Register contains zero if self-test passed. If self-test failed, PE contains an error code from Table B-3.

Figure B-3 BI VAX PORT Registers

TAPE PORT REGISTERS

VAXBI ADDRESS	31 0	NODE PRIVATE SPACE ADDRESS
bb + F0	PORT CONTROL REGISTER	2008 00F0
bb + F4	PORT STATUS REGISTER	2008 00F4
bb + F8	PORT ERROR REGISTER	2008 00F8
bb + FC	PORT DATA REGISTER	2008 00FC

NI PORT REGISTERS

	31 0	
bb + 204	PORT CONTROL REGISTER	2008 0204
bb + 208	PORT STATUS REGISTER	2008 0208
bb + 20C	PORT ERROR REGISTER	2008 020C
bb + 210	PORT DATA REGISTER	2008 0210

Table B-3 Self-Test Error Codes in PE Register

Code	Description
FF00 0003	The DEBNx has not attained minimum functionality. A test from test 1 through test 7 has failed.
FF00 0004	The DEBNx has attained minimum functionality but a noncritical component has failed; for example, the LANCE/NI test failed.

B.3.2 Self-Test Results in Port Data Register

Immediately after power-up self-test, the PD register contains zero if self-test passed. If self-test failed, PD contains the PUDR.

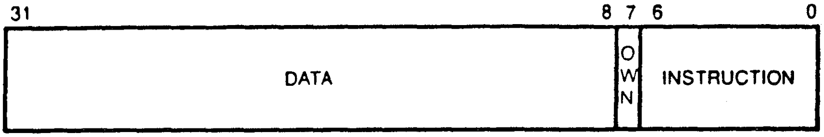
B.3.3 Port Control Register

Figure B-4 gives the format of the Port Control Register.

B.3.4 Port Status Register

Figure B-5 gives the format of the Port Status Register.

Figure B-4 Port Control Register



Where: Own 0 = Owned by host
1 = Owned by port

BU-2766

If the ETYPE field in the Port Status Register contains Fatal BI Error, Non-Fatal BI Error, or Transient BI Error, then the Port Error Register contains the contents of the Bus Error Register (BER) at the time of the error, and the Port Data Register contains the physical address of the failing transaction.

If the ETYPE field contains any other values (Adapter Exception X02, Data Structure Error X05, or Port Logical Error X06), then the Port Error Register contains an error code as follows:

Figure B-5 Port Status Register

31	30	29	28	27	26	25	24 19	18 16	15 8	7	6	5 0
OWN	RPSQ	STD	ACC	XSTP	ERL	FQE	RSVD	PST	ETYPE	RSQ	SUME	RSVD

OWN	0 = Owned by host
	1 = Owned by port
RPSQ	1 = Response to port status query
STD	1 = Self-test done
ACC	1 = Adapter can communicate
XSTP	1 = Extended self-test passed
ERL	1 = Error lost
FQE	1 = Free queue exhausted
PST	Port state:
	001 = Undefined
	010 = Initialized
	100 = Enabled
	110 = Stopped
	111 = Maint
ETYPE	= Error type:
	01 = Transient BI error
	02 = Adapter exception
	03 = Nonfatal BI error
	04 = Fatal BI error
	05 = Data structure error
	06 = Port logical error
	07 = Adapter hardware error
RSQ	1 = Response queued to host
SUME	1 = Summary error: error type field is valid

BU-2788

Error	Meaning
01	Adapter Hardware failure
02	Adapter Bugcheck Port Data register contains bugcheck code. These will be the same as the controller error codes listed under the controller error log packet.
03	Failed minimal functionality self-test
04	Failed BI specified self-test
05	Failed extended self-test
08	Invalid PQB PD register contains physical address of bad entry.
09	Queue interlock retry timeout PD contains address of error
0A	Illegal queue offset PD contains address of error
0B	Illegal port instruction PD contains snapshot of PC
0C	Port instruction illegal in current port state PD contains PS at time of the error
0D	Adapter timeout on clearing of PS ownership bit
0E	Adapter timeout due to host inactivity
0F	Free queue exhausted PD may contain index of exhausted free queue
10	Failed to enter MAINT state
11	Failed to enter INITIALIZED state
12	Failed to enter ENABLED state
13	Buffer key mismatch PD contains buffer name used
14	Data transfer larger than size of buffer
15	Buffer access check error
16	Virtual address translation failed PD contains the failed address
17	BDT index out of range PD contains address of the buffer name
18	Power failure
19	Host interrupt timeout PD contains the PIV contents
1A	Queue entry size violation PD contains queue entry address
1B	Address invalid for maintenance instruction PD contains the invalid address

B.4 POWER-UP DIAGNOSTIC REGISTER (PUDR)

The Power-Up Diagnostic Register (PUDR) displays the results of the DEBNx's self-test, which runs automatically on power-up or reset. After the entire self-test finishes, the operating system can read the register to determine which components on the DEBNx module passed self-test.

The PUDR operates as follows:

- The PUDR is initialized to all zeros on power-up or reset.
- When a subtest in the self-test routine passes, its corresponding bit in the PUDR sets.
- If a subtest fails, its corresponding bit remains cleared.

The PUDR of a DEBNx that passes self-test has the following value:

- The DEBNA—7FF80FDF (no tape drive is present)
- The DEBNK—7FFC1FDF (the tape bits are set)

Figures B-6 and B-7 show bit maps of the register, and Table B-4 provides bit definitions.

Figure B-6 Power-Up Diagnostic Register (PUDR)—Ethernet Bits

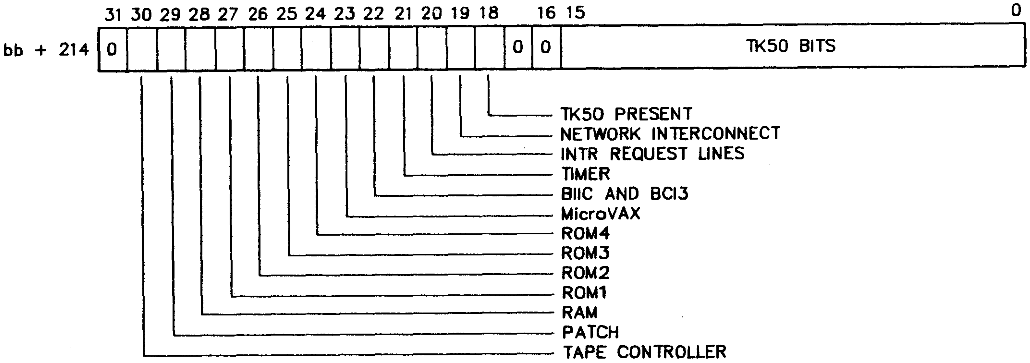


Figure B-7 Power-Up Diagnostic Register (PUDR)—TK50 Bits

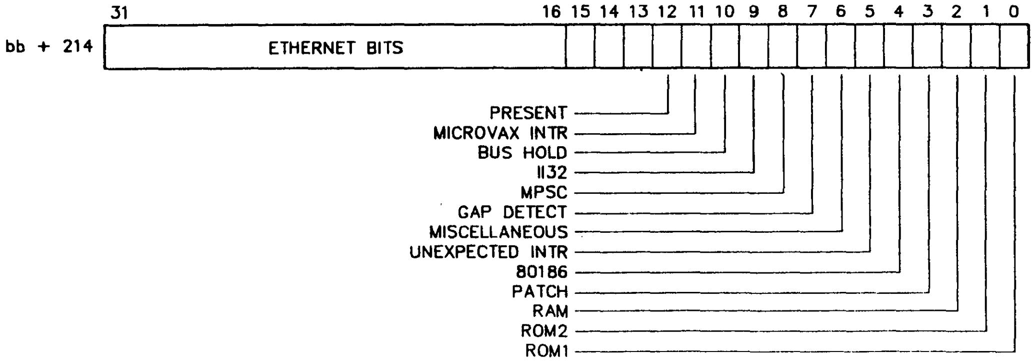


Table B-4 PUDR Bit Definitions

Bit	Mnemonic	Name	Description
31	MBZ		Bit must be zero.
30	TAP CTRL	Tape Controller	Sets when the TK50 and associated logic are good.
29	PATCH		Sets when the patch logic is good.
28	RAM	Random Access Memory	Sets when the DEBNx's on-board RAM passes self-test.
27 26 25 24	ROM1 ROM2 ROM3 ROM4	ROM chips 1-4	Each bit sets when the corresponding MicroVAX ROM checksum is good.
23	UVAX	MicroVAX	Sets when the MicroVAX passes self-test.
22	BI	BIIC and BCI3	Sets when the BIIC and BCI3 chips pass self-test.
21	TMR	Timer	Sets when the interval timer is good.
20	IRQ	INTR Request Lines	Sets when no IRQ line is stuck asserted.
19	NI	Network Interconnect	Sets when the Ethernet controller, including the LANCE chip, is good.
18	TK50 PR	TK50 Present	Sets when a TK50 tape drive is present.
17:13	MBZ		Bits must be zero.
12	PRES	Present	Sets when a TK50 tape drive is present.
11	UVINT	MicroVAX INTR	Sets when the MicroVAX-to-80186 interrupt logic is good.
10	BUSHD	Bus Hold	Sets when there are no bus hold errors.
9	II32		Sets when the II32 transceivers are good.
8	MPSC	MPSC chip	Sets when the MPSC logic is good.
7	GAP	Gap Detect	Sets when the gap-detect logic is good.
6	MISC	Miscellaneous	Sets when miscellaneous registers are good.
5	UNEXP	Unexpected INTR	Sets if an unexpected INTR is trapped.
4	80186	80186 Processor	Sets when the 80186 is good.
3	PATCH		Sets when the patch logic is good.
2	RAM		Sets when 80186 RAM is good.
1	ROM2		Sets when 80186 ROM2 is good.
0	ROM1		Sets when 80186 ROM1 is good.

C Bootstrapping with DEBNx

Most host systems in which the DEBNx resides can bootstrap their operating systems either locally (from disk) or remotely (from an Ethernet network).

Host systems can boot voluntarily (from a command that a user types at the system console) or involuntarily (from a command to boot that arrives via the network).

Whether voluntarily or involuntarily, the host system boots from a specified device. The device is usually specified by the user in a console command, and the booting hardware/software then reads this parameter.

The DEBNx plays a role in the booting whenever (a) a command to boot arrives over the network, or (b) the specified device from which to boot is the network.

The sections in this appendix describe these two bootstrap roles:

1. A remote system sends the DEBNx a command to boot—that the DEBNx's host system should reboot itself.
2. A local user on the DEBNx's host system commands the host system to boot itself, and this reboot takes place via the DEBNx over the network.

C.1 REMOTE SYSTEM SENDS DEBNx A COMMAND TO BOOT

This operation happens typically on networks where all systems are restarted and booted from a central system.

A MOP Boot message arrives at the DEBNx from the network:

1. If the boot-enable jumper is NOT in place, the DEBNx will pass the MOP Boot message to the port driver, if the MOP protocol type has been enabled.
2. If the boot jumper IS in place, the DEBNx asserts BI RESET, assuming the driver has enabled reset. The power-up default condition is to allow the DEBNx to assert BI RESET.

C.2 LOCAL USER REQUESTS A BOOT FROM THE NETWORK

A user on the DEBNx's host system can request a boot by using the system console and typing the console B (boot) command at the system prompt (> > >).

The B command has the following syntax:

```
B [/R5:<nnnn>] [<ddxy>]
```

where:

B — is the boot command.

/R5: — specifies GPR 5 (R5).

nnnn — are four hexadecimal digits that the B command can pass to the bootstrap program via R5.

ddxy — specifies the device from which to bootstrap.

- dd — is the code for the device type.
- x — is the DEBNx's VAXBI node ID.
- y — is the unit number of the Ethernet adapter (always zero).

The syntax varies slightly from system to system depending on the type of processor that controls the system's console.

Example (on a VAX 8250):

```
>>> B ET50
```

D Ethernet Address

The LANCE station address PROM contains the DEBNx controller's unique Ethernet address plus a fixed test pattern. The PROM provides 32 bytes of space for the address. When the PROM is accessed, it outputs 8 bits, which are stored as the low-order byte of a data word in MicroVAX buffer RAM.

On power-up or reset, firmware running on the MicroVAX writes the Ethernet address from the PROM into the Physical Address Register in the LANCE/processor memory interface. The Ethernet address is also copied into the DEBNx nodespace and can be examined from the system console. Before an operating system is booted, the hardware Ethernet address is the default DECnet address. The Ethernet address is used when a system is downline loaded.

D.1 HOW TO READ THE HARDWARE ETHERNET ADDRESS

If VMS is not running, use the console Examine command to read the Ethernet address. The address is stored in the six bytes:

```
bb + 218
bb + 219
bb + 21A
bb + 21B
bb + 21C
bb + 21D
```

The base address (bb) is computed (in hex) as follows:

```
2000 0000 + (2000) * (DEBNx VAXBI node number)
```

For example, if the DEBNx is at VAXBI node B, then 2000 * B is 16000 in hex (see Table D-1). The base address bb is 2000 0000 + 1 6000, or 2001 6000. You can find the hardware address by examining bytes 2001 6218 through 2001 621D.

If VMS is running, use the NCP SHOW LINE CHARACTERISTICS command, which displays the Ethernet address as Hardware Address. For example, the display from SHO LINE ETA-0 CHAR might include a line like the following:

```
Hardware Address 08-00-2B-00-10-5E
```

In DEBNx nodespace 08 is the byte in bb + 218, and so on.

Table D-1 Node Number Multiple for Ethernet Address Computation

VAXBI Node Number (hex)	2000 * VAXBI Node Number
0	0000
1	2000
2	4000
3	6000
4	8000
5	A000
6	C000
7	E000
8	10000
9	12000
A	14000
B	16000
C	18000
D	1A000
E	1C000
F	1E000

Index