TC2000™
Diagnostic Guide

March 1990

Revision: Preliminary 2.0

Part No. A330036G10
Document Rev: A

BBN Advanced Computers Inc.

RESTRICTED RIGHTS LEGEND

BBN Advanced Computers Inc
10 Fawcett St.
Cambridge MA 02138

RELEASE LEVEL

This manual conforms to the Released Version of the nX™ operating system software, released in March of 1990, for the TC2000™ multiprocessor.

NOTICE

BBN ACI has prepared this manual for the exclusive use of BBN customers, personnel, and licensees. The information in this manual is subject to change without notice, and should not be construed as a commitment by BBN ACI. BBN ACI assumes no responsibility for any errors that appear in this document.

TRADEMARKS

BBN ACI thanks the following contributors for their efforts in developing this manual:

Authors:
T. Calderwood
M. Lefebvre
C. Macfarlane
R. Preston

Key Contributors:
T. Blackadar
R. Clements
J. McLeish
W. Rowe

Contents

Chapter **Page**

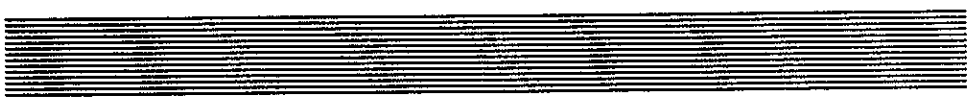
Chapter **Page**

Chapter **Page**

Chapter **Page**

Chapter **Page**

List of Figures



How to Use This Manual

Purpose of the Manual

The manual describes the TC2000 diagnostics and the Test and Control System and how to operate them.

Revision History

This a first version of the manual; it supports the 2.0 release the nX Operating System for the TC2000 multiprocessor.

This manual also supports the diagnostic software with the following versions:

Diagnostic	Version	Date
TEX.EXE	4.003	Nov 22 1989?
B2SWITCH.EXE	2.6	Aug 12 1989
B2CLK.EXE	2.6	Aug 12 1989
B2VME.88	2.106	Apr 21 1989
B2VME2.88	2.189	Nov 13 1989
TCFPV.88	2.737	Jan 17 1990
STP.88	6.400	Nov 22 1989
VMEDIAG.88	1.346	Jan 12 1990
B2MDM.EXE	1.200	Aug 12 1989
DISKTOOL.88	1.003	Dec 8 1989
periph_diag	1.078	Jan 11 1990
stress	1.015	Aug 20 1989

Other Places to Find Answers

If you experience any problems with our product, or if you have questions or suggestions, please do one of the following:

- Send electronic mail from anywhere on the Internet to:

  aci-questions@bbn.com

- Send mail to:

  ACI Bugs
  BBN Advanced Computers Inc.
  10 Fawcett St.
  Cambridge, MA 02138

- If you are under warranty, or have a software maintenance contract, you can also call our hotline number:

  1-800-4AC-BFLY (1-800-422-2359) in the United States
  1-617-873-8660 from any other location

If you are reporting a problem, please include as much information as you can, as follows:

- The operating system **version** and multiprocessor **model name**
- The **size** of your multiprocessor (number of function cards and amount of memory)
- The **number of nodes** that were in the cluster when the problem occurred (if relevant)
- The **total number of people** using the system when the problem occurred
- An **example** that illustrates the problem
- A **record** of the sequence of events that led to the problem; especially a stack backtrace (see the system administration guide)

We are also interested in your evaluation of our documentation. We would appreciate it if you would fill out the form at the back of this manual and return it to us.

Audience Level

This manual is written for system administrators, field service engineers, and technicians.

Other References

For more information on using the nX operating system, refer to *Getting Started with nX O.S., TC2000 System Administration Guide* or the *nX System Software Installation Guide*. For more information on the hardware, refer to the *TC2000 Maintenance Manual* and *Inside the TC2000*.

Organization

This manual is organized into seven chapters. The first three describe the diagnostics and the TCS and how to use them. The last four describe the four groups of diagnostic tests.

- Chapter 1 is a brief introduction
- Chapter 2 describes the TCS and how to use it
- Chapter 3 shows how to run the diagnostics
- Chapter 4 describes the processor diagnostics and how to use them
- Chapter 5 describes the Switch Diagnostics and how to use them
- Chapter 6 describes the Sinister Test Program (STP) and how to use it
- Chapter 7 describes the standalone Peripheral Diagnostics and how to use them
- Chapter 8 describes the nX Peripheral Diagnostics and how to use them
- Chapter 9 describes the Stress Test and how to use it
- Chapter 10 describes basic usage of Disktool
- Chapter 11 describes the Modem Diagnostic and how to use it
- Appendix A contains lists of error codes for the various I/O subsystem controllers and devices

Typographic Conventions

This manual uses the following conventions to present information:

bold	Text in **bold** indicates an exact filename, a command, or user input.
italics	Text in *italics* indicates a variable, or a value that the user supplies; for example, *filename* stands for the file under discussion.
`type`	Text in `typewriter` font represents computer output.
bold italics	Text in ***bold italics*** indicates an emphasized word or phrase.

<Delete>	Names enclosed in angle brackets indicate keyboard keys; for example, <Delete>, <Esc>, and <Return>.
<Control-Z>	Two key names enclosed in angle brackets indicate that you should press the keys simultaneously; for example, <Control-Z> means that you should hold down the Control key and press the Z key.<>
<Esc>Z	A single key name enclosed in angle brackets followed immediately by another key name indicates that you should press the first key and ***then*** the second; for example, <Esc>Z means that you press the Escape key and ***then*** press the Z key.
↵	This symbol represents the <Return> key in computer dialog examples.
[ ]	In command syntax descriptions, square brackets enclose optional items.
. . .	A horizontal ellipsis indicates a repetition of the previous command or input string.
. . .	A vertical ellipsis indicates that irrelevant portions of a program have been omitted.

1

Introduction

This guide describes the characteristics and operation of the software that performs diagnostics on the hardware components of the TC2000 multiprocessor. It also provides an overview of the Test and Control System (TCS).

1.1 System Diagnostics

There are five categories of TC2000 diagnostics:

- processor exclusive: B2VME, B2VME2, and TCFPV
- processor to switch: B2SWITCH
- memory and system-level harware: the STP and **stress**
- standalone VMEbus peripheral: VMEDIAG and DISKTOOL
- system peripheral: **periph_diag**

Each of these diagnostics is described in its own chapter (4–9).

1.1.1 Processor Diagnostics

The processor diagnostics (B2VME, B2VME2, and TCFPV) test:

- Processor memory
- Processor switch interface gate array (SIGA)
- Processor registers

1.1.2 Processor to Switch Diagnostics (B2SWITCH)

The processing and switching subsystems tested are:

- Processor boards (B2VME, TC/FPV)
- Butterfly switch (TC/SR, TC/SS)

1.1.3 STP Diagnostic (STP)

STP tests the memory partitions on each processor and the capability of the processor nodes to talk to each other over the switch.

1.1.4 Standalone Peripheral Diagnostics (VMEDIAG)

The peripheral subsystems tested are:

- $^1/_4$ Inch SCSI Tape Subsystem
  - Interphase 4210 SCSI controller (Jaguar board)
  - B2T125 (Tandberg TDC 3640 $^1/_4$ inch cartridge tape drive)
- $^1/_2$ Inch SCSI Tape Subsystem
  - Interphase 4210 SCSI controller (Jaguar board)
  - TC/T6250-2 (Cipher M990 GCR CacheTape® $^1/_2$ inch tape drive)
- Terminal Controller Subsystem
  - TC/TERM-1 or -2 (Systech HPS 6840 host adapter)
  - TC/CC16 (Systech HPS 7088 remote cluster controllers)
- Ethernet Interface Subsystem
  - TC/ETH (Excelan 301 Ethernet controller)
- SMD Fixed Disk Subsystem
  - TC/DC (Xylogics 451 disk controller)
  - TC/DSM81 (Northern Telecom NT 8414 disk drive)

1.1.5 System Peripheral Diagnostics (periph_diag)

These diagnostics run under the nX operating system and exercise the major functional operations of each peripheral device, such as reading and writing blocks of data to tapes and disks, and erasing and rewinding tapes.

1.1.6 SCSI Disk Exercising (DISKTOOL)

This utility program can be used in troubleshooting a SCSI disk problem. There are two versions one running under TEX and another under the nX OS.

1.2 Diagnostic Interface

The TC2000 diagnostics are part of the TCS. The TCS consists of its own power supply, a separate processor running DOS, and the master program called the TCS Executive (TEX). The TCS acts as the local TC2000 system console. An operator issues commands and makes menu selections to execute the diagnostics. See Chapter 2 for information on the TCS; see Chapter 3 for instructions on running the diagnostics.

1.3 Definition of Terms

Some definition of terminology used in this guide is necessary:

- Each line on a menu is a menu ***item***.
- ***Highlighting*** a menu item refers to using the arrow keys and moving the cursor up or down to a particular menu item which is then displayed in reverse video.
- Menu items are selected by highlighting the menu item and pressing <Return>. The menu can also be used in "command mode", where the menu item can be typed in as a command.

2

Test and Control System Overview

2.1 Introduction

This chapter is an overview of the Test and Control System (TCS). It describes software usage, TCS executive functions, and available commands and menus.

The TCS is a separate subsystem consisting of its own power supply, processor, and software. The TCS can control the hardware and run programs on the TC2000. The TCS software consists of an operating system (DOS), the master program, and the diagnostics. The master program, called the TCS Executive (TEX), is the main TCS software with which the user will interact.

This chapter is an overview of the TCS. It provides all the information you need to run diagnostics, but not all the information on using the TCS. For detailed information on the TCS, see the system administration guide.

2.2 TCS Executive (TEX)

TEX provides power on/off, reset, bootstrap, memory examine/deposit and TTY simulation functions, as well as direct access to the low-level TCS slave functions. It also contains routines to scan the system configuration and to report that information to the operating system software.

TEX provides a "console terminal" interface for the operating system running on the TC2000. Characters are passed between the TCS's console and the TC2000. TEX also provides a number of services to the TCS operating system at its request. These allow the TCS and the nX operating systems to cooperate in controlling the halting, reloading, reconfiguring and monitoring of the TC2000 hardware. In addition, TEX contains diagnostic routines for testing a processor.

TEX can be loaded from a hard or a floppy disk, running under DOS. The installation procedure is described in the *TC2000 System Software Installation*

Guide. TEX takes commands from either the system console or through the serial port and modem interface. It will execute a script of commands from a file if that file name is included on the DOS command line that starts TEX. Additionally, TEX will execute a script file by using the DOS input redirection mechanism.

2.3 Using the TCS

The TCS can be in one of four states:

- serving the operating system (with operating system prompt displayed)
- DOS (with the DOS prompt displayed)
- TEX (with the TEX prompt displayed)
- running diagnostics (with diagnostic menus or screens displayed) or application programs such as **sacopy** or **boot**

CAUTION

While the operating system is running, you can get to the TCS by typing back-quote period `.. However, using TEX while still running the nX operating system can destroy the file system if you use any TEX commands that interfere with the nX operating system. It is advised that, when possible, you use **/etc/shutdown** to warn other system users that you are shutting down the system. Then use **/etc/halt** to synchronize the disks and halt the operating system. You will then be in TEX, from which you can perform necessary operations, such as checking or reconfiguring the hardware. You can reboot the system from TEX.

At boot-up, the TCS displays its top-level menu:

```
Menu:  TEX
    1. Quit-to-Dos
    2. Help
    3. Application/                    System mode is application
    4. Boot/
    5. Configuration/
    6. Hardware/
    7. Log/
    8. Memory/
    9. Slave/
    10. Terminal-Type/                  Hardcopy, Menu-On
    11. Utility/
        TC2000   TEX [7.7.7] version x.x.x
                                        where 7.7.7 is the address of the master node
TEX ->
```

2.3.1 Using the TEX Menu System

TEX is a hierarchical menu-driven system, with a main menu, labelled as the TEX Menu. It accepts "words" on command lines, where each word can pop up the next level of the menu, prompting you for the next input.

Menu items can be selected by:

1. Typing a menu item number and <Return>,
2. Typing enough of the menu item to distinguish it from the other menu items and <Return>,
3. Highlighting the menu item with the arrow keys and pressing <Return>.

TEX is at the main menu when the top line is: `Menu: TEX`.

Because of the nested-level nature of the TEX menu package some commands will be executed, and others at the same level will have another menu pop up. If you are familiar with the sub-menus, you can also type a string of commands without being prompted by subsequent menus. The following two examples show how you can quicken your use of the TEX menu system.

Both examples describe how to change a terminal type to a VT320. The first example takes you through each menu. You always type commands at the end of the menu prompt (->).

Figure 2-1 Terminal-Type Selection with menus

`TEX -> t` ↲

Typing **t** at the main menu for the **Terminal-Type** menu brings up the following menu:

```
Menu:  TEX Terminal-Type
1. Quit
2. VT320
3. VT100
4. Sun
5. X
6. PC
7. Hardcopy
8. Menu
9. More
```

`Terminal-Type -> vt3` ↲

Typing **vt3** as the unique identifier for **vt320**.

The system then returns to the Main menu with a line that the terminal is now set to a VT320 terminal setup.

```
Menu:  TEX
    1. Quit-to-Dos
    2. Help
    3. Application/                        System mode is application
    4. Boot/
    5. Configuration/
    6. Hardware/
    7. Log/
    8. Memory/
    9. Slave/
    10. Terminal-Type/                      VT320, Menu-On
    11. Utility/
          TC2000   TEX [7.7.7] version x.x.x
TEX ->
```

The second example shows that the TEX menu system also allows you to type all commands at one menu, without being prompted through the menu system. For example, if you know that you are changing your terminal to a VT320 configuration, and you know the required menu commands, you could type everything on one line at the TEX-> menu prompt. For example:

Figure 2–2 Terminal–Type Selection by Command

```
TEX -> t vt32
```

Typing **t** at the main menu for the **Terminal–Type** menu and **vt32** as the minimum amount of letters for **vt320**. will change the terminal type. This returns the following menu automatically:

```
Menu:  TEX
    1. Quit-to-Dos
    2. Help
    3. Application/                        System mode is application
    4. Boot/
    5. Configuration/
    6. Hardware/
    7. Log/
    8. Memory/
    9. Slave/
    10. Terminal-Type/                      VT320, Menu-On
    11. Utility/
          TC2000   TEX [7.7.7] version x.x.x
TEX ->
```

Becoming comfortable with the TEX menu system so that you can type most of the commands at one menu prompt is a technique well worth mastering.

Menu Command Line Editor

The menu package also includes a full command–line editor and history mechanism. The history buffer allows you to to repeat a command or correct a mistyped command. Note that the use of ! has nothing to do with the history mechanism and the control sequences are **not** case sensitive. Figure 2–3 contains a list of these commands.

Figure 2–3 TEX Menu Line Editor

<Control–M> or <Control–J>	Execute command–line or menu–item
<Control–L>	Redisplay
<Control–N>	Get next command–line entry
<Control–P>	Get previous command–line entry
<Control–G>	Abort command
<Control–K>	Kill to end of line from cursor
<Control–U>	Undo the whole line
<Control–A>	Go to beginning of line
<Control–E>	Go to end of line
<Control–F>	Go forward one character
<Control–B>	Go back one character
<Control–D>	Delete this character (under cursor)
<Control–H>	Delete the previous character
<Control–I>	Insert <Tab>
<Control–X>	Execute a menu item
<Escape> b	Backward one word
<Escape> f	Forward one word
<Escape> n	Next menu item
<Escape> p	Previous menu item
up–arrow	Previous menu–item
down–arrow	Next menu–item
left–arrow	Go to previous character
right–arrow	Go to next character
vt220/vt320 "do key"	Execute a menu–item
< *filename*	Invokes command re–direction
!*command–name*	Invokes looping until error
!!*command–name*	Invokes looping regardless of errors

Note that looping may be stopped by hitting any key.

2.3.2 Moving Around in the TCS

You can move easily between DOS, TEX, the diagnostics software, the kernel debugger **kdb**, and the operating system once the TCS is running. You will see relevant screen displays according to the following:

- When the TCS is serving the nX operating system, the login prompt, shell prompt, or an application (if running) will be displayed.
- When in DOS, the prompt is `C:` followed by the directory and a >. For example, `C:\TCS>`.
- When in TEX at the main menu level, the prompt is `TEX->`. The prompt is shown with a TEX menu, if the menu display option is ON. The menu system is described in Section 2.3.
- When using TCS for diagnostics, the screen will display a menu or test results, depending on the diagnostic.

Using DOS

When DOS is running, the console will display the DOS prompt:

`C:\`*directory*>

where *directory* is the name of the DOS directory. In the root directory, the prompt is: `C:\>`

To get to the TEX prompt, type:

`C:\>` **CD \TCS** ↵
`C:\TCS >` **TEX** ↵ The TCS banner and prompt will be displayed:

```
Menu:  TEX
    1. Quit-to-Dos
    2. Help
    3. Application/                    System mode is application
    4. Boot/
    5. Configuration/
    6. Hardware/
    7. Log/
    8. Memory/
    9. Slave/
    10. Terminal-Type/                  VT320, Menu-On
    11. Utility/
              TC2000   TEX [7.7.7] version x.x.x
TEX ->
```

From this point you could boot the system, by using the appropriate TEX command as described in section 2.3.4.

Using TEX

When TEX is running, the console will display the current TEX menu:

```
Menu:  TEX
    1. Quit-to-Dos
    2. Help
    3. Application/                    System mode is application
    4. Boot/
    5. Configuration/
    6. Hardware/
    7. Log/
    8. Memory/
    9. Slave/
    10. Terminal-Type/                  VT320, Menu-On
    11. Utility/
              TC2000   TEX [7.7.7] version x.x.x
TEX ->
```

To return to DOS, type **q**, **quit**, or **8** at the menu system prompt.

If you reached this TEX menu by typing **'.** at the nX console, you can return to the nX operating system (if it is still running) by typing **Application Tty** or **a t**.

To boot the operating system if it is no longer running, follow the steps shown in Section 2.3.4. To run diagnostics follow the steps in Chapter 3.

Entering the kernel debugger (kdb)

kdb is the kernel debugger that analyzes the kernel of the nX operating system. You can access **kdb** from TEX as well as from the nX operating system. You can only access **kdb** if the following are true:

- **kdb** was initialized at bootup time
- you are using the console terminal from which the machine was booted

If the **kdb** option was not selected during bootup, you will not be able to access its kernel debugging capabilities later in that session. The bootup line which enables **kdb** is:

```
Start KDB [y]? y
```

You must specify **y** or a <Return> (uses default) to enable **kdb**. Once you have enabled **kdb** and the kernel is finished booting, you can enter **kdb** through TEX by typing **'.** to get into TEX and then typing the appropriate commands:

`TEX` -> **hard nmi** ↲	Sets the NMI flag allowing entry to **kdb**
`TEX` -> **app run** ↲	Starts the "application"

To exit the **kdb** session, type:

:c ↲

to allow the kernel to continue from where it left off. For further information on **kdb**, refer to *System Administration* Appendix E, "Using **kdb**".

2.3.3 Booting the nX Operating System from DOS

NOTE

Do not attempt to boot the nX operating system when it is already running. Shut down the operating system with **/etc/shutdown** and **/etc/halt** first, then boot up when you wish. If the operating system is already running, type **Application Tty** from the TEX prompt to resume it.

To boot the operating system in a system with autoboot disabled, follow the steps shown in Figure 2–4. Using autoboot you can setup the system to boot directly on power-up or to boot without operator intervention as soon as "start" is typed. See the description of **BOOTCFG.TCS** in section 2.4.4 for more details on configuring the system.

Figure 2–4 Booting the nX Operating System from DOS

`C: \TCS >` **TEX –AUTO** ↲

or you can type

`C: \TCS >` **START** ↲

(miscellaneous messages as system boots and loads the nX software)

`Boot:` **(xy0,0,0)** ↲	This causes the nX operating system to boot from Xylogics disk controller 0, disk drive 0, partition 0 (root partition).
`Specify Boothowto flags[y]?` **n** or **y**	Typing **y** causes the system to prompt for single/multi user boot up, and asks for **kdb** startup (see following lines). Typing **n** causes rebooting to begin without the next three queries. Defaults are in brackets: [].

`Come up Single User[y]?` **y** or **n**	Type **y** for single user, **n** for multiuser.
`Start KDB[y]?` **y** or **n**	Type **y** to enable **kdb** for possible later use, type **n** to disable **kdb**.
`Read Machine Configuration [y]?` **y** or **n**	Type **y** to read machine configuration, type **n** to skip.

2.3.4 Booting the nX Operating System from TEX

NOTE

Do not attempt to boot the nX operating system when it is already running. Shut down the operating system with **/etc/shutdown** and **/etc/halt** first, then boot up when you wish.

Before booting the operating system, be sure that you are at the top–level of the TEX menu, then follow the steps shown in Figure 2–5.

Figure 2–5 Booting the nX Operating System from TEX

`TEX->` **c** ↲	From the Main menu, this moves to the **Configuration** menu.
`Configuration->` **f** ↲	Clears the last configuration (**Forget**).
`Configuration->` **r** ↲	Reread **BOOTCFG.TCS** and **SLOTCFG.TCS**.
`Configuration->` **q** ↲	Quit back to the Main menu.
`TEX->` **a r** ↲	Enters the Application menu and runs in automatic, which in this case boots the system as though **tex -auto** or **start** had been used, at the DOS prompt.

2.4 Using TEX

You can enter into the TEX program at any time if you are at the TC2000 console or you are connected to the TCS serial port (possibly through a remote system). To enter the TEX program from DOS, type **tex** (see section 2.3.2). Once in the TEX program, you can enter TEX commands directly at the keyboard, or you can create a DOS file of commands to be accessed from TEX (such as configuration files). TEX accepts some command–line switches which can be typed in full or uniquely abbreviated (*e.g.*, **tex –auto**). The currently available switches are:

-autoboot This causes TEX automatically to read the bootstrap configuration file **BOOTCFG.TCS** and the slot configuration file **SLOTCFG.TCS** before executing any other commands. These files configure TEX for a particular installation hardware and operational characteristics. Refer to Section 2.4.4 for descriptions of **SLOTCFG.TCS** and **BOOTCFG.TCS**. TEX then begins to operate the system as described for the **Application Run** command.

NOTE

Entering the DOS command **start** will boot the same way as **tex -autoboot**.

In addition to **-autoboot**, there are four additional "boot" switches which modify the actions of **-autoboot** for specific situations. They act similarly to **-autoboot** but act as though different settings of the **BOOTCFG.TCS** file are active, thus modifying the bootstrap process.

-MultiUser causes TEX to set the bootstrap variables so that the nX bootstrap, **BOOT.88**, brings up the nX OS in multi-user mode.

-SingleUser causes TEX to set the bootstrap variables so that the nX bootstrap, **BOOT.88**, brings up the nX OS in single-user mode.

-PromptedBoot causes TEX to set the bootstrap variables so that the nX bootstrap, **BOOT.88**, prompts the console user for answers to its startup questions rather than using the values in the **BOOTCFG.TCS** file.

-Uniprocessor is exactly like -PromptedBoot except that TEX marks all processors other than the master node as "disabled". This is for faster booting of single processor diagnostics.

There are also these switches:

-TCS-Addr *hex address*
This sets the default slave address. This is equivalent to the **Utility Slave** command at the TEX command level, but is provided for use in batch files.

-help This switch displays the list of command line switches.

filename TEX also accepts a file name on the same command line. This causes TEX to execute the commands contained in *filename*.

At the TEX prompt, you can type any TCS command described in this section. Many TEX commands take one or more arguments. If these arguments are card addresses, they need to follow a predefined format, as described in section 2.4.3. Refer to section 2.3.1, "Using the TEX Menu System" for a description of how to use the menu system to enter and edit commands.

2.4.1 TEX Commands

The following is a list of menu items that may also be entered as commands at the TEX prompt with associated arguments. You may need these commands in the course of testing the system. Section 2.4.2 contains a summary list of these commands.

? Displays the list of TEX commands.

Application Configuration

Writes configuration information for the system into the memory of the current slave (which should be a processor, not a switch or clock card). The information must have been previously obtained, either by a **Configuration Scan** or **Utility Preview** command or automatically as a result of a **Boot** or **Application Run** command. This information is written in a predefined format in the processor's memory. The serial number of the current slave is also read from its TCS EEPROM and stored in the processor's memory.

In automatic operation, the configuration information is written to each node after the microboot successfully completes and again before the application bootstrap starts.

Application Examine-Card

Displays information about the selected TC2000 card or cards. The card's type, serial number and revision levels are displayed (from the EEPROM on the card). The current use, goal and state information are also shown.

Application File-Select a|p|u *filename*

Specifies a *filename* to be used by **Application Run**. *filename* must exist on the TCS's DOS file system. This file is usually named **BOOT.88** for bootstrapping the operating system, but may be a specific diagnostic or other program.

a specifies that *filename* will be used as an application (or its bootstrap). *filename* here is usually **BOOT.88**.

p specifies that *filename* will be used as the POST (Power-On Self Test). *filename* here is usually **POST.88**.

u specifies that *filename* is to be loaded into TEX. It will later be used as microboot. *filename* here is usually **UBOOT.88**.

Application King-Node *card_address*

Specifies the *card_address* of the king (master) node.

Application Mode idle|application|boot|off

Sets the "system mode" that directs TEX's operation under the **Application Run** command.

idle directs TEX to discover cards in the system, and to turn on their power and initialize them, but not load application programs. TEX will run the microboot and POST programs on processor nodes.

application directs TEX to first initialize the system as for **idle**. It then loads and runs the application (or its bootstrap) as specified in the **Application File–Select a** command into the current king node as specified by the **Application King–Node** command. The bootstrap and king node are also specified in the **BOOTCFG.TCS** file (see **reconfigure**).

boot directs TEX to halt all processors and then restart the initialization and bootstrap procedures as described in the **application** option.

off directs TEX to halt all processors and then turn off power to all processors, switches, and clocks.

Application Run

Places TEX in automatic operation as selected by the **Application Mode** command. It performs such operations as initialization, running microboot, and loading applications into the king node. Once the king node is operational, TEX acts as the console terminal for it.

If TEX is running an application, typing **'.** returns you to the TEX prompt.

Application System–Status

Displays the state of the clock cards and any switch and processor cards.

The states will depend on the current system mode and on whether the cards have successfully been initialized and whether their bootstraps and POST have run.

Application TTY

This command enters the console TTY protocol routine. Characters are passed to and from the TC2000 processor at the default address over the TCS bus, using ring buffers in the processor's memory. The escape sequence to return to the TEX prompt, is backquote period **'.**. (Backquote is generally an unshifted tilde, so this is similar to an escape in **tip**, but different, so that TEX can be run from a **tip** line.)

TEX Boot auto|multi–user|prompt|single–user|uniprocessor

The other selections in the Boot menu are variations of this operation, to override this default action in various ways.

The order of this is as follows:

TEX scans the machine for all existing clock, switch and processor cards. It reads the **BOOTCFG.TCS**, **SLOTCFG.TCS**

and **DRIVERCF.TCS** files. It reports any inconsistencies between the **SLOTCFG.TCS** file and the actual machine.

It checks to see whether all cards are on and running, and if so it goes directly to servicing the king node. Otherwise, TEX initializes all cards in the machine (except those disabled by the **BOOTCFG.TCS** file). It then runs the micro-boot program and the POST program on each node, supplying them with configuration information as required.

Finally it reports configuration and test results to the king node and loads and runs the application (or, usually, its bootstrap, **BOOT.88**). The bootstrap variables and bootstrap file string are supplied to the king node from the **BOOTCFG.TCS** file to control the bootstrap operation.

automatic causes TEX to bring up the TC2000 computer according to the configuration information in the file **BOOTCFG.TCS** in the current directory.

multi-user causes TEX to bring up the TC2000 computer as described for **automatic**, but with the following changes:

After reading the BOOTCFG.TCS file, TEX forces the bootstrap variable settings to those which cause BOOT.88 to boot the operating system in multi-user mode, without asking any questions at the console terminal.

prompt causes TEX to bring up the TC2000 computer as described for **automatic**, but with the following changes:

After reading the BOOTCFG.TCS file, TEX forces the bootstrap variable settings to those which cause BOOT.88 to prompt the operator for mode settings from the console terminal. The operator can specify the bootstrap file string and the other choices provided by BOOT.88 such as single- versus multi-user operation.

single-user causes TEX to bring up the TC2000 computer as described for **multi-user**, except that a bootstrap parameter is set to force the operating system into single-user operation.

uniprocessor causes TEX to bring up the TC2000 computer as described for **prompt**, except that TEX disables all processors except for the king node. This is used to speed the booting process when only one processor is to be used. This is typically used to load a diagnostic program for an I/O device that is accessed from just one node.

Configuration Card-use off|isolate|system|auxiliary|forget *card_address*
Sets the configuration state for one or more cards.

If **off**, TEX will treat the card as *disabled* and, in addition, will leave its power turned off.

If **isolate**, TEX turns on the card, but leaves its switch port turned off so that it cannot interact with other cards in the system. This is intended to allow power–on servicing of TC2000 cards.

If **system**, TEX makes the card available for normal operational use any disabling done by **BOOTCFG.TCS**, or a previous **card–use isolate**, is removed.

If **auxiliary**, TEX will use the card normally (**system**), until TEX receives a service request to use the card for special application such as nodes running under pSOS.

If **forget**, TEX will not make use of the card and will forget that it was in **SLOTCFG.TCS**.

Configuration Forget

Clears the configuration found by the **Configuration Scan** and **Application Run** commands, so that it will be rediscovered by the next **Application Run** command.

NOTE

Under normal use of **Configuration Forget**, all nodes should be halted or powered down before rebooting the operating system. Otherwise, running nodes may interfere with the bootstrap process.

Configuration Reconfigure

Makes TEX reread the two configuration files **BOOTCFG.TCS** and **SLOTCFG.TCS**. **SLOTCFG.TCS** contains a record of the cards found within the machine when the system was last configured explicitly scanned and written. **BOOTCFG.TCS** contains bootstrap parameters for the boot process. See section 2.4.4 for more information on these configuration files.

Configuration Scan file | small | medium | large

Causes TEX to contact all TCS slaves for the configuration parameter that is specified and record and display what it finds.

If **file** is specified, all cards mentioned in the **SLOTCFG.TCS** file are scanned. If **large** is specified, all eight midplanes in all eight bays are scanned. If **medium** is specified, all eight midplanes in bay 7 are scanned. If **small** is specified, only midplane 7 of bay 7 is scanned.

As the scan is done, the card type, serial number, and revision levels are displayed. The card types are also saved in the TCS for further use of the **Application Configuration** command. See also the **Utility Preview** command.

Configuration Terminal–Driver Show

Displays the current settings of configuration parameters related to the serial terminal driver which communicates with the console port and the TCS's dial–up modem.

The parameters displayed are: Version number, Terminal XON/XOFF flow control, dialup modem access, Initialization string, and dialup port active.

Configuration Terminal–Driver Dialup–Access

Sets the parameter which allows or disallows access via the dialup modem port. With this parameter enabled, dialup access is available.

Configuration Terminal–Driver Dialup–Password

Sets the password required for access via the dialup modem. This password can also be set by the configuration file DRIVERCF.TCS.

Configuration Terminal–Driver Modem–Init–String

Sets the initialization string which is sent to the dialup modem at startup and between dialup calls. This string can also be set by the configuration file DRIVERCF.TCS.

Configuration Terminal–Driver XON/XOFF

Sets the parameter which enables or disables use of XON/XOFF flow control on the console serial port. This parameter can also be set by the configuration file DRIVERCF.TCS.

Configuration Write Boot–Configuration *filename*

Creates a bootstrap–time configuration file. Normally, **BOOTCFG.TCS** is used, but a different file name can be specified to create other copies or versions of the configuration. The contents of the **BOOTCFG.TCS** file are described in section 2.4.4. This command is typically used as follows:

Read in the existing **BOOTCFG.TCS** file by **Configuration Forget** followed by **Configuration Reconfigure**. Change a configuration item, such as **Configuration Card Use OFF** or **Configuration Set AutoBoot Yes**. Finally, write the new **BOOTCFG.TCS** file using this command.

Configuration Write Driver–Configuration

Creates a configuration file for the parameters which control the serial device driver. The file used by TEX is DRIVERCF.TCS, but a different file name can be used. Typical use would look similar to **Boot–Configuration** above.

Configuration Write Slot–Configuration *filename*

Creates a slot configuration file. The default *filename* is **SLOTCFG.TCS**, but can be a different name as well. If you enter a *filename*, TEX automatically places this information into *filename*.

Hardware Initialize *card–address*

Performs a card–specific initialization sequence for a processor, switch, or clock card. *card_address* may have wildcards.

For the processor, the sequence reads and displays the card status, turns on the power, asserts processor and card resets, and turns on the LED blinker. It then waits a while and removes the card reset. The processor remains halted. Then it checks and reports the card status register again.

For the switch cards, the sequence is similar, except that the switch gate arrays are initialized and all eight function card ports are enabled.

For the clock card, the card is reset and powered on and the phase–locked loop is set to the frequency specified by a previous **Hardware Frequency** command (standard 38 MHz).

Hardware Blink 0|1|2|3 *card_address*

Sets the "blinking" state of the yellow LED on one or more cards. These states are:
0: off **1**: slow blink **2**: fast blink **3**: on

In normal operation the yellow LED indicates the following:

ON:	not yet initialized or cannot be
Blinking:	initializing or in need of service
OFF:	operational

Hardware Diagnostic

This command runs one of a set of builtin diagnostic routines. Test number zero means to run all of the tests in order. The following tests exist:

1 Write/Read floating 0 and floating 1 patterns to the TCS slave's action register 7 (a RAM register for testing Master–to–Slave communication path).

2 Write/Read floating 0 and floating 1 patterns to SIGA–A on the current default slave's board. This tests the communication path from the TCS Master to the SIGA.

3 Write/Read floating 0 and floating 1 patterns to location zero (32–bits) of the processor's RAM. This tests the path from the SIGA through the T–bus to the RAM.

4 Write/Read the values 00 through FF to locations 00 through FF of the processor's RAM. Then write/read the complement of those values. This tests the addressing of processor's RAM, but only the low eight address lines.

5 Test that the 88K processor can be started and that it can access memory. This test loads a short program into memory. If the program ran, the test passes.

Hardware Go *address*

Starts processor at memory *address*. If the address specified is zero or omitted, the last start address from a loaded file is used.

Hardware Halt *card_address*

Halts processor node. Refer to section 2.4.3 for a description of how card addresses are defined.

Hardware PDU on|off 1|2|both

Be sure to write-protect the disk drives before using this command. If you execute the command without protecting the disks you can lose data.

Turns **on/off** bulk power supplies for the system by controlling main power relays **1** and **2**. If TEX is started with a command file or with the **-Autoboot** switch, it will turn on both PDUs when starting.

Hardware NMI-Flag

This command is a shorthand for **Memory Deposit Word 0x01F60004 1**. Writing a "one" to this location asserts the non-maskable interrupt flag in the processor. This is used by the operating system to force entry into its debugger (**kdb**).

Hardware Power on|off *card_address*

Turns **on/off** power at a card or group of cards. If **off** is specified with *card_address* "all", all cards are turned off. **Hardware Power On All** is the same as **Hardware Initialize** and is used to sequentially power on and initialize all cards.

Hardware Temperature *card_address*

Displays a report of the temperature found at the sensor(s) on specified card(s).

Hardware Voltage *card_address*

Displays a report of the various voltages on specified card(s).

Memory Deposit b|h|w *address data*

Performs one write of **data** into the processor memory space. This command should be run while the system is halted.

b, **h**, and **w** indicate byte, halfword, or word cycle respectively.

address is the memory space to which *data* is to be written. *address* does not have to be in local RAM.

Memory Load-File *filename*

Loads *filename* into the current default node, which should be a processor (not a switch or clock).

Header information is summarized on the console and periods are displayed as the loading proceeds (one period per 256 bytes). The default starting address for **Hardware Go** is set from *filename*'s entry vector. Note **Memory Load-File** does not start the processor, for that you must use **Hardware Go**.

Memory UBWait *card_address*
Causes system to wait for microboot to finish. This command should be used in a script file to allow the microboot to finish. It knows where the microboot's state register lives in processor memory and monitors that cell for completion. It will time out after a while in case the boot fails.
Use **Memory UBWait** to wait for the microboot to finish before loading the next application or bootstrap program.

Terminal–Type vt320|vt100|sun|x|pc
Tells TEX menu interface what type of terminal is in use so that it can properly maintain what appears on your terminal screen. The **x** is used for X windows and **pc** for a standard PC monitor.

Terminal–Type hardcopy
Tells TEX to not display the menus to save paper while using a hardcopy–based console.

Terminal–Type menu on|off
Turns **on** or **off** the TEX menu displays on the terminal. You can always type a question mark **?** to display the current menu.

Utility Do *filename*
Executes command(s) contained in *filename* located on the TCS hard disk.

Utility Preview–Cards
Scan all possible card addresses for any response. It reports the number of cards of each type that are present. This also allows TEX to build its internal table of existing cards which is used by the * form of card addresses and by the **Application Run** command.

Utility Slave
Sets the default address for those commands which allow a default. See also the **king** command. The **king** or **master** node can be set differently from this default slave address. The **king** and **default** addresses apply to different sets of commands.

Utility Version
Displays current version number and compilation date.

Quit Quits from the current menu and to DOS if at the main menu.

Quit–to–DOS Quits to DOS from any menu.

2.4.2 Menu-Item/Command Summary

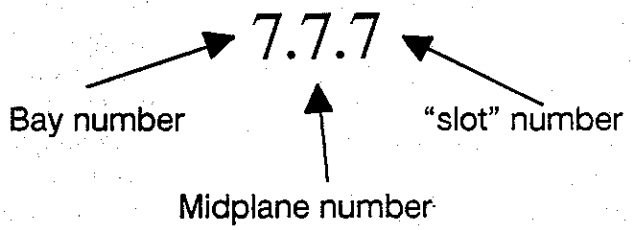
This is a summary of some important TEX menu-items which you may also use as commands.

Menu-Item/Command	Arguments	Description
?		Displays current menu-items with usage.
Application Configuration		Writes config table into node
Application Examine-Card	*card_address*	Displays information on the specified card(s).
Application File-Select	u|p|a *filename*	Selects file for bootstraps: **u** loads microboot; **p** loads power-on tests; **a** loads the application
Application King-Node	*card_address*	Defines king node for application
Application Mode	i|a|r|o	Sets system mode: **i**dle, **a**pplication, **r**eboot, **o**ff
Application Run		Start program or bootstrap system
Application System-Status		Displays the state of nodes in system
Application TTY	*card_address*	Performs console TTY protocol
Configuration Card-use	e|d|f|o *card_address*	**e**nable/**d**isable/**f**orget/**o**ff
Configuration Forget		Forgets configuration info
Configuration Reconfigure		Re-reads configuration files
Configuration Scan	file|large|medium|small	Scans system looking for TCS slaves
Configuration Write Slot	*filename*	Creates slot configuration file
Configuration Write Boot	*filename*	Creates bootstrap configuration file
Hardware Blink	0-3 *card_address*	Controls LED on a card
Hardware Go	*address*	Starts node at memory address
Hardware Halt	*card_address*	Stops a node
Hardware Initialize	*card_address*	Initializes a card
Hardware NMI-Flag		Sets the NMI flag allowing entry to **kdb**
Hardware PDU	on|off 1|2|both	Turns on or off main power supplies
Hardware Power	on|off *card_address*	Turns power on or off at card(s)
Hardware Temperature	*card_address*	Reports temperature of card(s)
Hardware Voltage	*card_address*	Reports voltage(s) on card
Memory Deposit	b|h|w *address data*	Writes **data** into node main memory at *address*. **b**, **h**, and **w** indicate byte, halfword, or word cycle respectively.
Memory Ubwait	*card_address*	Waits for microboot to finish
Terminal-Type	vt320|vt100|sun|x|pc|hardcopy|menu-on|menu-off	Tells TEX menu interface what type of terminal is in use so that it can properly maintain what appears on your terminal screen.
Utility Do	*filename*	Executes script of commands in *filename*
Utility Slave		Sets default target TCS address for commands which allow defaults or take no address argument.
Utility Version		Reports the version number of TEX
Quit		Quit to the previous menu or to DOS
Quit-to-DOS		Quit to the previous menu or to DOS

2.4.3 Card Addresses

Many of TEX's commands refer to a specific TC2000 card or a group of cards. In the descriptions below, the term *card address* means a description of this address argument to a TEX command. The simplest case of a *card address* is a group of three numbers separated by periods. For example, the command **blink 0 7.7.7** turns off the yellow LED on the processor node card which is located in bay 7, midplane 7, slot 7 of the machine.

Figure 2–6 Card Address Format



The third field of a card address, the "slot" number, can also refer to a switch or clock card. The "slots" SA, SB, RA, RB, CA and CB refer to the A and B Switch Server cards, Switch Requestor cards and Clock generator cards respectively (these can be upper– or lower–case). For example, the command **Hardware Blink 0 7.7.RA** turns off the yellow LED on the Requestor card of the A switch in bay 7, midplane 7.

For the clock cards, the Bay and Midplane fields should be omitted. The command **Hardware Blink 0 CA** turns off the yellow LED on Clock card A.

The *card address* field can also be replaced by the letter **g** and a hex number from **0** to **FF**. This indicates that the system is to use a target address containing a "Group address". The command **Hardware Blink 0 G4** sends a command to all slaves which respond to group address 4 (normally all processor nodes). The special address "ALL" is recognized as the "broadcast" group address. Not all commands will accept a group address.

Card addresses can also contain "wildcards" (* or !). These cause a command to be executed for each of a number of addresses. For example, **Hardware Blink 0 6.*.*** performs a **Hardware Blink 0** command for each existing node in bay 6. The command **Hardware Blink 0 *.*.R*** turns off the yellow LED on each existing Switch Requestor card connected to the machine switch. The slots **C***, **R*** and **S*** mean both A and B clock, switch receiver or switch sender cards, respectively.

In the preceding paragraph, "existing" means that a card was mentioned in the slot configuration file or that it has responded to a message from TEX since TEX was started. This speeds operations up by skipping cards which are not

actually in the machine. If the * is replaced by a !, however, the wild card will be taken to include all cards in the described range, even if they have not been heard from and are not in the slot configuration file.

2.4.4 Configuration Files

When TEX is invoked with the **-autoboot** switch, it automatically reads the two configuration files, **SLOTCFG.TCS** and **BOOTCFG.TCS**, in the current directory. **SLOTCFG.TCS** is a machine-dependent file that is generated during the TCS installation. **BOOTCFG.TCS** is a default bootstrap configuration file. Both files are described in the following sections.

SLOTCFG.TCS

The **SLOTCFG.TCS** file is generated during the installation of TCS. It has one line for each card with its card address, type, serial number, and revision levels. You should never have to manually edit this file. If your configuration changes, you can generate a new **SLOTCFG.TCS** file with the command sequence (make sure you start TEX from the \TCS directory):

```
TEX -> Config scan medium ↲
TEX -> write s slotcfg.tcs ↲
```

This creates a new **SLOTCFG.TCS** file with the current configurations.

The cards listed in this file are used in the "*" wildcard arguments that some of the commands described in section 2.4.1 can accept. For example, the command sequence **Hardware Temperature *.*.*** will report the temperatures of all the processor cards contained in **SLOTCFG.TCS**. **SLOTCFG.TCS** helps TEX to avoid polling cards which are disabled or isolated. Similarly, the **run** command, which initializes all the cards in the machine, will attempt to initialize cards only if they are contained in **SLOTCFG.TCS**.

BOOTCFG.TCS

The **BOOTCFG.TCS** file delivered with the TC2000 TCS software contains default bootstrap commands. Different versions of this file are kept in the different directories of the TCS software hierarchy. For example, the directory for diagnostics usually has a different **BOOTCFG.TCS** from the one for booting in \TCS. **BOOTCFG.TCS** contains a number of boot parameters, including automatic boot up of the nX operating system, and specifies what cards should be disabled (non-initialized), etc. Figure 2-7 illustrates the default **BOOTCFG.TCS** file in the \TCS directory.

Figure 2–7 Default BOOTCFG.TCS File

```
bootparam: 0: 0
bootparam: 1: 5
bootparam: 2: 1
bootstring: (xy0,0,0)/vmunix
autoboot: Yes
primaryclock: A
primarynode: 7.7.7
switchfreq: 38
ubootfile: uboot.88
postfile: post.88
bootfile: boot.88
alternatepaths: A: Yes
; End of BOOTCFG.TCS
```

A number of commands can be contained in **BOOTCFG.TCS**. Each command line has the command name followed immediately by a colon, optional spaces, and the command argument. The commands are **not** case-sensitive. The following is a description of the commands that **BOOTCFG.TCS** can use. Normally, you do not need to edit this file.

alternatepaths Tells TEX whether or not to set up the processor cards to use available Butterfly switch alternate paths. A redundant system contains two switch paths. By default, the alternate paths of both switches are enabled for improved switch performance in the \TCS directory **BOOTCFG.TCS** file. To simplify debugging, the alternate paths are disabled in the \DIAG directories.

This command takes two arguments. The first argument can be either **A** or **B**, specifying one of the switches in the redundant system. This argument is followed immediately by a colon. The second argument is either **Yes** or **No**, specifying wether or not the switch path is used.

autoboot Takes one argument, either **Yes** or **No**. **Yes** causes TEX to boot an application automatically. **No** halts TEX after reading configuration files, and waits at the TEX prompt.

bootfile Takes one argument, a *filename* for a file on the TCS master's disk. This file is run on the primary node after the microboot and POST programs have been run on all nodes.

bootparam Takes two arguments, both decimal numbers. The first number selects one of sixteen bootstrap variables. The second number is what the variable is set to.

Currently, only variable numbers 0, 1, and 2 mean anything (panic count, panic limit, and multiuser). These are as follows:

0: 0	Set the current panic count to 0.
1: 10	Tells TEX to try rebooting the machine 10 times before halting.
2: 0	Prompts the nX boot program to ask the user if he/she wants to boot single or multiuser after the **tex -auto** or **start** commands.
2: 1	Prompts the nX boot program to automatically come up multiuser at **tex -auto** or **start**.

bootstring — Takes anything following the colon (:) as the string argument. The default string is *(xy,0,0,0)/vmunix*. This allows you to simply press the <Return> key at the BOOT: prompt.

carduse — Takes two arguments, a card group address and a "use" definition. The "use" may be one of "system", "auxiliary", "isolated", or "off". See the **Configuration Card-Use** command for further details.

disabledcards — Causes each listed card to be marked as disabled so that it will not be brought up automatically, even though it can respond to TCS commands and can be included in **SLOTCFG.TCS**. It takes one argument, a card address, possibly containing "!"-type wildcards. There can be multiple **disabledcards** commands in **BOOTCFG.TCS**.

This command can be overridden by the TEX commands **Configure card-use system** and **forget**.

postfile — Specifies the file which will be run on each node which successfully completes the microboot. It uses *filename* as its only argument.

primaryclock — Specifies whether TC/CLK card A or B is to be used if both are present. [Currently not implemented.]

primarynode — Specifies which node is to be used for the application bootstrap (**bootfile** command) if it is working. Takes a specific *card address* as its argument. It is also known as "king " or "master"

switchcolumns — Takes one argument, a decimal number, representing the number of columns in the switch. This number must be either *2* or *3*. [Currently, only two switch columns have been implemented.]

switchfreq — Takes one argument, a decimal number, representing the frequency (in Megahertz) of the Butterfly switch. These switches were designed to operate only at 38 MHz. Do not change the switch frequency to anything else.

ubootfile — Specifies the file which will be run on each node listed in **SLOTCFG.TCS** which has not been disabled by a **disabled-cards** command. It uses *filename* as its only argument.

2.4.5 BOOTCFG.TCS File Parameters

Instead of editing the configuration file **BOOTCFG.TCS** using the DOS text editor **EDLIN**, you can change and/or add commands and parameters in this file through the TEX menu. The procedures to change and add **BOOTCFG.TCS** file commands are different. This section describes both.

Changing BOOTCFG.TCS File Parameters

While performing TEX operations on the TC2000, you can change the values of parameters that are already contained in and set up by the **BOOTCFG.TCS** file. This can be done by reading the current file into TEX, performing typical TEX operations like resetting the state of card use, then writing any changes into the back to the file. The following example shows how to change a card's use. The first step is to read in the **BOOTCFG.TCS** file. You can do this with the command sequence of **Configuration Forget** followed by **Configuration Reconfigure**.

Figure 2–8 Changing BOOTCFG.TCS

```
TEX-> c ↲                                  Go to the Configuration menu.
Configuration-> f ↲                        Clear (forget) the configuration.
Configuration-> r ↲                        Reread (reconfigure) BOOTCFG.TCS and
                                           SLOTCFG.TCS. At this time, you can leave the Con-
                                           figuration menu and perform other TEX operations,
                                           such as changing card-use state:
Configuration-> c ↲                        To select a card-use state.
Function: Card-Use
  off|isolate|system|auxiliary|forget
                          <String>
Card group.               <Node-Group>        7.1.2
Card use
Enter off|isolate|system|auxiliary|forget-> o ↲
                                           This would then tell TEX to keep card 7.1.2 turned off
                                           and ignored even when the system reboots. This is
                                           useful for known defective cards.
Configuration-> w ↲                        Go to the Write menu.
Write-> b ↲                                Write the appropriate changes made during this TEX
                                           session to the BOOTCFG.TCS file.
```

The next time you boot the system, the new **BOOTCFG.TCS** is used. Note that this procedure can only be used to change parameters that already exist in the **BOOTCFG.TCS** file. If you want to add parameters that do not exist in **BOOTCFG.TCS**, or parameters that are not normally changed by normal TC2000 operations, but which are available for **BOOTCFG.TCS**, see the following section.

Adding BOOTCFG.TCS File Parameters

In the course of performing TEX operations on the TC2000 machine, you can add new parameters to the current **BOOTCFG.TCS** file, or change some that you did not change during other TC2000 TEX operations. You can do this by reading the current **BOOTCFG.TCS** file into TEX, using the **Configuration set** command to set **BOOTCFG.TCS** parameters, then writing any changes into the **BOOTCFG.TCS** file. The following illustrates an example of what to do if you disable the Autoboot parameter of the **BOOTCFG.TCS** file. The first step is to read in the **BOOTCFG.TCS** file. You can do this with the command **Configuration Forget** followed by **Configuration Reconfigure**.

Figure 2–9 Adding BOOTCFG.TCS parameters

`TEX-> c ↲`	Go to the **Configuration** menu.
`Configuration-> f ↲`	Clear the configuration.
`Configuration-> r ↲`	Reread **BOOTCFG.TCS** and **SLOTCFG.TCS**. At this time, you can leave the Configuration menu and perform other TC2000 operations, *e.g.*, turn processors on and off, check voltage levels, setting new configuration parameters, etc. One of these operations can then be setting up the Autoboot state of the machine.
`Configuration-> s ↲`	Select the **BOOTCFG.TCS set** option.
`Set-> au ↲`	To set the Autoboot parameter.

```
Function: AutoBoot
Set Autoboot Flag to Yes or No.
      Yes or No                    <String>
-> AutoBoot
Enter Yes or No <String> -> No ↲
```

This disables automatic reboots after the **BOOTCFG.TCS** is read during a **tex -a** or **start** command to boot the system. If you want to change parameters that do not exist in the Set menu with TC2000 TEX operations, see the previous section.

The next time you boot the system, the new **BOOTCFG.TCS** is used.

3

Running the Diagnostics

3.1 Overview

This chapter describes how to use use the diagnostic software:

- considerations
- strategy
- entering and exiting diagnostics
- starting diagnostic tests

The five sets of diagnostics are described in the chapters that follow.

3.2 Considerations

Before running the diagnostics:

- If the system crashed or panicked, be sure to dump the system to tape, as described in Section 3.3.4.
- Be sure that the operating system is shut down properly (see Section 3.4)
- Be sure that the tests match the revision levels of the boards (test descriptions specify the board revision levels).
- Check the revision level of the TCS Executive software; this guide applies to revision levels listed in the 'How to Use This Manual' section. The revision is displayed at the main TEX menu.

3.3 Diagnostic Strategy

This section discusses the sequence of diagnostics and how to find problems quickly. It is especially useful for anyone who is unfamiliar with the diagnos-

tics. It deals with two general situations: running diagnostics to verify an installation and running diagnostics to find problems.

3.3.1 Running Diagnostics to Verify an Installation

After installing the system, you should run the five diagnostic suites to ensure that it is operating properly. The fastest way to find problems is to run the tests in the following order:

1. Processor diagnostics establish a reliable base for interpreting the other diagnostic results; they find most problems that are specific to the processors.
2. Switch diagnostics verify that the paths through the machine are working.
3. STP diagnostics provide a good overall test of the processor/switch interaction.. Their results also allow you to find most problems if you have already ruled out the processor and the switch. For an 8–node system, run STP for 80 passes. This takes about an hour. For a 16–node system, run STP for 400 passes (5 hours). For a 24–node system, run STP for 800 passes (10 hours). For a 32–node system, run STP for 1600 passes (20 hours).
4. Standalone Peripheral Diagnostics test the controllers and peripheral devices; you should run the first three diagnostics before running this one.
5. The "all tests" option of the nX Operating System Peripheral Diagnostics (**periph_diag**) provides a system–level test of the peripherals. This test takes about five hours.

3.3.2 Running Diagnostics to Find Problems

If the system has crashed, dump the system to tape, as described in Section 3.3.4. If you are trying to find an intermittent problem or cannot dump the system to tape:

1. Run the first four diagnostics in the order defined in Section 3.3.1.
2. If all tests pass, reboot the system. If any test fails, refer to the appropriate chapter on that test to diagnose the problem.
3. If system crashes persist and hardware is suspected, the same diagnostics can be run to give the system a more thorough test.

3.3.3 Diagnosing System Problems

The first step in finding problems is to dump the system to tape (if possible) as described in Section 3.3.4. Return this to BBN ACI for analysis; it could prove essential in finding some problems.

System Panics in kdb

Most problems with the system result in a panic message and leave the system in the **kdb** debugger (assuming **kdb** was activated). If the system is in **kdb**, follow the crashdump procedure in Section 3.3.4.

System Panics without kdb

If the system does not end up in **kdb**, try to enter **kdb** through TEX by typing backquote period (`.) to get into TEX and then typing the appropriate commands:

TEX -> **hard nmi** ↵	Sets the NMI flag allowing entry to **kdb**
TEX -> **app run** ↵	Starts the "application"

Then follow the crashdump procedure in Section 3.3.4.

System Hangs

If the system is hung and will not respond to <Control>C, <Control>Z, <Control>D, <Control>Q, or <Return> check the RECEIVE light on the master node. If it is on, look for the SEND light on one of the slaves. If it is on, unplug this slave from the system. This will cause the system to panic and leave the system in the **kdb** debugger (assuming **kdb** was activated). Then follow the crashdump procedure in Subsection 3.3.4.

System Crashes

If the system crashes and the TCS does not respond to the keyboard, try to reset the TCS and get into **kdb**. To reset the TCS, use the system reset switch.

It this does not work, try shutting down the system and rebooting. ***This should be your last resort, since rebooting destroys all system information that might be helpful in diagnosing the problem.***

3.3.4 Dumping a System to Tape (crashdump)

NOTE

If you have no time to produce a crash dump tape of all the nodes in your system panic (nodes take about 4 minutes each to dump) call BBN ACI at 1-800-4AC-BFLY. The customer support staff may be able to help you to identify a subset of the nodes that you need to dump to tape.

Note also that one SCSI cartridge tape cannot accommodate more than 32 nodes of crash dump information. If your tape runs out before all the nodes are dumped, simply send to BBN ACI what was actually saved on that single tape.

To dump a system to tape, **kdb** must be running. If **kdb** is running follow the procedure in Figure 3-1. If not, try to activate **kdb** by using one of the methods described in Subsection 3.3.3. (For more information on dumping the system. see the *TC2000 System Administration Guide.*)

Figure 3-1 Crashdump Procedure

`# $%` ↲	Starts the dump session
`Please put a tape in the tapedrive, type 'y' to start dump...` **y**	Insert the tape into the 1/4" tape drive. Type **y** to start the dump.
`Do you wish to dump all nodes? Please type 'y' or 'n'...`	
y	You should respond with a **y** "yes". The system then proceeds to dump all nodes, displaying miscellaneous messages.

`Dumping all nodes.`
`Dumping node 0 = (physical node 0x8) 512 pages`
`Dumping node 1 = (physical node 0x9) 512 pages`
.
.
.
`Dump completed...`
	If you typed an **n** "no", you would be prompted for each node.

To exit the **kdb** session, type:

:c

3.4 Entering and Exiting Diagnostics

The diagnostics are started from one of three places; the DOS prompt, the TEX prompt, or the nX OS prompt. If you are running diagnostics for the first time after a system installation (or re-installation), follow the instructions in Subsection 3.3.1. If the operating system is running, you must halt it first. If the system is crashed, the console may display the **kdb**, TEX or DOS prompt (see Chapter 2).

DOS-based Diagnostics:

B2SWITCH	for switch diagnostics
B2CLK	for TC/CLK diagnostics
B2MODEM	for modem diagnostics

TEX-based Diagnostics:

B2VME2	for B2VME diagnostics
TCFPV	for TC/FPV diagnostics
STP	for STP diagnosticss
VMEDIAG	for peripheral diagnostics
DISKTOOL	for SCSI disk exercising

B2MDM requires rebooting from the second diagnostic floppy disk (See Chapter 10).

nX OS-based Diagnostics:

periph_diag	for peripherals diagnostics
stress	for peripheral and system exercising
disktool	for SCSI disk exercising

periph_diag and **stress** are a separate programs that run from the nX Operating System (see Chapters 8 and 9).

3.4.1 Starting Diagnostics from DOS

To start a diagnostic from the DOS prompt, type the following:

`C:\TCS>`**CD \DIAG***diagname*↲ Changes to the desired directory.where *diagname* is one of the DOS-based or TEX-based diagnostics as listed above.

`C:\diag\`*diagname*`>` **START**↲ Initializes the system and displays the diagnostic's prompt.

3.4.2 Starting Diagnostics from TEX

To start a TEX-based diagnostic from the TEX main menu, follow the example in Figure 3-2. Although abbreviations have been used you may wish to type the full command name or use the menus. The **BOOTCFG.TCS** file in the current directory will be used; that file may differ from the one in the **diag** directory. For instance, the **diag** directory **BOOTCFG.TCS** file disables alternate paths by default.

Figure 3-2 Starting Diagnostics from TEX

```
TEX -> c r↲                          Configure Reconfigure; Reconfigures the system
TCS: Previewing machine state...
                .
                .
                .
; End of BOOTCFG.TCS
TEX -> a f a↲                        Application File-Select Application; Load the following application
Function: Application
Application Bootstrap File.
    Filename                 <String>
-> Application
 Enter Filename -> \diag\diagname\diagname.88   This is the diagnostic to load, where diagname is one of the TEX-based diagnostics.
TEX -> a r↲                          Application Run; Starts the loaded application.
Clock card CA already running.
                .
                .
                .
TCS: Switching console terminal to application.
Clearing bss (0x33860 -> 0x33940)...Done
```

3.4.3 Running Diagnostics

Most of the diagnostics start by prompting you for the type of terminal you are currently using. At the terminal-type menu prompt, select the terminal type that is the closest to the type you are using, by typing in the appropriate number and pressing <Return>.

If you are using a printer or dumb terminal, type **6**<Return>. The diagnostics won't use screen refreshes, highlighting, or automatic menus display (to save paper). You can type **?**<Return> to display menu choices.

You may need to quit from the terminal-type menu to get to the diagnostics main menu. You can then select the tests you want to run. The terminal type is listed on the same line as the Terminal Type menu item.

See the appropriate chapter for details of the desired diagnostic.

```
Menu:  Terminal-Type
    1. Quit
    2. VT320
    3. VT100
    4. Sun
    5. X
    6. PC
    7. Hardcopy
    8. Menu
    9. More
                                    ver x.xx
Terminal-Type ->
```

3.4.4 Exiting Diagnostics

To exit from the diagnostics, you must first return to the TEX prompt, as described in Chapter 2.

When the TEX diagnostic has finished, return to the TEX prompt by typing a backquote, period (`.). Note that the backquote is not echoed on the screen until you type the period. You can then quit out of TEX or start the next diagnostic.

4

Processor Diagnostics

4.1 Selecting the Diagnostic

This chapter includes information on processor diagnostics, for B2VME and TC/FPV processors. There are two processor diagnostics, one for each of the two processor types. To find out which type of processors are in the system, type the following at the TEX prompt:

```
TEX -> c scan      Configure: Scan
```

When TEX prompts for size use **small** for systems with 8 or less, **medium** for systems with 9–32, and **large** for systems with greater than 32 processors. TEX displays the processor type and revision levels as below.

```
7.7.0:   Type: '8' B2VME-B, (Processor node)  Ser #: H946-18
7.7.0:   Type: 'G' TC/FPV-4, (Processor node) Ser #: C007-03
```

TC2000 systems may have more than one type of processor. Use the appropriate diagnostic for the processor in the system:

Processor	Card type	Diagnostic to use	Section to turn to
B2VME	8 or A	B2VME2.88	4.2
TC/FPV	G or H	TCFPV.88	4.5

4.2 B2VME2 Overview

This section describes the diagnostics for the TC2000 B2VME card (type 8 or A). This diagnostic is useful for verifying an installation, locating a possible processor fault, or confirming that a particular processor is bad.

The purpose of these tests are to fully test the hardware on the B2VME card. There are a number of script files that save the time of loading the diagnostic in each processor.

4.3 How to use the B2VME2 Diagnostics

Run the B2VME2 diagnostics on each B2VME; first the master node and then each slave node. After initializing and loading the diagnostic, the system displays the terminal-type menu. Select your terminal. The diagnostic menu is displayed. To exit from the diagnostics, type a backquote and a period (`.) or type **quit** at the B2VME2 main menu. This brings up the TEX prompt.

4.3.1 Run Times

The typical time to run the system script on a 4 MB B2VME is 15 minutes and on a 16 MB B2VME it is about one hour.

4.3.2 Initializing the System and Loading Diagnostics

The B2VME2 diagnostics are started from the DOS or TEX prompt (see Chapter 3). From DOS by type:

`C:\` **CD \DIAG\B2VME2** ↵	Changes to the B2VME directory.
`C:\DIAG\B2VME2>` **START** ↵	Initializes the system.

This brings up the initial (terminal type) menu. Select your terminal type. This brings up the B2VME2 diagnostic main menu:

```
Menu:  B2VME2                                  Node Number = 7.7.7

1. Memory-Systems
2. Siga-Tests
3. Local-Registers
4. Augmentation-Tests
5. Board-Options
6. Software-Options
7. Execute-Scripts
8. Terminal-Type
9. Quit                                        vt320, Menu-On
```

4.3.3 Executing the Test Scripts

Select the Execute-Scripts menu (type **Ex**). This brings up the Power-Up and Field-Level Diagnostic menu. (The rest of the diagnostics are for board-level testing and should only be used at BBN.)

```
1. Power-up-Diagnostics
2. Bench-Power-up-Diagnostics
3. 4 Power-up-Diagnostics_REV_A4
4. Manufacturing-Diagnostics
5. Bench-Manufacturing-Diagnostics
6. Field-Diagnostics
7. Quit
```

To execute the B2VME2 diagnostics, type **1** or **6**. Currently, both of these tests are the same. The end results are something similar to the following:

```
Ram-Data-Test: Test Passed
Field Diagnostics: Test Passed
```

At this point you may rerun tests or go back to the TEX menu.

4.4 Executing on Multiple B2VMEs

There are several scripts included in the diagnostic directory to assist you in loading all the B2VMEs in the system at once. The scripts are set up to load eight nodes at once, in one of 4 configurations (midplanes 7, 6, 5 and 4) or 32 nodes all at once. To run one of these scripts, first initialize the system (see Subsection 4.3.2). Then type:

TEX -> **do load7.tex**

This loads and start all of the B2VMEs associated with mid-plane 7 (7.7.0 through 7.7.7), but you still need to still need to run the actual diagnostic on each loaded node. When it finishes loading, the TEX prompt is displayed.

You can then run the diagnostics on the master node by typing **a tty** this displays the terminal-type menu (shown in Subsection 11.2.2). To test another B2VME, go back to TEX by typing backquote and a period (**`.**) then type the following commands:

TEX -> **u slave x.y.z** where x.y.z is the number of another slave started by the script

TEX -> **a tty**

Proceed until all nodes are tested in the 7.7.x midplane series.

There are 4 other load files. These are:

load6.tex (7.6 midplane)
load5.tex (7.5 midplane)
load4.tex (7.4 midplane)
load32.tex (7.all midplanes) This takes about 1/2 hour to load.

4.5 TCFPV Overview

This section describes the diagnostics for the TC2000 TC/FPV card (type G or H). This diagnostic is useful for verifying an installation, locating a possible processor fault, or confirming that a particular processor is bad.

The purpose of these tests are to fully test the hardware on the TC/FPV card. There are a number of script files that save the time of loading the diagnostic in each processor.

4.6 How to use the TCFPV Diagnostics

Run the TCFPV diagnostics on each TC/FPV; first the master node and then each slave node. After initializing and loading the diagnostic, the system displays the terminal-type menu. Select your terminal type. The TC/FPV menu is displayed. Select the Execute-Scripts menu item, then choose a script to run. To exit from the diagnostics, type a backquote and a period (`.) or type **quit** at the TC/FPV main menu. This brings up the TEX prompt.

4.6.1 Run Times

The typical time to run the system script on a 4 MB TC/FPV is 7 minutes and on a 16 MB it is about one half hour. The typical time for the Fast-System-Script is two minutes, since it only uses a small amount (8 kb) of memory for testing.

4.6.2 Initializing the System and Loading Diagnostics

The TC/FPV diagnostics are started from the DOS or TEX prompt (see Chapter 3). From DOS type:

`C:\` **CD \DIAG\TCFPV** ↵	Changes to the TCFPV directory.
`C:\DIAG\TCFPV>` **START** ↵	Initializes the system.

Once loaded, the initial (terminal type) menu is displayed. Select your terminal type. This brings up the TC/FPV diagnostic main menu.

```
Menu:  TC/FPV                                  Node Number = 7.7.7
     1. Memory-Systems
     2. Cpu-88k-Tests
     3. Siga-Tests
     4. Local-Registers
     5. Board-Options
     6. Software-Options
     7. Execute-Scripts
     8. Terminal-Type                    vt320, Menu-On
     9. Debugger
     10. Quit
```

4.6.3 Executing the Test Scripts

Select the Execute-Scripts menu (type **Ex**). This brings up the Diagnostic Scripts menu:

```
Menu:  TC/FPV Script-Menu
     1. Fast-System-Diags
     2. Fast-Bench-Diags
     3. System-Diags
     4. Bench-Diags
     5. Manufacturing-Diags
     6. Manufacturing-Bench-Diags
     7. Run-All-Scripts
     7. Quit
```

To execute the TC/FPV test script, type **3** for a complete board test or type **1** for a quick test. The rest of the diagnostics are for board-level testing and should only be used at BBN. The results for **1** or **3** are similar to the following:

```
Ram-Data-Test: Test Passed
Field Diagnostics: Test Passed
```

At this point you may rerun tests or go back to the TEX menu. If you want to stop the current script type a <Control-C>.

4.7 Executing on Multiple TC/FPVs

There are several scripts included in the diagnostic directory to assist you in loading all the TC/FPVs in the system at once. The scripts are set up to load eight nodes at once, in one of 4 configurations (midplanes 7, 6, 5 and 4) or 32 nodes all at once. To run one of these scripts, first initialize the system (see Subsection 4.6.2). Then type

```
TEX -> u do load7.tex
```

This loads and starts all of the TC/FPVs associated with mid-plane 7 (7.7.0 through 7.7.7), but you still need to run the actual diagnostic on each loaded node. When it finishes loading, the TEX prompt is displayed.

You can then run the diagnostics on the master node by typing **a tty** this displays the terminal-type menu (shown in Subsection 11.2.2). To test another TC/FPV, go back to TEX (‘.) and type the following commands:

```
TEX -> u slave x.y.z        where x.y.z is the number of another slave
                            started by the script
TEX -> a tty
```

You may now run the diagnostics on the next node.

Exit the TCFPV diagnostic by typing backquote and a period (‘.). You can then run another script or specify and test another TC/FPV.

Proceed until all nodes are tested in the 7.7.x midplane series.

There are 8 other load files. These are:

```
load6.tex  (7.6 midplane)
load5.tex  (7.5 midplane)
      .
      .
      .
load0.tex  (7.0 midplane)
load32.tex (7.all midplanes)    This file takes about 1/2 hour to load.
```

5

Switch Diagnostics

5.1 Overview

This chapter describes the diagnostic for the TC2000 Switch Cards. This diagnostic is useful for verifying an installation, locating a possible switch fault, or confirming that a particular switch card is bad.

The purpose of these tests are to fully test the hardware on the TC/SS and TC/SR cards.

5.2 How to use the Switch Diagnostics

After initializing, start with the Switch Status Function, then select the Connectivity tests on the whole system to help isolate any problems. To exit from the diagnostics type **quit** at the Switch Main Menu. This returns you to the DOS prompt.

5.2.1 Run Times

The typical time to run the Switch test is 10 to 15 minutes per TC2000 Switch.

5.3 Executing the Switch Diagnostics

To run the switch diagnostics, you must be at the DOS prompt. After initializing and loading the diagnostic, the system displays the terminal-type menu. From this menu set up your terminal and then quit. This will bring up the diagnostic menu.

5.3.1 Initializing the System and Loading Diagnostics

The switch diagnostics can only be started from the DOS prompt:

`C:\>` **CD \DIAG\B2SWITCH** ↵	Changes to the B2SWITCH directory.
`C:\DIAG\B2SWITCH>` **START** ↵	Initializes the system.

This brings up the initial (terminal type) menu. After selecting your terminal type the switch diagnostic main menu is displayed.

Typing **quit** at the main menu prompt returns you to the DOS prompt because the switch diagnostics execute in the TCS and do not use TEX.

5.4 Switch Diagnostic Main Menu

The switch diagnostic main menu give you the following choices:

```
Menu:  TCS
    1. TC/SS-Diags                    BE SURE SYSTEM IS INITIALIZED
    2. Initialize-System
    3. Display-Switch-Configuration
    4. Logging-On/Off                 Logging Off
    5. Terminal-Type                  Hardcopy, Menu-On
    6. Quit
```

Always begin by initializing the system (**2**), then select the TC/SS-Diags (**1**).

5.5 Initialize System

If you are not sure that the system is initialized, it is recommended that you do so. This resets all boards and should not be executed while the system is running the nX operating system.

After selecting this item from the main menu, you are presented with the following prompts:

```
Function: Initialize-System
Initialize Clock
    Clock (a/b) or Test Jig (t)
                              <String>            A
    Frequency                 <Number>            40
-> Initialize-System
Enter Clock (a/b) or Test Jig(t) <String> -> (A)↵

-> Initialize-System A
Enter Frequency <Number> -> (40)38↵

-> Initialize-System A 38
This will completely re-initialize the system!!!
Do you want to continue  (y or n) y

[typing y results in the following messages]

Scanning System for Processors and switch Cards
 . . . . . .
Scanning System for Processors and switch Cards
 . . . . . .
Initialize System:  Test Passed
```

The defaults are "A" for the clock and "40" (40MHz) for the frequency. Currently, you **must** change the frequency to 38MHz or the system will not operate correctly. Pressing any key returns you to the main menu.

5.6 Display Switch Configuration

The Display-Switch-Configuration function is a configuration aid. It is executed by selecting item **3** from the Diagnostic Main Menu. Once that item is selected, the menu is displayed.

This menu displays the connections from switch to switch for the number of midplanes. It prompts you to enter the number of midplanes in the system and then a table of connections is displayed (and logged if logging is turned on).

```
Function: Display-Switch-Configuration
Switch Configuration
    Midplanes                 <Number>
-> Display-Switch-Configuration
Enter Midplanes <Number> -> ()7↵
-> Display-Switch-Configuration 7
Displaying Configuration for 7 Midplanes

Connect Midplane 7 connector 7 to Midplane 7 connector 7
Connect Midplane 7 connector 6 to Midplane 6 connector 7
Connect Midplane 7 connector 5 to Midplane 5 connector 7
    .
    .  more messages
    .
Connect Midplane 1 connector 3 to Midplane 3 connector 1
Connect Midplane 1 connector 2 to Midplane 2 connector 1
Connect Midplane 1 connector 1 to Midplane 1 connector 1
```

5.7 Logging On/Off

The logging function saves results of a test into a file. To turn on logging select item **4** from the Diagnostic Main Menu. You are prompted for a filename; if the file exists you may overwrite it or append the results to the end of it. To turn off logging select item **4** and don't enter a filename (press <Return>).

5.8 TC/SS Diagnostics

Selecting item **1** from the Diagnostic Main Menu displays a sub–menu that prompts you for the type of testing you plan to do.

```
Menu:  TCS TC/SS(SR)-Switch-Diags
    1. Board-Level-Diags
    2. System-Level-Diags
    3. Command-Level-Diags
    4. Quit
```

5.8.1 Board Level Diagnostics

These diagnostics are used for 'burning in' data into the EEPROMs on the switch boards. This is unlikely to be used in normal testing and troubleshooting operations, and is not covered in this guide.

5.8.2 System Level Diagnostics

The System Level Diagnostics Menu is displayed by selecting item 2 from the TCS TC/SS(SR)–Switch–Diagnostics Menu shown in Section 5.8. Once that item is selected, the following screen is displayed.

```
Menu:  TCS TC/SS(SR)-Switch-Diags System-Level-Diags
    1. Switch-Status
    2. Connectivity-Test
    3. Margin-Power
    4. Bulk-Voltage-Read
    5. Temperature-Read
    6. Quit
```

Switch Status Function

Selecting item **1** from the System Level Menu brings up the Switch Status menu.. Running this test reports the activity status of the specified switch boards. The test prompts you to specify which switches to test as shown in the following screen:

```
Menu:  TCS TC/SS(SR)-Switch-Diags System-Level-Diags Switch-
Status
    1. Add-a-Path
    2. Delete-a-Path
    3. Test-for-all-Paths
    4. Logging-On/Off                    Logging Off
    5. Run-Test
    6. Quit
```

To check all the paths, select item **3**. To check individual paths, or a subset of paths, add and delete paths with items **1** and **2**. When editing paths the current list of addresses will be displayed between the menu and the prompt line. After you have entered all the addresses select item **5** (Run–Test).

NOTE

The b2switch status test always reports that the clock signal is bad; ignore this message if your system boots and runs.

For each switch requestor and server the following information is displayed:

```
Serial Number C937-05   Address 7.7.9   Server

Clock Signal Bad  Temperature OK Voltage Levels OK Slave Processor OK

INPUT PORT      0   1   2   3   4   5   6   7
Activity        0   0   0   0   0   0   0   0

OUTPUT PORT     0   1   2   3   4   5   6   7
Activity        0   0   0   0   0   0   0   0
Priority        0   0   0   0   0   0   0   0

Revisions
        Artwork         A
        Electrical      A
        TCS Slave       B1

Test Passed
```

Connectivity Tests

Selecting item **2** from the System Level Menu displays the Connectivity Tests Menu. This tests the connections between the processors and the switch and between the server and the requestor. This does not test out all the paths or the switch board logic.

When the connectivity test is selected the following menu is displayed:

```
Menu:  TCS TC/SS(SR) ... System-Level-Diags Con-Tests
    1. Processor-To-Switch-Test
    2. Switch-To-Switch-Test
    3. Switch-To-Clock-Test
    4. Full-System-Test
    5. Quit
```

The first three tests are very similar in operation. Item **1** tests the control and data lines from the processor to the switch (both requestor and server). Item **2** tests the control and data lines from the requestor to the server. The path sub-menu for both tests is similar to the Switch Status Function. The number of entries that make up a path is slightly different. Turning off Verbose mode allows you to get a simple pass/fail type test instead of the complete information from the test.

1. Processor–to–Switch–Test

To run the processor–to–switch test, type **1** at the menu prompt. The processor–to–switch test menu is displayed:

```
Menu:  TCS TC/SS(SR)-Switch-Diags ... Proc-to-Switch-Test
    1. Add-a-Path
    2. Delete-a-Path
    3. Test-for-all-Paths
    4. Logging-On/Off                    Logging Off
    5. Verbose-On/Off                    Verbose Mode On
    6. Run-Test
    7. Quit
```

Type **3** to test for all paths, and then **6** to run the tests. Then respond to the queries as follows:

```
This will interrupt traffic on the switch
Do you want to continue? (y or n) y        Don't type a <Return>.

Scanning System for Processors and Switch Cards
                .
                .          more messages.
                .
Checking data paths; sent 64 received 64
Checking data paths; sent 128 received 128
Press any key to continue
```

4. Full–System–Test

Item **4** (Full–System–Test) tests all paths for both the Switch to Switch connections and the Processor to Switch connection. Select **4** to display the menu:

```
Menu:  TCS TC/SS(SR)-Switch-Diags ... Full-System-Test
    1. Logging-On/Off                    Logging Off
    2. Verbose-On/Off                    Verbose Mode On
    3. Run-Test
    4. Quit

Full-System-Test ->
```

The Full–System–Test tests all paths for both the Switch–to–Switch connections and the Processor–to–Switch connection. On the Full–System–Test menu

you may toggle both logging and verbose mode. After setting the paths you want to test, select item **3** (Run–Test). Then respond as follows:

```
This will interrupt traffic on the switch
Do you want to continue? (y or n) y    Don't type a <Return>.


Scanning System for Processors and Switch Cards
                .
                .                      more messages.
                .
Press any key to continue
Checking communications from . . .
                .                      System will return with many screens of
                .                      messages, each of which end with Press
                .                      any key to continue. Press any key to
                                       continue. The main menu is displayed
                                       when the test is completed.
Press any key to continue
```

Changing Verbosity Mode

You can change the verbosity mode in any of the switch tests. Select verbosity with (option **5**) followed by **0** (zero for verbosity mode **off**) or **1** (one for verbosity mode **on**). You can also just type the verbose menu item number and respond to the prompt. Note, however, that you can do this for any of the tests with the verbosity option.

```
Proc-to-Switch-Test -> 5 0↵
```

The 5 selects the Verbose–On/Off option, and the 0 indicates verbosity mode off. The dialog then returns you to the main menu.

5.8.3 Command Level Diagnostics

These diagnostics are used for debugging an individual switch board. They are unlikely to be used in normal testing and troubleshooting operations at the system level, and are not covered in this guide.

6

Sinister Test Program (STP) Diagnostics

6.1 Overview

The Sinister Test Program (STP) was not initially intended for use outside BBN, so it is not user-friendly. You can follow the sample session in Section 6.3 and analyze the results by referencing sections starting with Section 6.5.7.

This diagnostic is used to exercise multiple-processor TC2000 systems. It runs on TC2000 function cards, and its operation is manually controlled via the TCS console. STP consists of a user-level command interpreter, and a set of canned exercise procedures. In this sense, it is very similar to the board level diagnostic program in the processor diagnostics (Chapter 4). However, STP organizes all of the nodes in a system so that they can communicate among themselves, and so that they can perform various exercises simultaneously. This environment allows testing for:

1. A function card's ability to generate switch transactions to remote nodes.
2. A function card's ability to service switch transactions from remote nodes.
3. The switching network's ability to carry transactions between nodes.
4. A function card's ability to support intermixed transactions by the local CPU and the switch.

While other programs can perform a limited amount of this form of testing, STP is the most comprehensive multiprocessor testing program.

6.2 How to use the STP Diagnostics

The following section (6.3) has an example of how to run STP on a 7–processor machine using the TCS. It shows the results of one node failing. The steps are:

1. load and start STP via the TCS
2. initialize the STP master with parameters appropriate for that machine
3. bootstrap slave nodes
4. set up one or more nodes to run a test function
5. start the tests running
6. check for errors
7. display results
8. halt the tests

These steps are described in more detail in following the example.

6.2.1 Run Times

For an 8–node system, run STP for 0x80 passes. This takes about an hour. For a 16–node system, run STP for 0x400 passes (5 hours). For a 24–node system, run STP for 0x800 passes (10 hours). For a 32–node system, run STP for 0x1600 passes (20 hours).

The amount of time required for a system to complete a given number of passes will vary with the number of processors, the speed of the switch, and the availability of alternate paths. Large machines usually run passes faster than small machines. If you believe that you have a system that has faulty hardware, but does not fail in 30 minutes, run the test longer—one or two hours. In the worst case, let it run all night. The most convenient way to do timed tests is to set the slaves up with a pass count of 0, and abort the test after its time is up.

6.2.2 Basic Operation

STP sets up memory partitions on each processor and then writes data patterns into memory and verifies them. Essentially three types of errors may occur. These are either processors that cannot write specific data transfers, processors that cannot receive specific data transfers, or a switch that cannot process specific data transfers.

STP tests all the alternate paths, but it is difficult to isolate alternate switch path problems, if a failure occurs.

NOTE

Before running STP on any board or machine: be sure that the processor diagnostics have run to completion and run the Switch Connectivity test in the switch diagnostics to guarantee that all the cables and B2Loops are in place.

6.3 Running a Sample Session

Below is is an example of how to run STP on a 4–processor machine using the TCS. It shows the results of one node failing.

6.3.1 Loading and Starting

The STP diagnostic is started from the DOS or TEX prompt (see Chapter 3). From DOS type:

```
C:\TCS> CD \DIAG\STP ↲          Changes to the STP directory.
C:\DIAG\STP> TEX -A ↲           Automatically loads and starts STP.
TCS: Previewing machine state...
TCS: Number of clocks found = 1, disabled = 0.

                        more messages

Data of stp.88 loaded.
Node 7.7.7 has been loaded
Program entry point is 0x1c000.
TCS: Switching console terminal to application starting

*** STP - Sinister Test Program ***
    version 6.300, compiled Oct 13 1989.  Type ? for help
->
```

6.3.2 Initializing

After loading the diagnostic initialize the system:

```
-> stp i ↲                      Initialize using default parameters
Using POST alternate path randomizer
found 8 (decimal) processors in 1 midplane

Attention: this processor [7] will be master
Sinister Test Program  initialized master
```

6.3.3 Booting up Slaves

After initializing, boot up the slaves:

-> **stp b** ↲ Loads and starts the STP kernel on slave nodes.

```
booting slaves . . .
0x01    0x02    0x03    0x04    0x05    0x06    0x07
boot done
```

6.3.4 Setting Up Nodes

After booting, set up the nodes:

-> **stp s d 1 101 0** ↲ Setup a slave to run a certain test.

Setting up test 0xD: scattered write-verify

-> **stp v 1** ↲ Spread test parameters from one node throughout the system.

6.3.5 Starting the Tests

After setting up the nodes, start the tests:

-> **stp g** ↲ Start tests which have been previously set up.

6.3.6 Checking for Errors

After setting up the nodes, check for errors:

-> **stp f** ↲ Find processors with test faults.

```
total: 1 faults on 7 active nodes
checking watchdog counters..
done
```

6.3.7 Displaying Results

If you find an error, use the edify command to see a digested version of the results:

```
-> stp e 1 ↲                          1 is the node that failed above (normally shown)
---------- STP node 0x1               CURRENT pass in 0x42 ----------

destination node: 0x00                transaction: xmem word
fast path: ON      cache: OFF         short cut: ON      concurrency: ON

NODE HAS HAD 3 MEMORY VERIFY ERRORS
first error was at address 87EFA84    [node 1, local address 3EFA84]

expected value was:       0000A0AB
actual value was:         00001689
destination node: 0x01                transaction: halfword
fast path: ON      cache: OFF         short cut: OFF     concurrency: OFF
```

6.3.8 Halting the Tests

After displaying the results, halt the test:

-> **stp a** ↲ Stops a test in progress. Allows for new test to be set up.

```
 sending abort signals . . .
       0x01    0x02    0x03    0x04    0x05    0x06    0x07
 done
```

-> **stp u** ↲ Pushes all slaves into the microboot loop and uninitializes the master.

```
  Sinister Test Program deactivated
```

To get back to TEX type a backquote followed by a period (`.).

6.4 Introduction to "Scatter–Verify"

The example STP session used test 0xd ("scatter–verify"), which is the most important tool for stressing a multiprocessor system. Whenever you are attempting to qualify a system, this is the test you should run. The rest of this chapter often uses "STP" to mean the STP scatter–verify test.

STP's scatter-verify exercises many components at once. When something malfunctions, you must figure out which of the many possibilities caused the problem. STP catches and holds information when errors occur, and you must use this information to isolate the fault. Sometimes this requires sophisticated detective work.

6.4.1 Overview of Scatter-Verify

In essence, the scatter-verify test is a high-intensity memory exerciser. During operation it systematically writes and reads memory all over the machine. In the STP environment, there is a division in the memory of a processor. On every processor, the first one megabyte of memory is reserved for use by STP code, STP data, and memory management page tables. The remainder of memory (3 MB on the B2VME) is known as the STP "target zone". It is this region of memory that is energetically written and read in the scatter-verify test. This target zone is divided up into 4 KB slices. Each processor "owns" a number of slices on every processor in the system.

6.4.2 Basic Operation

The basic unit of STP operation is the "pass". A pass by one node consists of writing, then reading and verifying, all of its slices on a fixed destination node. The data transfers in a given pass are performed using a fixed type of bus transaction. For example, here is a snapshot of one node's activity during this test. The **edify** command provides information about what an STP slave is doing:

```
-> stp e 1 ↲          Displays a digested version of results for node 1.
------------    STP node 0x1        current pass = 0x20        ----------

destination node:  02              transaction:       byte
fastpath:          ON              code cache:        ON
switch short cut:  ON              88K concurrency:   ON
```

This informs us that:

- node 1 is on scatter-verify pass 0x20
- its current destination node is node 2
- it is using byte transactions

The number of passes to perform, as well as the kind of transactions to use are specified in the test set-up command:

-> **stp s d** *node type passes* ↲

The very first pass a slave makes is always directed at its own memory. Successive passes are made to successively higher numbered nodes. For example, node 1 makes its first pass in its own memory, its second pass in the memory of node 2, and third in node 3. When the highest node is done, the pattern starts again at node 0. The memory exercise is made into every node which has been booted with STP, regardless of whether that particular node is running the scatter-verify test itself. There is no way to restrict the test to operate on a subset of the booted nodes.

When the prescribed number of passes have been completed, the test stops. If the number specified was 0, the test will run forever unless it is stopped manually. The ordinary transaction types are listed below:

0	bytes
1	half words
2	full words
3	quad words
4	Xmem words
5	augmented locked full words

Thus,

-> **stp s d 1 3 100** ↵

Sets up test 0xD on node 1, using transaction type quad words, for 0x100 passes. And,

-> **stp s d 2 5 0** ↵

Sets up test 0xD on node 2, using type locked words, for an infinite number of passes.

NOTE

When a slave has been set up to run an infinite number of passes, it may be stopped at any time with the abort-test command (**stp a**).

6.4.3 Super Transaction Types

While this test provides the capability to restrict transaction types, it is normally desirable to use every kind possible. Towards this end, STP supports a pair of "super" transaction types. The first is type 0x100:

-> **stp s d 1 100 0** ↵

When type 0x100 is specified, the node running the test cycles through all the different transaction types. This is done by picking a new transaction type at the beginning of each pass. The entire pass is made with that type. For the very first pass, each processor chooses a transaction type as a function of its

node number. In successive passes, the types are used in the numerical order listed above.

In addition, another level of behavior modification is available. It is the second "super" type, number 0x101

```
-> stp s d 1 101 0 ↲
```

Looking again at the sample test results in Section 6.4.2:

```
-> stp e 1 ↲          Displays a digested version of results for node 1.
------------     STP node 0x1        current pass = 0x20        ----------

destination node:  02              transaction:       byte
fastpath:          ON              code cache:        ON
switch short cut:  ON              88K concurrency:   ON
```

Option 0x101 uses the same cycling transaction types as option 0x100 but also affects the use of the last four features listed in the **edify** command:

B2VME fast path enable
88K instruction cache enable
switch short cut enable
88K concurrency enable.

These are features STP is free to turn on or off as it likes. STP runs with any of them on or off, but it runs a little differently. Normally, they are all enabled, as shown in the status message above. However, when transaction type 101 is specified, the scatter–verify procedure regularly turns each of these features on and off. When the test is done, they are returned to their normal states.

6.5 Using and Analyzing STP

This section goes through a sample STP diagnostic in detail, including analysis of the test results. When running STP, you should follow the seven steps described in this Section.

6.5.1 Step 1: Load and Start STP via the TCS

The Sinister Test Program is stored on the hard disk of the TCS master. It is kept in the directory C:\DIAG\STP. This directory also contains the necessary TCS configuration files to initialize the system and load STP.

To load and start STP:

```
C:\TCS> CD \DIAG\STP ↲          Changes to the STP directory.
C:\DIAG\STP> TEX -A ↲           Starts the diagnostic.

.
.       more messages
.
starting

*** STP - Sinister Test Program ***
    version 6.300, compiled Oct 13, 1989.  Type ? for help
```

If your "start" does not get this far, something has malfunctioned. There are three most likely causes:

- The boot configuration file in \diag\stp is broken or missing.
- An executable file is missing (eg: uboot, post or stp)
- The system is broken in such a way that the TCS cannot initialize it.

You can check the first two using the TCS (see Chapter 2); for the last one call BBN ACI.

6.5.2 Step 2: Initialize STP Master Node Parameters

The node on which STP is first loaded serves as the STP "master". This node need not be the same as the nX OS master node. If circumstances require it, the BOOTCFG.TCS file can be set to load STP on any node in the machine.

You can set up to three system parameters during STP initialization. In most cases, however, you will not need to specify any of them. The following command is sufficient:

```
-> stp i ↲
Using POST alternate path randomizer
found 8(decimal) processors in 1 midplanes
 Attention: this processor [0] will be master
Sinister Test Program initialized
```

The complete form of this command is described later in Section 6.6.3.

6.5.3 Step 3: Bootstrap Slave Nodes

The objective of this phase is to get the STP kernel running on all the slave nodes in the machine. For each node, the STP master first checks that the slave is ready to accept an application program. It then copies itself onto the slave and tells the slave node to start. Once all of the slaves have been loaded, the master goes back and checks that each one has successfully completed initializing its STP environment. No explicit initialization command is needed for the slaves.

```
-> stp b ↵
booting slaves...
      0x01   0x02   0x03   0x04   0x05   0x06   0x07
boot done
```

This is the first place where problems can show up. If the switch is not functioning correctly, or if a slave node is seriously broken, the bootstrap may fail.

The Switch

A typical switch problem may produce error messages like:

```
booting slaves...          0x01   0x02
unexpected trap - Bus error ..........
 berr_vec ..........
 Data CMMU caused this trap
 Pipeline: ........
 Bus error cause: Sw Reject Timeout
 Data CMMU fault code: Bus Error
```

The problem here is that the master node went to read the memory of a slave node (node 2, in this case) and got a bus error. It may be due to a switch reject timeout, a switch wait timeout, a switch checksum error, etc. The key is that the cause is identified as a switch ("Sw") induced bus error of some sort.

An error like this can occur if STP attempts to boot a node which is not present. This should never happen when the automatic TEX start-up is used. However, if you something unusual (like turning a node off out from under STP), this is possible.

This error can also occur if the switch itself is not working. This might be because the switch hardware is broken, or because it is not being properly initialized (e.g. the switch frequency specified in **BOOTCFG.TCS** is wrong).

Lastly, this can occur when STP attempts to use alternate paths in a machine which does not have them. Use of these paths is controlled by the bootcfg.tcs file. If your machine does not have ALL of its alternate paths working, the configuration command controlling the use of alternate paths must disallow them.

Processors

When STP goes to boot a slave node, it has some assurance that the processor is at least partly working. It will have run both the micro-boot and power-on self test successfully (otherwise, the TCS deletes the node from the configuration list). It is possible, however, that a node getting this far still refuses to start STP properly.

Faults of this kind may still be indicated by bus errors during bootstrap. A bus error like "Downstream Late", or "Downstream OTL", or any other error not of "Sw" type is more likely indicative of a processor problem than a switch problem. The most effective way to differentiate between a switch error and a processor error is to swap processor nodes around. If the error travels with the board, then the switch is probably okay. If the error stays with the slot into which news boards are installed, then the switch is the likely culprit.

When STP attempts to boot a node, it makes three explicit checks on that slave's activity. First, before the bootstrap begins, the master verifies that the micro-boot state flag indicates that the board is "ready". This flag indicates that both **uboot** and **post** ran to completion and that the node is ready to receive an application. If this flag is in the wrong state, STP says:

```
slave 0x04 micro boot is not ready, state = s
```

Where s is the (wrong) state which was found. When you use the automatic TCS start-up, this error should not happen. Reaching the "ready" state is a prerequisite for TCS listing a node as present and useable. It is possible to get this error after STP has been shut down and then restarted. If an STP slave is busy before the master is initialized, it will be in the wrong state when the master is restarted. The master then complains. When this happens, just restart STP from scratch.

Once the STP kernel has been loaded on a slave, the processor is given a "go" signal. After this is done, the master again checks the node's state flag. This time it verifies that the flag has gone to state **6** ("gone"). If this does not happen, the master complains:

```
slave 0x4 has not started: micro-boot state= 5
```

This usually indicates a processor node malfunction.

The last bootstrap check made is for the completion of STP initialization. The master checks the STP state flag on every node to ensure that they are ready. If this does not happen, the master says:

```
giving up on slave 0x04 .........
```

This, usually, is also indicative of a processor node malfunction.

Master and Slave Nodes

The distinction between master and slave is somewhat arbitrary. The master node talks to the TCS console, and it is responsible for organizing slaves to run tests. But in actual execution of the tests, there is very little difference. The master node is capable of running all the same tests as the slaves. When the master node issues a system-wide start signal, it also sends a start to itself. It then enters "slave mode" and begins executing the very same code as the slave nodes. If no test has been set up on the master, it returns from slave mode to the STP command interpreter. Otherwise, it performs the indicated test function.

6.5.4 Step 4: Setting Up STP Tests on One or More Slaves

Once a slave node has been booted, it sits in an idle loop, waiting for instructions to do something. These "instructions" take the form of a test number and a list of arguments. To make a slave run the test, first perform an explicit set-up from the master. For example:

```
-> stp s d 1 101 200 ↵
```

This sets up test 0xD on node 1 with transaction type 101 for 0x200 passes. Test 0xD and test 0x17 are the primary tools for field use, and their semantics are the same.

You will usually not be using other test functions. If you do, keep in mind that the input parameter after the test number is ***always*** the node upon which the test is to run. It is ***not*** an argument to the test.

In a fully working system, the scatter-verify test should run indefinitely without errors. STP will detect a seriously malfunctioning processor node very quickly—in 0x40 passes or less. A board with subtle flaws may run for 0x1000 passes or more before failing. An overnight run without errors is a good indicator that the processor boards are functioning properly.

The test setup command prepares a test on only a single processor node. Obviously, doing a per-node setup in a large machine would be rather painful. The **viralize** command exists to handle the task of preparing an entire machine for a test run:

```
-> stp v 1 ↲
```

or

```
-> stp v 1 2 3 4 5 6 7 ↲
```

The first argument to **viralize** is the node whose setup information is to be spread. The remaining arguments are the nodes that are to receive the setup. If no list is given, all nodes except the master receive the setup.

If your slave nodes booted successfully, you should have no errors performing test setups. Both **setup** and **viralize** check to see that the node(s) upon which a setup is being attempted is not already busy doing something. No setup is performed on a slave that is busy.

Because running a test on the master limits the operator's ability to control the machine, the scatter-verify test is often run only on the slave nodes. In this case, the master will not be exercised as thoroughly as the slaves. ***To fully qualify the master node set the master up to run the test.*** To do this, you can use **viralize** to set up all the slaves to run forever, then issue a separate command to set up the master to run with a finite pass count. When the master is running test "do", it will announce its pass count periodically. The master can be stopped by striking any key on the console.

6.5.5 Step 5: Start the Tests Running

When a test setup has been received by a slave, it enters an "eager" state, waiting for a start signal. The test does not actually run until this signal is received.

The tests are started by either one of these commands:

```
-> stp g ↲
```

```
-> stp g 1 2 3 4 5 6 ↲
```

If a node list is given, those nodes receive a start signal. If no node list is given, a start signal is sent to every node. No harm is done in trying to start a node that has no test prepared, or one that is already running a test.

6.5.6 Step 6: Check for Errors

Each STP node keeps a small collection of memory locations for the results of the various functions it performs. There are 10 32-bit words reserved for this purpose. These words can be inspected by the STP **results** command, and the specialized **edify** command.

```
-> stp r 4 ↲
slave 04 completed test 0x3 (initialization)
results:     0      0      0      0
```

Identifying Results

Two words of the results are reserved for identifying the last test which was run, and whether that test completed normally. The **results** command decodes these two values and displays them in the first output line ("completed" = completed normally). The remaining eight words of results record various types of information, depending upon the test being run. The **results** command always shows the first words of results, but only displays the last four if one or more of them is non-zero. If the node has experienced a bus error, some information pertaining to that error is displayed.

Interpreting Results

Because the interpretation of these results varies widely with the test that is running, it is unwise for the inexperienced user to casually inspect them. For the purposes of interpreting the results from the scatter-verify test, use the special decoding function, STP **edify**.

```
-> stp e 6 ↲
 --------------- STP node 0x6        CURRENT pass is 0x301 ------------

destination node: 0x4                transaction: word
fast path: ON    cache: ON           short cut: ON   concurrency: ON
```

The results edified above are indicative of a node which has not had any errors. If STP encounters a problem of some sort, it will not report "cpu 5 is bad" or "midplane 0 is miscabled". In the course of this test, STP identifies only three explicit kinds of failures:

1. memory verify failure
2. bus error
3. processor failure

Memory Verify Failure: The first kind error occurs when the data from a memory read does not match the data which was previously written. When a node gets a memory verify failure, its edified results look like this:

```
--------------- STP node 0x6       CURRENT pass is 0x301 ------------

destination node: 0x4                transaction: byte
fast path: ON    cache: ON           short cut: ON   concurrency: ON

NODE HAS HAD 3 MEMORY VERIFY ERRORS
first error was at address 9512ABE     [node 5, local address 112ABE]

expected value was: 0000CC2E
actual value was:   00000001
destination node: 0x5                transaction: half word
fast path: ON    cache: ON           short cut: ON   concurrency: ON
```

When a verify error occurs, the STP node continues to run. The pertinent information for that error is saved. When subsequent errors occur, STP increments its error count, but no additional information is saved. In the above case, node 6 had three errors. The first happened when it was attempting to write half-words to node 5. The address 9512ABE is the STP global address at which the error occurred. The decoding of this global address into a node and a local address on that node is also given.

Bus Error: A bus error occurs when the hardware is fatally unable to service either a read or write request. When STP gets a bus error, the test *stops*. The results resemble:

```
--------------- STP node 0x6       FINAL pass was 0x85 ------------

destination node: 0x1                transaction: byte
fast path: ON    cache: ON           short cut: ON   concurrency: ON

NODE GOT A BUS ERROR at address  8505000    [node 01, local address 105000]
bus error type:  switch reject timeout
```

Note that the herald says "final pass".

Processor Failure: The last kind of STP error, a hung or failed processor node, does not show up on either a "results" or "edify" output. By definition, the node failed so badly that it could not record what happened. A hung node will appear to be running normally, except that its pass count never grows.

To help detect this situation, use the "fault finder" command:

```
-> stp f ↲
fault on processor 1
fault on processor 5
 total 2 faults on 7 active nodes
 checking watchdog counters....
 *** PROCESSOR 06 IS HUNG ***
done
```

This command checks for all three kinds of errors. A "fault" may be either a verify error or a bus error. The "watchdog" counters are numbers kept on each node; they are incremented several times during the course of a pass. This command checks that each node's counter is advancing. If the counter is not advancing, the tests assumes that the node is bad and indicates this.

In a typical field situation, you would use the **find faults** command to determine if any errors have occurred, and then use the **edify** command to get details from the nodes which have failed.

6.5.7 Step 7: Isolate the Faults

When you encounter a fault, isolate the board that caused it. The errors that occur when running STP almost always take place during bus cycles that are exercising the memory of some node. When a fault occurs, you are given two crucial pieces of information: the identity of the source node (what processor reports the fault), and the identity of the destination node (contained in the fault report). The specific type of fault encountered (i.e., what kind of bus error, what kind of memory transaction, which bits didn't verify) is also important. But the first priority is to establish not how the source processor failed, but to whom it was talking when the failure occurred. A failure can occur in one of three places:

1. the source node
2. the destination node
3. the switch nodes in between them

It will ***always*** be the source node that reports the error—even when the error occurs on the destination or in the switch. You must use the error information to determine which component is actually at fault. As a matter of practice, it has been observed that switch hardware rarely fails. When isolating a fault, concentrate first on the processor nodes involved.

For example, suppose you repeatedly run STP and node 1 fails like this:

```
---------- STP node 0x01        FINAL pass was 0x1 ---------------

destination node: 0x2           transaction: word
fast path: ON      cache: ON    short cut: ON      concurrency: OFF

NODE HAS HAD 2 MEMORY VERIFY ERRORS
first error was at address  8901000   [node 02, local address 101000]

expected value was:    89488C6DB
actual value was:      07AF00001
```

Node 1 is having trouble talking to node 2. The most direct way to isolate the failure in this situation is to swap boards. Put a known good board in place of either the source or destination node, and run the test again. If the failure goes away, the board you removed is probably defective. If the failure is still present, the other board is probably defective. To be ***certain*** of this second case, return the previously-removed board to its slot, and put the believed good board in place of the new suspect. The test should, of course, now run successfully; if it does not, then something else is malfunctioning.

Sometimes the fault data immediately suggests which node is failing:

```
---------- STP node 0x01        FINAL pass was 0x1 ---------------
 destination node: 0x2          transaction: word
fast path: ON      cache: ON    short cut: ON      concurrency: OFF

NODE GOT A BUS ERROR at address  8901000  [node 02, local address 101000]
bus error type: Downstream OTL

---------- STP node 0x03        FINAL pass was 0x9 ---------------

destination node: 0x2           transaction: word
fast path: ON      cache: ON    short cut: ON      concurrency: OFF

NODE GOT A BUS ERROR at address  8903000  [node 02, local address 103000]
bus error type: Downstream OTL


---------- STP node 0x04        FINAL pass was 0x8 ---------------

destination node: 0x2           transaction: word
fast path: ON      cache: ON    short cut: ON      concurrency: OFF

NODE GOT A BUS ERROR at address  8904000  [node 02, local address 104000]
bus error type: Downstream OTL
```

Notice that three different nodes report errors while conversing with node 2. Node 2 is almost certainly defective. This kind of failure is quite common—one node goes bad and causes errors in any processor that attempts to access it. Over time, every node in the machine touches that node and gets an error.

6.5.8 Dealing with Multiple Failures

The examples given in Section 6.5.7 are classic cases of a mostly-working system, with one or maybe two malfunctioning processor nodes. Inspection of the STP error messages gave a quick indication of where the problems were. A system with major problems may display many errors, scattered all over the place. In this situation, you must pare the machine down to some subset of processors that runs reliably. If necessary, remove or deconfigure processors until you have a stable base to work from. Then add in questionable nodes slowly. In some situations, you may find it helpful to run the scatter-verify test on only one node at a time, rather than on every node at once.

In a machine that is failing badly, it is sometimes helpful to run the scatter-verify test with a restricted transaction type. If a system has a lot of trouble with option 101, try using option 100. This exercises fewer parts of the system. You may also use one of the fixed transaction types (0, 1, 2, 3, 4, or 5). If you do use a restricted type, remember to re-test with option 101 before declaring a machine healthy.

6.5.9 Additional Considerations

The system activity provoked by STP is complicated and dynamic. Consequently, some errors are hard to characterize and isolate. Boards tested with STP should have passed single-node diagnostics. This means that any failures you see are caused by the interactions of two or more processors touching the same memory module. The failure modes in this case can be quite subtle.

An important point to remember is that altering the number of processors in a machine alters STP's traffic patterns. A machine with only half its complement of processors will not be stressed as heavily as it would be when fully configured. Also, removing a node may cause a fault occurring on one processor to re-appear somewhere else. In general, deconfiguring a node is ***not*** equivalent to replacing it. As a practical matter, you can isolate many faulty boards by taking them in and out of a machine's active processor set. But the final demonstration that a node is malfunctioning is made by substituting another processor in its place, and running the test without errors.

Be careful if you deconfigure a large number of boards in a machine which has more than one midplane. If all the boards in a midplane are disabled, STP reconfigures itself to act as though that midplane did not exist at all. For example, if boards 0 to 7 are pulled out of a 16-processor system, STP would run

as though it were in an 8 processor machine, and the 8 nodes will be numbered 0 to 7, **not** 8 to 15.

Keep in mind that if you run STP and get bus errors, memory verify errors, or hung processors, something basic is wrong. You should go back to a more specific diagnostic. STP may not always be helpful about isolating problems, but it does not raise false alarms.

6.6 STP Commands

This section contains a command summary, a list of frequently used commands, a more complete command reference, and a list of test functions.

6.6.1 Summary

This is a summary of the STP commands. The **stp** command and a space must precede each modifier. For a more detailed description, see Section 6.6.3. Note that for each command that follows only the first letter (in bold) needs to be typed.

abort	stops a test in progress. Allows for new test to be set up.
boot	loads and starts the STP kernel on slave nodes.
continue	frees a hung slave that has halted due to a bus error.
dump	displays a bunch of STP, processor, and SIGA state information. See also the **n**ode-list command.
edify	Displays a digested version of results from the "scatter-verify". system exerciser. See also the **r**esults command.
faults	Find processors with test faults.
go	start tests which have been previously set up.
initialize	initializes master node, enables alternate paths, and selects. memory mapping options.
microboot	return slaves to micro-boot wait loop.
node-list	Displays a summary of the processor node configuration.
poll	display a slave's state and test parameters.
Repeat	Sets up each slave with the parameters used in the previous test.
results	get results of the current (or previous) test.
setup	setup a slave to run a certain test.
tty	enable/disable TCS console output on slave nodes.
uninitialize	Pushes all slaves into the microboot loop and uninitializes master.
viralize	spread test parameters from one node throughout the system.
xyzzy	removes the setup of a slave.
zero	zeroes all STP exercise memory on every node.
?	Displays a list of the function numbers/function names available via "setup test".

6.6.2 Frequently-Used Commands

Remember to give the node number after commands that require it; otherwise you might get an error message.

stp a	Stops the scatter-verify test on all slaves. Also terminates bus error wait loops (slave goes to idle state).
stp e *node_list*	Provides English language details on the progress of the scatter verify test.
stp f	Identifies nodes with faults.
stp g	Starts the slave nodes running the test.
stp p	Polls the slave nodes running the test for status.
stp r *node_list*	Displays the raw results from any test.
stp s d 0 101 0	Prepares node 0 to run the scatter-verify test. A pass count of 0 makes the test run in an infinite loop.
stp v 0	Copies test setup from node 0 to the rest of the nodes in the system.

6.6.3 Command Reference

This is a summary of (most) of the STP functions available from the command line interpreter. All STP commands are of the form "stp *x y z*", where *x y z* are function specifics. Optional arguments are in square brackets "[]" .

NOTE

Commands cannot accept argument lists of more than 8 nodes. You must issue multiple commands to treat more nodes (e.g. **stp r 0 1 2 3 4 5 6 7** and **stp r 8 9**).

abort test

`stp a` [*node-list*]

Stops a test in progress. If no node specified, every non-idle processor receives an abort. The abort signal must be polled for: if the slave is running a test which does not check for it, the abort has no effect. Aborts terminate bus error wait loops (slave goes to idle state).

boot slave

`stp b` [*node–list*]

Loads and starts stp on the specified slave nodes. If no list is given, every node listed in the TCS config structure is booted.

continue after trap

`stp c` *node*

Pushes a slave out of a bus error wait loop and into the normal idle loop.

dump configuration information

`stp d`

Displays a bunch of STP, processor, and SIGA state information. See also the node list command.

edify test results

`stp e` *node–list*

Displays a digested version of results from the "scatter–verify" system exerciser. See also the results command. If no node list is specified, every eligible node is edified.

find faulted processors

`stp f`

Checks for fault indication on all booted slaves. A fault is typically indicative of a bus error or a memory verify mismatch. If some of the slaves are running the scatter–verify test, this function checks their watchdog counters to make sure that they are still running the test and are not wedged.

go (start)

`stp g` [*node–list*]

Sends a start–test signal to each node in the list. If no list specified, every booted node receives a start signal.

initialize stp master

`stp i` [*path–randomizer*] [*page–table–option*] [*midplane–override–flag midplane–number*]

Initializes the master node. This must be done before any other STP command. This command controls the settings of hardware functions, and the manner in which various parts of memory are mapped. If no arguments are specified, STP uses the settings already in place and uses the standard memory mapping scheme.

The *path randomizer* is a 3–bit quantity. A **1** in any bit position allows the SIGA to randomize that bit position in the "midplane number" in the switch route. If no *path randomizer* is specified, it defaults to **0**.

There are five *page–table options*, primarily controlling how local memory is mapped. Option **1** (all local) maps all of a processor's local memory using local addresses. Option **2** (part global) maps the first 1 MB of the local STP target zone as local and maps the upper 2 MB of target space using the global address for that node. Option **3** (part VMEbus), the first 2 MB are mapped as in option **2**, and the upper 1 MB is mapped as VMEbus space memory. Option **4** (interleaved) creates page tables for interleaving the entire 3 MB target zone. This option must be used in conjunction with the interleaver enable test (#0x13). Option **5** (all VME) maps all 3 MB of a node's local memory through the VMEbus mappers. If no option is specified, option **2** will default.

The *optional midplane* argument allows an 8 slot system to run as though it were on a different midplane than the one set in the hardware. This option is used by specifying a **1** in the override flag, and the desired midplane number in the last argument. This operation currently has a limitation: in order to run on a non–standard midplane number, it is first necessary to initialize on the standard midplane, boot all the slaves, then uninitialize.

put slave into micro–boot

`stp m` [*node–list*]

Makes the specified slaves exit STP and return to the micro–boot wait loop. If no list is specified, all booted nodes, except the master are affected.

node list

`stp n`

Displays a summary of the processor node configuration.

poll slave

`stp p` *node–list*

Displays current state of the slaves in the list, including the test parameter list. This is different from the information garnered from the **results** command. Be sure to give a node number.

query

`stp ?`

Displays a list of the function numbers/names available via "setup test".

repeat test

`stp R` *slave–list*

Sets up each slave with the parameters used in the previous test.

results of slave(s)

`stp r` *slave–list*

Displays the current state and the result words for each slave in the list. If the last 4 (of 8) result words are zero, they are not displayed. Extra information is given if the slave has a bus error. If no slave list is given, results are displayed for every node.

setup test

`stp s` *test–function–number* [*arguments*]

Set up an STP slave function. Be sure to give a test number (see Section 6.6.4).

slave tty inhibit on/off

`stp t` *slave* `0`

Changes the state of the "tty_inhibit" flag on a slave. Character output is normally suppressed on slave nodes. Clearing this flag by typing **stp t** *slave* **0** allows output to take place. To see this output, the TCS console must be instructed to connect to that slave.

uninitialize stp

`stp u` *[force-flag]*

Pushes all slaves into the micro-boot loop and unitializes the master. If the force flag is not specified, or is **0** (zero), the shutdown is aborted if the master finds a node that is not in the idle state. If the flag is **1** (one), shutdown continues even if a slave is not ready.

viralize

`stp v` *node*

Copies the test parameters from the specified node to every booted node in the system (except the master).

xyzzy (escape from test)

`stp x` *node*

Removes the set up test parameters on the specified node. Allows you to change your mind after setting up a test but ***before*** issuing the "go" command.

zero memory

`stp z`

Makes the master clear the target zone memory of every booted node.

6.6.4 STP Test Functions

The test functions now range from 0x4 to 0x17 (0x18 is listed, but empty). Here is a summary of the more useful ones:

"fast write"

`stp s 7` *slave-node source-node passes*

This function writes full words into the entire target memory of the destination node. The data written will be: upper 16 bits = upper 16 bits of RTC at start of test, lower 16 bits = source node (this is NOT a memory copy loop).

"fast read"

`stp s 9` *slave–node source–node passes*

Reads every full word in the source's target memory; nothing is done with the data read.

timing uniformity test

`stp s b` *node option*

Checks the regularity of the real time clock by running a series of timing loops. The options (**0**, **1**, **2** or **3**) select from a progressively longer series of tests. The shortest, **0**, runs about 40 short timing loops. For each loop, two data are displayed: the value of the RTC at the beginning of the loop, and the elapsed RTC time computed after the end of the loop. For a healthy RTC, all the elapsed times should be very nearly equal (within 1 microsecond). The first loop is usually slow since it must be loaded into the instruction cache.

With option **1**, the processor runs the short test above, then runs a long timing loop of one million iterations. This loop is a "calibration". The same timing loop is then run four more times, for 16 million iterations. The reported time is be 16 times the elapsed time for the calibration run. With option **2**, two additional timing runs are made, each of 16*16 (= 256) million loops. Option **3** is total: 1 run of 16*16*16 (= 4096) million loops [this takes 20–30 minutes].

"write–verify"

`stp s c` *source–slave–node destination–slave–node passes option*

Writes, then reads every word in the destination target memory. The data word starts as (RTC & 0xFFFF0000) and is incremented for each location. This is the behavior for the default option, **0**. With option **1**, the data word is starts at 0xPP000000, where PP = node number. With option **2**, no memory is written, but a verify pass is made. This pass expects to find the data left from option **1**. The result codes for this test are as follows:

0:	elapsed RTC time
1:	RTC at start of test
2:	RTC at end of test
4:	number of verify errors
5:	location of first verify error
6:	expected value at location
7:	actual value at location

"scattered write-verify"

`stp s d` *slave-node transaction-type passes*

Writes, then reads selected portions of memory on every node in a system. For the purpose of this test, memory on each node is "allocated" in 4 KB stripes. A stripe is reserved for each slot in the system, even if some slots are empty. Starting with itself, the source node writes, then reads all of its stripes on successive destination nodes. Writing and reading one node constitutes a "pass". There are several options for *transaction-type*:

0	bytes
1	half words
2	full words
3	quad words
4	Xmem words
5	augmented locked full words
100	cycle through all above types
101	cycle through all above types and other parameters

If **101** is specified, the transaction types cycle, but other system parameters will also change. In particular, the code cache gets turned off and on, the switch short cut is turned off and on, concurrence is turned off and on, and the fast path is turned off and on. The normal state of these parameters in STP are:

code cache:	on	fast path:	on
short cut:	on	concurrency:	on

If *passes* is **0** (zero), the test loops forever until stopped manually.

The results for this test are updated on every pass; their meaning are as follows:

2:	RTC start time
3:	last pass performed
4:	destination (upper word) + transaction type (medium low byte) + misc bits (low byte)
5:	misc bits of first verify error
6:	number of verify errors
7:	location of first error
8:	expected value at location
9:	actual value at location

Results 3 and 4 are updated on every pass. The misc bits control the cache, short cut and fast path.

Bit 0 = 0	short cut enabled
Bit 1 = 0	cache enabled
Bit 2 = 0	fast path DISabled
Bit 3 = 0	concurrency ON (serialize off)

xmem-locked atomic add

`stp s f` *source-slave-node destination-slave-node passes*

This is an atomic add function based upon an xmem lock. The source node uses the first two words in the target nodes memory: the first word is the lock, the second is the running total. The lock protocol is bit 31 = 0 (not locked) or 1 (locked). When the source node gets the lock, it adds one to the sum word, then releases the lock. Neither the lock nor the sum is initialized; they are used as they are found.

interleaver enable/disable

`stp s 13` *node option*

Option **1** (enabled), causes a slave node to initialize its interleaver modulus RAM, and to set the "interleaver enable" bit in the SIGA. Option **0** (disable) causes the SIGA interleaver enable to be cleared. This function is only useful with the interleaved page tables (see **stp i**). When the interleaver on a given node is enabled, all of the STP target memory in the system is accessed in an interleaved fashion. To make every node use interleaved access, the enable function must be run on each.

7

Peripheral Diagnostics

7.1 Overview

TC2000 Standalone Peripheral Diagnostics are designed to test the functional and operational integrity of both the TC2000 peripheral device controller boards and the peripheral devices themselves. The standalone diagnostics run without the support of an operating system. They are intended to verify that I/O devices and I/O controller boards on the TC2000 are connected properly and are in good working order.

They are also intended to assist field/sustaining/manufacturing engineering personnel in the diagnosis of failed devices down to a Field Replaceable Unit (FRU) level. The peripheral diagnostics attempt to localize failures as much as possible to the smallest possible component. In some instances though, the diagnostics will only be able to determine that an I/O device or I/O controller board has simply failed.

The peripheral diagnostics are designed to test and exercise the TC2000 peripheral subsystems only. No attempt is made to do any testing or diagnosing of processor node cards or other hardware.

7.2 Typical Symptoms

The typical symptoms are listed under each peripheral subsection.

7.3 How to use the Peripheral Diagnostics

After selecting a terminal type, the system will displays the main menu. Run the tests on each peripheral. If the device passes all of these tests, try the nX based peripheral tests or try Stress test. Verbose comments can be shut off

by selecting **Verbose Off** at the main menu. To exit from the diagnostics, type a backquote and a period (‘.). Do not use the **quit** menu item, it stops the test, but won't go back to TEX.

7.3.1 Run Times

The typical times are listed under each peripheral subsection.

7.4 TC2000 Peripheral Tests

The TC2000 peripherals consist of both Multibus/VMI[illegible] controller cards and the actual peripherals attached to these cards. Both [illegible]troller cards and peripherals are tested. All of the diagnostics for the peripherals will be downloaded to the appropriate B2VME from the TCS. The diagnostics are downloaded to the B2VME node connected to the VMEbus where they execute in that node's RAM.

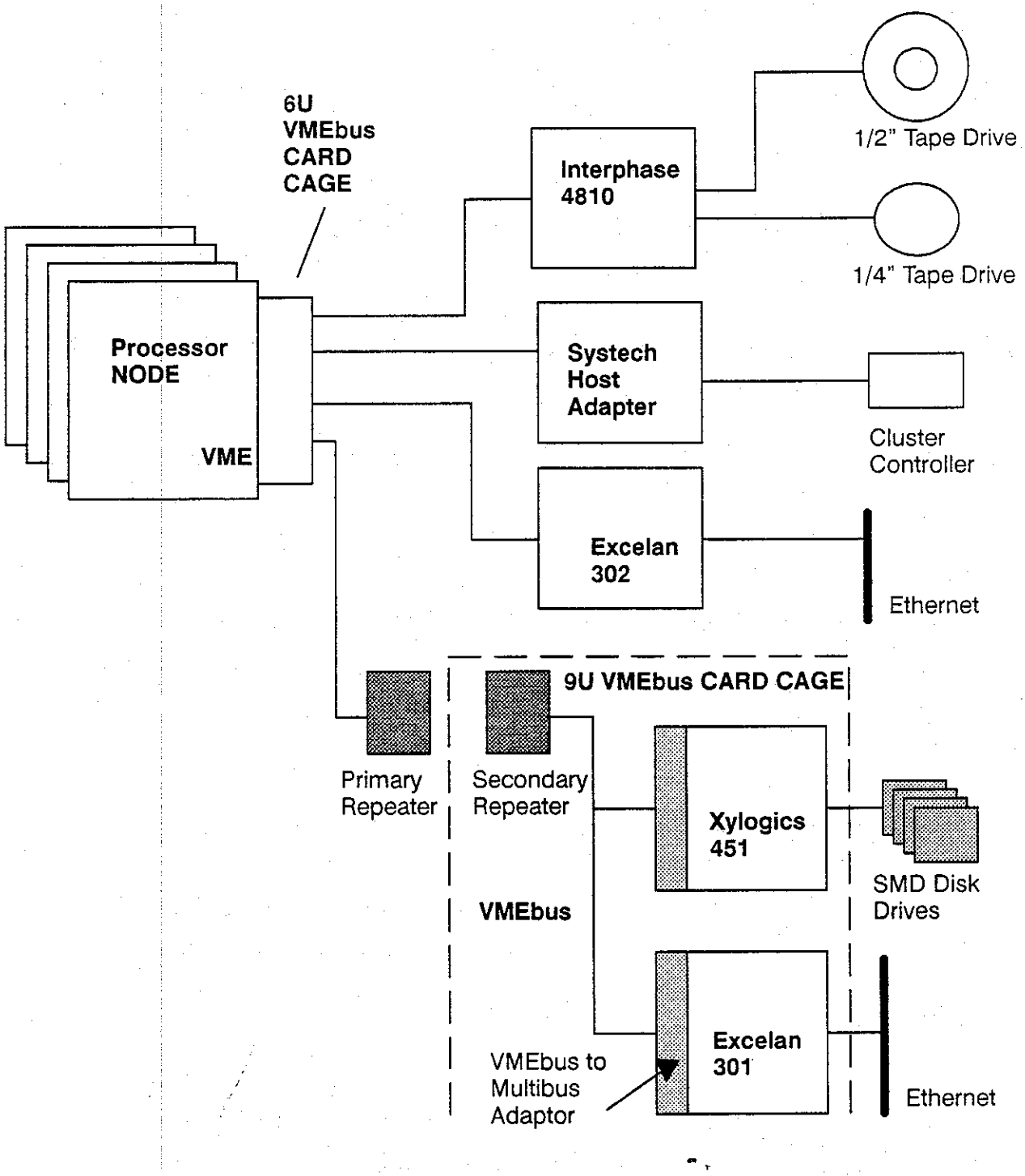
7.5 TC2000 Peripheral Controllers

The TC2000 I/O devices and controller boards are listed in Figure 7–1 and shown in a block diagram in Figure 7–2. Figure 7–2 includes all I/O interconnections and some devices not directly related to I/O. These devices are the Sun VME–to–Multibus Adaptor cards and the HVE Primary/Secondary VMEbus Repeater cards. These are transparent to VMEbus software.

Figure 7–1 TC2000 Peripheral Subsystems

1/4" SCSI Tape Subsystem	Interphase 4810 VMEbus SCSI controller Tandberg 3640 1/4" Tape Drive
1/2" Tape Drive Subsystem	Interphase 4810 VMEbus SCSI controller Cipher M990 1/2" Tape Drive
Terminal Controller Subsystem	Systech 6840 Host Adapter Systech 7088 Cluster Controller Various CRT's, Printers, etc.
Ethernet Interface Subsystem	Excelan 301 Ethernet Controller(Multibus) or Excelan 302 Ethernet Controller (VMEbus)
SMD Fixed Disk Subsystem	Xylogics 451 Disk Controller NT 8414 Disk

Figure 7–2 TC2000 Peripheral Set Block Diagram



7.6 Starting the Diagnostics

The peripheral diagnostics are started from the DOS or TEX prompt (see Chapter 3). From DOS type:

```
C:\> CD \DIAG\VMEDIAG ↲      Changes to the vmediag directory.
C:\DIAG\VMEDIAG> START ↲     Initializes the system.
```

When the TC2000 Standalone Diagnostics are downloaded to the processor node and started, a terminal type menu is displayed.

Select the appropriate terminal type or item **6** if you intend print the output. After making your selection, the main menu will be displayed.

7.7 Standalone Peripheral Diagnostics Main Menu

At this level, you can go to the peripheral submenu, start the automatic tests. change your terminal type, or change the verbose mode:

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS
    1. Peripheral-Diagnostics          TC2000, B2VME-B/TCFPV Nodes
    2. VME-bus-Diagnostics
    3. Automatic-Mode
    4. Terminal-Type                   Hardcopy, Menu-On
    5. Verbose-Mode-On-Off             On
    6. Help
    7. Quit
                                ver x.xx
STANDALONE-PERIPHERAL-DIAGNOSTICS -> 1
```

Entering a **1** at the menu above displays the peripheral diagnostics submenu:

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000)
    1. 1/4"-Tape-Subsystem             Controller
    2. 1/2"-Tape-Subsystem                  responds at      0x1000800
    3. Disk-Drive-Subsystem            Controller responds at 0x100ee40
    4. Multibus-Ethernet-Subsystem     Controller doesn't respond
    5. VMEbus-Ethernet-Subsystem       Controller responds at 0x1010000
    6. Terminal-Controller-Subsystem   Controller responds at 0x1020000
    7. System-Configuration
    8. Quit

1/4" & 1/2" Tape Drives share the same Controller
```

7.8 Automatic Mode

You can perform tests in automatic mode. Entering a **3** at the main menu displays the automatic mode dialog:

```
Function:  Automatic-Mode
   qrtr|disk|ethr|half|term|<string>  ->  all

--> Automatic-Mode

Enter qrtr|disk|ethr|half|term|all <string>  -> ↲

Do you wish to continue? (y or n) y↲
```

You can hit <Return> for all the tests, or enter one.

7.9 1/4" Tape Drive Tests

Entering a **1** displays the 1/4" Tape Drive Menu:

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000) 1/4"-Tape-Drive
    1. Power-On-Self-Test              Controller
    2. Configure-Controller-Test       Controller: *not* configured
    3. Extended-Self-Test              Controller
    4. Dump-Initialization-Params      Controller
    5. Interrupt-Test                  Controller
    6. Inquiry-Test                    Tape Drive
    7. Erase-Test                      Tape Drive
    8. Media-Removal-Test              Tape Drive
    9. Data-Buffer-Mem-Test            Tape Drive
    10. Low-Level-R/W-Test             Tape Drive
    11. Block-Address-Test             Tape Drive
    12. Rewind-Test                    Tape Drive
    13. Seek-Test                      Tape Drive
    14. Space-Test                     Tape Drive
    15. Filemark-Test                  Tape Drive
    16. First-Self-Test                Tape Drive
    17. Second-Self-Test               Tape Drive
    18. Quit

Jaguar    = Interphase 4210 SCSI Controller
TANDBERG  = SCSI Tape Drive
1/4"-Tape-Drive ->
```

Be sure to initialize the controller (menu item **2**) before running any of the tests.

7.9.1 Jaguar SCSI Controller Power Up Self-Test

Entering a **1** starts the Jaguar SCSI Controller Power-Up Self-Test. The screen displays the following messages, and ends with a prompt.

```
Jaguar SCSI Controller Power Up Self-Test.

Setting RESET bit on Jaguar board...
Clearing RESET bit on Jaguar board...
Absolute value of status register = 0x0002

<Board OK> BIT IS SET: POWER ON DIAGNOSTICS PASSED.
<Controller Not Available>  BIT IS CLEAR (OK).
CONTROLLER BOARD AVAILABLE FOR COMMANDS.

Attempting to Read CONTROLLER SPECIFIC SPACE:
Jaguar Product Code             = 077
Jaguar Product Variation Code   = 0
Jaguar Firmware Revision Level = XAC
Jaguar Firmware Release Date    = 03/21/1989
JAGUAR POWER ON SELF-TEST PASSED.
Power-On-Self-Test: Test Passed
```

7.9.2 Jaguar SCSI Controller Configuration Test

Entering a **2** starts the Jaguar SCSI Controller Configuration Test. The screen displays the following messages, and ends with a prompt:

```
Setting RESET bit on Jaguar board...
Clearing RESET bit on Jaguar board...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
Starting 'Initialize Work Queue' Command...

About to set 'Start Queue Mode' in Master Cntrl Register...
Current state of Cmd Rsp Blk status word = 0x0000
Command Response Block status word = 0x0021
CONTROLLER HAS BEEN SUCCESSFULLY INITIALIZED.
Configure-Controller-Test: Test Passed
```

7.9.3 Jaguar SCSI Controller Extended Self-Test

Typing a **3** executes the Jaguar SCSI controller extended self-test. This test is set up for executing on controller boards with revision 'XAJ' PROMs or later. (The system returns with the message "FAILED" otherwise.) To check the controller board revision, run the power-up self-test.

According to the Interphase/Jaguar documentation, the extended self-test results should all be '0xffff' to indicate that the test has passed. You may see a 0001 for the ROM test on an older board. You may also see the SCSI secondary port test returning a '0x0000', because there is no attached daughterboard.

```
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
ROM TEST RESULT                          = 0x0001        should be a 0xffff
SCRATCHPAD RAM TEST RESULT               = 0xffff
BUFFER RAM TEST RESULT                   = 0xffff
EVENT RAM TEST RESULT                    = 0xffff
SCSI PRIMARY PORT REGISTER TEST RESULT   = 0xffff
SCSI SECONDARY PORT REGISTER TEST RESULT = 0x0000

JAGUAR EXTENDED SELF TEST PASSED.
Extended-Self-Test: Test Passed
```

7.9.4 Jaguar SCSI Controller Dump Initialization Parameters Test

Typing a **4** executes the Jaguar SCSI dump initialization parameters test.

```
Executing 'Dump Initialization Parameters' Command first...
.
.       more messages.
.
Initialization Parameters as returned by Controller:
Number of command queue slots = 10
Primary SCSI Bus ID           = 8
Command Response Block Offset = 0x073c

Number of slots in Work Queue #1: 5
Work Queue #1 Priority Level    : 1
DUMP INITIALIZATION PARAMETERS TEST PASSED.
Dump-Initialization-Params: Test Passed
```

7.9.5 SCSI Interrupt Test

Typing a **5** executes the SCSI interrupt test.

```
SCSI Controller Interrupt Test

VMEbus INTERRUPT OCCURRED!
VMEbus INTERRUPT LEVEL = 3
VMEbus INTERRUPT VECTOR = 0x55
SCSI Controller returned NORMAL INTERRUPT VECTOR.
Number of Work Queue Zero Entries flushed = 0
Interrupt-Test: Test Passed
```

7.9.6 SCSI Inquiry (Revision Level) Test

Typing a **6** executes the SCSI inquiry test. This test requires that you insert a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) into the tape drive. Without a scratch tape, the test will fail.

```
Starting tape load...
                .       more messages.
                .

Inquiry Data Table:
  Peripheral Device Type code  = 1 (0x01)
  Device Type Qualifier        = 0 (0x00)
  ISO  Version (2 bits)        = 0x0
  ECMA Version (3 bits)        = 0x0
  ANSI Version (2 bits)        = 0x01
  Additional Length field      = 46
  Vendor Identification field  = >TANDBERG<
  Product Identification field = > TDC 3600<
  Product Revision Level       = >-04.<
  Option Level                 = >02<
  Software Identification      = >CREATED<
  Timestamp field:
    Month                      = >04
    Day                        = >22
    Year                       = >89
Starting tape unload...
                .
                .       more messages.
                .

Inquiry-Test: Test Passed
```

7.9.7 1/4" Tape Drive Erase Test

Typing a 7 executes the Erase test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive. Without a scratch tape, the test will fail. This test will take a few minutes while it erases the tape.

```
Test requires scratch tape to be loaded into tape drive.
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...

                .       more messages.

Polling of 'Command Complete' bit commencing...
ERASE TEST PASSED.
Erase-Test: Test Passed
```

7.9.8 1/4" Tape Drive Media Removal Command Test

Typing a **8** executes the Media Removal Command test. This test will attempt to blink the TANDBERG Drive Front Panel LED on and off 10 times by issuing 'Prevent/Allow Media Removal' commands.

```
Quarter-Inch Tape Drive Media Removal Test
This test will attempt to blink the TANDBERG Drive Front Panel
LED on and off 10 times by issuing 'Prevent/Allow Media Removal' commands.
Turning TANDBERG Front Panel LED on...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
Turning TANDBERG Front Panel LED off...

                .       more messages.

Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
```

7.9.9 1/4" Tape Drive Data Buffer Diagnostic Read/Write

Typing an **9** executes the Data Buffer Diagnostic Read/Write test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...

        .       more messages.
        .

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
R/W DIAGNOSTIC BUFFER TEST PASSED.
Data-Buffer-Mem.-Test: Test Passed
```

7.9.10 1/4" Tape Drive Low-Level Read/Write Test

Typing a **10** executes the Low-Level Read/Write test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...

Starting WRITE portion of R/W Test...

        .       more messages.
        .

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
R/W TEST PASSED.
Low-Level-R/W-Test: Test Passed
```

7.9.11 1/4" Tape Drive Request Current Block Address Test

Typing an **11** executes the Request Current Block Address test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
```

more messages.

```
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
Block Address returned by tape drive = 0x0000
REQUEST BLOCK ADDRESS COMMAND PASSED.
Block-Address-Test: Test Passed
```

7.9.12 1/4" Tape Drive Rewind Test

Typing a **12** executes the Rewind test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
```

more messages.

```
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
REWIND COMMAND PASSED.
Rewind-Test: Test Passed
```

7.9.13 1/4" Tape Drive Seek Test

Typing a **13** executes the Seek test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Starting tape load...

Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
.
.       more messages.
.
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
SEEK TEST PASSED.
Seek-Test: Test Passed
```

7.9.14 1/4" Tape Drive Space Test

Typing a **14** executes the Space test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

NOTE

This test has not been fully implemented. Do not run this test.

```
Starting tape load...
.
.       more messages.
.
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
SPACE TEST PASSED.
Space-Test: Test Passed
```

7.9.15 1/4" Tape Drive Write Filemark Test

Typing a **15** executes the Write Filemark test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

```
Test requires PRESENCE of scratch tape: 3M/600A or DEI/Series
II Gold tape or equivalent.
Starting tape load...
      .
      .     more messages.
      .
Controller Card has reset GO-BUSY bit, indicating it has
acknowledged receipt of command and is starting to process it.
Polling of CRBV bit commencing...
Polling of 'Command Complete' bit commencing...
WRITE FILEMARK TEST PASSED.
Filemark-Test: Test Passed
```

7.9.16 1/4" Tape Drive Self Test #1

Typing a **16** executes the Self Test #1 test. This test requires a scratch tape (3M/600A or DEI/Series II Gold tape or equivalent) to be loaded in the tape drive.

If the test passes it displays appropriate messages:

```
TANDBERG SELF TEST #1 PASSED.
First-Self-Test: Test Passed
```

7.9.17 1/4" Tape Drive Self Test #2

Typing a **17** executes the TANDBERG Self Test #2 test. This test writes data to the tape cartridge and requires the use of a scratch tape. TANDBERG Self Test #2 is a very stringent test allowing a very limited number of retries when exercising the read and write capabilities of the drive. For this reason, this test may fail on occasion if the tape is new or of poor quality, or if the tape has been overused.

For the first 40 seconds of this test the tape will not spin. After that time, the tape medium will be accessed and the tape will spin for 5–7 minutes worth of testing.

If the test passes it displays appropriate messages:

```
TANDBERG SELF TEST #2 PASSED.
Second-Self-Test: Test Passed
```

7.10 1/2" Tape Drive Diagnostics

The Jaguar/CIPHER Diagnostics Menu is displayed by selecting item **5** 1/2" Tape Drive from the Peripheral Diagnostics Menu shown in Section 7.7. Once that item is selected, the following screen is displayed:

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000) 1/2"-Tape-Drive
    1. Power-On-Self-Test              Controller
    2. Configure-Controller-Test       Controller: *not* configured
    3. Extended-Self-Test              Controller
    4. Dump-Initialization-Params      Controller
    5. Interrupt-Test                  Controller
    6. Mode-Select-Test                Drive *not* configured
    7. Inquiry-Test                    Tape Drive
    8. Test-Unit-Ready-Test            Tape Drive
    9. Erase-Test                      Tape Drive
    10. Rewind-Test                    Tape Drive
    11. Short-R/W-Test                 Tape Drive
    12. Long-R/W-Test                  Tape Drive
    13. Read-in-Reverse-Test           Tape Drive
    14. Filemark-Test                  Tape Drive
    15. Space-Records-Test             Tape Drive
    16. Diagnostic-Send-Receive-Test   Tape Drive
    17. Extended-Status-Test           Tape Drive
    18. Manual-Cipher-Self-Test        Tape Drive
    19. Quit

Jaguar    = Interphase 4210 SCSI Controller
CIPHER  = SCSI Tape Drive
1/2"-Tape-Drive ->
```

The controller tests (menu items 1–6) are the same as those under the 1/4" Tape Drive Diagnostics menu. Please refer to sections 7.9.1 through 7.9.5 for information on these tests.

Before running any of the controller tests (menu items 1–6), the Configure-Controller-Test must be run. Before running any of the tape drive tests (menu items 6–17), the Mode-Select-Test must be run.

NOTE

If any of the 1/2-inch tape (drive or controller) tests fail, every 1/2-inch tape test run after that fails. Re-run tests **2** and **6** to reset this condition.

7.10.1 Mode Select Test

Selecting item **6** from the 1/2" Tape Drive Diagnostic menu executes the Mode Select Test.

If the test passes it displays appropriate messages:

```
Polling of `Command Complete' bit commencing...
Mod-Select-Test:  Test Passed
```

7.10.2 Inquiry Test

Selecting item **7** from the 1/2" Tape Drive Diagnostic menu executes the Inquiry Test.

If the test passes it displays appropriate messages:

```
Model Name                = >M990-64K        <

Inquiry-Test:  Test Passed
```

7.10.3 Test Unit Ready Test

Selecting item **8** from the 1/2" Tape Drive Diagnostic menu executes the Test Unit Ready Test. The tape unit must be **on-line**.

If the test passes it displays appropriate messages:

```
Test-Unit-Ready-Test:  Test Passed
```

7.10.4 Erase Test

Selecting item **9** from the 1/2" Tape Drive Diagnostic menu executes the Fixed Erase Test. This is a variation of the Security Erase Test and can be used as a quick check of erase capability, as it erases a short length of tape, approximately 4 inches. Erasure begins from the current tape position. This test requires a scratch tape to be loaded in the tape drive. The tape unit must be **on-line**.

If the test passes it displays an appropriate message:

```
Erase-Test:  Test Passed
```

7.10.5 Rewind Test

Selecting item **10** from the 1/2" Tape Drive Diagnostic menu executes the Rewind Test. This verifies that the tape drive can rewind the tape. This test requires a scratch tape to be loaded in the tape drive. The tape unit must be **on-line**.

If the test passes it displays an appropriate message:

```
Rewind-Test:  Test Passed
```

7.10.6 Short Read/Write Test

Selecting item **11** from the 1/2" Tape Drive Diagnostic menu executes the Short Read/Write Test. This verifies that the tape drive can write data to and then read data back from the magnetic tape. This test performs a low-level write and read, comparing data written with data read (about 1K bytes). This test requires a scratch tape to be loaded in the tape drive. The tape unit must be **on-line**.

If the test passes it displays an appropriate message:

```
*** Starting Write Portion of R/W Test ***
*** Starting COMPARE of data WRITTEN with data READ back ***
Short-R/W-Test:  Test Passed
```

7.10.7 Long Read/Write Test

Selecting item **12** from the 1/2" Tape Drive Diagnostic menu executes the Long Read/Write Test. Data written is compared with data read. Data will be written (at both high and low speed) at 1600, 3200, and 6250 CPI. This test requires a scratch tape to be loaded in the tape drive.

If the test passes it displays an appropriate message:

```
Long-R/W-Test:  Test Passed
```

7.10.8 Read In Reverse Test

Selecting item **13** from the 1/2" Tape Drive Diagnostic menu executes the Read In Reverse Test. Data written is compared with data read. Data will be written at 1600, 3200, and 6250 CPI. This test requires a scratch tape to be loaded in the tape drive.

If the test passes it displays an appropriate message:

```
Read-in-Reverse-Test:  Test Passed
```

7.10.9 Filemark Test

Selecting item **14** from the 1/2" Tape Drive Diagnostic menu executes the Filemark Test. This verifies that the drive can distinguish file starting and ending points. The tape unit must be **on–line**.

Tape filemarks separate one file from another on the tape medium. The Write Filemark test commands the tape drive to write filemarks on the selected tape drive at the current tape position.

If the test passes it displays appropriate pass messages.

7.10.10 Space Records Test

Selecting item **15** from the 1/2" Tape Drive Diagnostic menu executes the Space Records Test. This verifies that the drive can space over filemarks in both directions and can space over data records in both directions.

If the test passes it displays appropriate pass message.

7.10.11 Diagnostic Send Receive Test

Selecting item **16** from the 1/2" Tape Drive Diagnostic menu executes the Diagnostic Send Receive Test. The tape unit must be **on–line**.

If the test passes it displays appropriate pass message.

7.10.12 Extended Status Test

Selecting item **17** from the 1/2" Tape Drive Diagnostic menu executes the Extended Status Test. This test requests more status information from the 1/2" tape drive. The drive then sends 64 bytes of status information, such as density and error history. The tape unit must be **on–line**.

7.10.13 Manual Cipher Self–Test

Selecting item **18** from the 1/2" Tape Drive Diagnostic menu displays the Cipher Self–Test instructions. The instructions are displayed on the screen. The self–test will test most of the electro–mechanical components.

This is a summary of the text on the screen and should be used as an aid to understanding. The numbers in the summary refer to keys on the front panel on the Cipher 1/2" tape drive and are numbered as follows:

1. "LOAD REWIND" switch

2. "UNLOAD" switch
3. "ON-LINE" switch
4. "WRT EN TEST" switch
5. "DENSITY SELECT" switch

Manual Test Summary

TEST 111 OSCILLATE SERVOS	Press 451115 to start Press 1 to start oscillation. Press 5 then 4 to end.
TEST 124 VOLTAGE DISPLAY	Press 451245 to start. Watch for "VOLTAGE". Press 4 to end.
TEST 131 FILE PROTECT AND REEL SEAT	Put tape in drive. Press 451315 to start Press 5 to end.
TEST 132 DOOR AND HUB LOCK	Press 451325 to start. Watch for drive to unlock or lock as you open or close door. Press 4 to end.
TEST 133 DOOR OPEN	Press 451335 to start. Watch for "DOOR OPEN" on display.
TEST 134 BLOWER MOTOR	Press 451345 to start. Press 1 to turn motor on and off. Press 4 to end.
TEST 212 READ/WRITE DATA	Put tape in drive. Press 1 and wait for tape to load. Press 452125 to start. Watch for "WRITING" on display. Press 5 to switch tape direction. Watch for "READ REV". Press 4 to end.
TEST 222 TAPE SHUTTLE	Press 452225 to start. Press 1, 2, or 3 to select the amount of time for the runs. Press 5 to start the selected test. Press 4 to end.
TEST 223 READ/WRITE TEST	Press 452235 to start. Press 4 to enter status mode, then press each of 1,2, then 3 to display the various status messages (each button will display one message). Press 5 to end.

7.11 Disk Drive Diagnostics

The Xylogics 451/SMD-Interface Disk Diagnostics menu is displayed by selecting item 2 from the Peripheral Diagnostics Menu shown in Section 7.9.

NOTE

Notice the message next to item 6 that indicates the drive is not initialized. To execute any disk test from six through ten, you must first initialize with item 6 (Initialize-Disk-Type).

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000) Hard-Disk-Drive
    1. Xylogics-Self-Test              Controller
    2. Register-Test                   Controller
    3. DMA-Test                        Controller
    4. Chained-Command-Test            Controller
    5. Interrupt-Test                  Controller
    6. Initialize-Disk-Type            Disk Drive: *not* initialized
    7. Reset-Test                      Disk Drive
    8. Read-Status-Test                Disk Drive
    9. Short-R/W-Test                  Disk Drive
   10. Long-R/W-Test                   Disk Drive
   11. Seek-Test                       Disk Drive
   12. Quit

[Allowable Drives:  (#Heads  #Cylinders  #Sectors)
 NT 8408 (500)          8       1496        65
 NT 8414 (850)         14       1511        65

Current drive is set to: None
Unit number: None
```

All subsequent menus, except those for the Short and Long Read/Write Tests, will suggest that the WRITE PROTECT switch be set on with the following message:

```
ALTHOUGH THIS DIAGNOSTIC SHOULD NOT DESTROY DISK DRIVE
DATA UNDER NORMAL CONDITIONS, IT IS RECOMMENDED THAT ANY
ATTACHED DISK DRIVES HAVE THEIR **WRITE PROTECT** SWITCHES
ENABLED (ON) WHILE RUNNING THIS TEST.
```

CAUTION

The Short and Long Read/Write tests destroy disk data. Neither should be used on a disk containing data you wish to keep.

7.11.1 Disk Controller Self-Test

Selecting item **1** from the Disk Drive Diagnostic menu executes the Xylogics Self-Test. This verifies that the Xylogics 451 Controller passes the same self-test that occurs on power up; it does not read, write, or otherwise affect the disk(s). Note that the WRITE PROTECT switch on the disk drive should be ON.

If the test passes it displays appropriate messages:

```
Command issued to controller passed
DISK CONTROLLER SELF-TEST PASSED
```

7.11.2 Disk Controller Register Test

Selecting item **2** from the Disk Drive Diagnostic menu executes the Register Test. This verifies that the 451 Controller responds properly to writing and reading a walking-ones pattern through each of the six internal registers. Note that the WRITE PROTECT switch on the disk drive should be ON.

If the test passes it displays appropriate messages:

```
CONTROLLER REGISTER TEST PASSED.
CONTROL/STATUS REGISTER PORTION OF
DISK CONTROLLER REGISTER TEST PASSED.
FIXED DISK CONTROLLER REGISTER TEST PASSED.
```

7.11.3 Disk Controller DMA Test

Selecting item **3** from the Disk Drive Diagnostic menu executes the DMA Test. This verifies that the 451 Xylogics Controller can successfully DMA to and from system (TC2000) memory. Note that the WRITE PROTECT switch on the disk drive should be on.

This function must work properly before any reading or writing to disk can take place. If disk drive read/write tests fail, this test can be used to determine whether it's the controller's fault and whether or not the controller can perform even a minimum communication with the host.

If the test passes it displays an appropriate message:

```
CONTROLLER DMA TEST PASSED
```

7.11.4 Chained Command Test

Selecting item **4** from the Disk Drive Diagnostic menu executes the Chained Command Test. This verifies that the disk controller is able to execute several disk controller command (IOPB's) each of which is linked to the next com-

mand in a list. This tests writes data from system memory into a controller FIFO and back then compares it against the original data. This disk is **not** accessed during this test. Note that the WRITE PROTECT switch on the disk drive should be ON.

If the test passes it displays an appropriate message:

```
Command issued to controller passed
FIFO DATA GOOD...
CONTROLLER BUFFER LOAD/DUMP TEST PASSED
```

7.11.5 Interrupt Test

Typing a **5** executes the disk drive interrupt test.

If the test passes it displays an appropriate message:

```
Interrupt-Test: Test Passed
```

7.11.6 Initialize Disk Type

Selecting item **6** from the Disk Drive Diagnostic menu initializes the disk type for the diagnostics.

The following screen is displayed and you are prompted for the Unit Number and Drive Type. If the defaults are appropriate, just press <Return>.

```
Function: Initialize-Disk-Type
Initialize Disk Type.
    Drive-Unit#              <Number>          0
    Drive-Type               <String>          NT-8414
-> Initialize-Disk-Type
Enter Drive-Unit# <Number> -> (0)↵

-> Initialize-Disk-Type 0
Enter Drive-Type <String> -> (NT-8414)↵

-> Initialize-Disk-Type 0 NT-8414
Resetting Disk Controller Board
Checking Control/Status Register...
```

If the test passes it displays an appropriate message:

```
Command issued to controller passed
SET DRIVE SIZE COMMAND PASSED
```

7.11.7 Reset Test

Selecting item 7 from the Disk Drive Diagnostic menu executes the Reset Test. This verifies that the ability of the Disk Drive under test to recalibrate its disk heads and seek them to the zero position. Note that the WRITE PROTECT switch on the disk drive should be ON.

If the test passes it displays an appropriate message:

```
DRIVE RESET TEST PASSED
```

7.11.8 Read Status Test

Selecting item **8** from the Disk Drive Diagnostic menu executes the Read Status Test. This test reports on the disk drive status (ONLINE, READY, etc.) and the disk label parameters (disk drive ID, maximum number of sectors, maximum number of cylinders, etc.). This test does not write to the disk drive. The disk drive must be formatted. Note that the WRITE PROTECT switch on the disk drive should be ON.

```
Disk Drive Read Status/Label Test.

Resetting Disk Controller Board
Checking Control/Status Register...
Control/Status register, after reset = 0x09
Starting execution of controller board command...

Execution of I/O Parameter Block (IOPB) follows:
Disk Controller Control/Status Register = 0x09
Absolute address of IOPB                = 0x4d9b0
Idle count                              = 1116

Command issued to controller passed
Control/Status register = 0x09
iopbptr->byte[STAT1]    = 0x05
iopbptr->byte[STAT2]    = 0x00
Read Drive Status test passed.
Results from status command:
Drive Status byte: 0x00
    .
    .
    .
Sector offset of start of partition #[08] = 0
Number of sectors in partition      #[08] = 0
```

7.11.9 Short Read/Write Test

Selecting item **9** from the Disk Drive Diagnostic menu executes the Short Read/Write Test. This verifies that the disk drive can write data to and then read data back from the disk platter. Only a few sectors are used. The data read back will be compared with the data written. Note that for this test, the WRITE PROTECT switch must be OFF. If you leave the WRITE PROTECT switch on by mistake, this test fails with a "write protect error". To run this test, you must use a **formatted** disk.

CAUTION

This test destroys data on the disk. Do not use it on a disk containing information you want to retain.

If the test passes it displays an appropriate message:

```
Starting compare of READ data with WRITE data
DISK READ/WRITE TEST (SHORT VERSION) PASSED
```

7.11.10 Long Read/Write Test

Selecting item **10** from the Disk Drive Diagnostic menu executes the Long Read/Write Test. This verifies that the disk drive can write data to and then read data back from the disk platter. This test is more stressful on the drive, the drive interface, and the VMEbus than the short version by attempting to read and write almost every sector on the disk, similar to the disk formatter utility. Disk data will be lost. This test takes about 8 minutes. To run this test, you must use a **formatted** disk.

CAUTION

This test destroys data on the disk. Do not use it on a disk containing information you want to retain.

If the test passes it displays appropriate messages:

```
Command issued to controller passed
'READ LABEL' WAS SUCCESSFUL...
DISK DRIVE LONG READ/WRITE TEST PASSED
```

7.11.11 Seek Test

Selecting item **11** from the Disk Drive Diagnostic menu executes the Disk Drive Seek Test. This verifies that the disk drive can perform explicit seek commands. If the disk drive fails the read/write test, this test can be used to check a more basic function. Inherent in every read and write is a hidden seek. The diagnostics issues explicit seeks to complement the read/write tests.

If the test passes it displays appropriate messages:

```
Command issued to controller passed
'READ LABEL' WAS SUCCESSFUL...
SEEK TEST PASSED
```

7.12 Multibus Ethernet Controller Diagnostics

The Multibus Ethernet Controller Menu is displayed by selecting item **4** 'Multibus-Ethernet-Controller' from the Peripheral Diagnostics Menu shown in Section 7.7. Once that item is selected, the Ethernet Controller Diagnostics Menu is displayed. The Ethernet card must be plugged into a transceiver for the diagnostics to pass, otherwise the self-test will fail its loopback test.

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000) Multibus-Ethernet-Cntrllr
    1. Self-Test
    2. Link-Level-Mode-Test
    3. Quit

Excelan Multibus 301 Card

Ethernet-Controller ->
```

7.12.1 Ethernet Controller Self-Test

Selecting item **1** from the Ethernet Controller Diagnostic menu executes the Excelan 301 Self-Test. This verifies that the EXOS 301 Intelligent Ethernet Controller is functioning properly. The communication capability over the Ethernet itself is not tested.

If the test passes it displays an appropriate message:

```
EXCELAN: SELF TEST PASSED.
```

7.12.2 Link Level Mode Test

Selecting item **2** from the Ethernet Controller Diagnostic menu executes the Link Level Mode Test. This verifies that the host computer can download small packets of commands to the Excelan controller in link-level mode to test sending and receiving data. The test does NOT exercise the Ethernet itself.

If the test passes it displays an appropriate message:

```
Link-Level-Mode-Test:  Test Passed
```

7.13 VMEbus Ethernet Controller Diagnostics

The Ethernet Controller Diagnostics Menu is displayed by selecting item **5** 'Ethernet-Controller' from the Peripheral Diagnostics Menu shown in Section 7.7. Once that item is selected, the Ethernet Controller Diagnostics Menu is displayed. The Ethernet card must be plugged into a transceiver for the diagnostics to pass, otherwise the self-test will fail its loopback test.

```
Menu:  STANDALONE-PERIPHERAL-DIAGNOSTICS (TC2000) VMEbus-Ethernet-Cntrllr
    1. Self-Test
    2. Link-Level-Mode-Test
    3. Quit

Excelan VMEbus 302 Card

Ethernet-Controller ->
```

7.13.1 Ethernet Controller Self-Test

Selecting item **1** from the Ethernet Controller Diagnostic menu executes the Excelan 302 Self-Test. This verifies that the EXOS 302 Intelligent Ethernet Controller is functioning properly. The communication capability over the Ethernet itself is not tested.

If the test passes it displays an appropriate message:

```
EXCELAN: SELF TEST PASSED.
```

7.13.2 Link Level Mode Test

NOTE

If the system contains the old Systech Controller (mutlibus), the test will run as noted, else the test may fail.

Selecting item **2** from the Ethernet Controller Diagnostic menu executes the Link Level Mode Test. This verifies that the host computer can download small packets of commands to the Excelan controller in link-level mode to test sending and receiving data. The test does NOT exercise the Ethernet itself.

If the test passes it displays an appropriate message:

```
Link-Level-Mode-Test:  Test Passed
```

7.14 Terminal Driver Diagnostics

The Terminal Driver Diagnostics Menu is displayed by selecting item 4 from the Peripheral Diagnostics Menu shown in Section 7.7. Once that item is selected, the following screen is displayed:

```
Menu:   STANDALONE PERIPHERAL DIAGNOSTICS  (TC2000) CRT-Terminal-Controller
    1. Handshake-HA-Self-Test           Host Adapter
    2. HA-RAM-Test                      Host Adapter
    3. HA-LED-Test                      Host Adapter
    4. Interrupt-Test                   Host Adapter
    5. Get-Error-Log-Table              Host Adapter
    6. Get-System-Address-Table         Host Adapter
    7. Quit

HA = Systech HPS 6840 Host Adapter
CC = Systech Cluster Controller

Terminal ->
```

7.14.1 Handshake Host Adapter Self-Test

Selecting item **1** from the Terminal Controller Diagnostic menu executes the Handshake Systech Host Adapter Self-Test. The sixteen self-tests are comprehensive and are intended to catch and diagnose almost all errors short of catastrophic ones such as a no clock or no power to the Host Adapter board. In addition to executing the previous self-test this test performs a "handshake" with the host computer to check the Multibus data bus interface. The test also performs a DATA LOOPBACK TEST and an INTERRUPT LOOPBACK TEST.

Check the jumpers with the instructions for correct position. For only this test to successfully complete, the E22 jumper pins **3** and **4** must be **out**.

If the test passes it displays an appropriate message:

HOST ADAPTER HANDSHAKE VERSION OF SELF TEST PASSED.

7.14.2 Host Adapter RAM Test

Selecting item **2** from the Terminal Controller Diagnostic menu executes the Systech Host Adapter RAM Test. This verifies that the Multibus-accessible RAM on the Adapter board (96 bytes worth) is present and working. Pattern and fill RAM tests are used.

If the test passes it displays an appropriate message:

```
Pass Message...
```

7.14.3 Host Adapter LED Test

Selecting item **3** from the Terminal Controller Diagnostic menu executes the Systech Host Adapter Self-Test. The manufacturer's self-tests do not provide an LED test for the Host Adapter. This diagnostic will exercise the on-board tri-color LED, which is used during power-up and self-test. This diagnostic will cause the Host Adapter LED to flash.

When indicated by the test, check the LED on the Host Adapter(terminal controller). There are no pass or failure messages displayed.

7.14.4 Interrupt Test

Typing a **4** executes the disk drive interrupt test. The TC2000 has the controller card generate an interrupt. This exercises the interrupt cycle with the controller.

If the test passes it displays an appropriate message:

```
Interrupt-Test: Test Passed
```

7.14.5 Get Error Log Table

Selecting item **5** from the Terminal Controller Diagnostic menu executes the Get Error Log Table function. This function displays a table of board information and network statistics.

7.14.6 Get System Address Table

Selecting item 7 from the Terminal Controller Diagnostic menu executes the Get System Address Table function. This function displays a list of vector and routing table addresses.

8

nX Based Peripheral Diagnostics (periph_diag)

8.1 Overview

Unlike the standalone peripherals test described in Chapter 7, the nX based diagnostics test (**periph_diag**) is a separate program that runs from the nX operating system. **periph_diag** exercises the major functional operations of each peripheral device, such as reading and writing blocks of data to tapes and disks, formatting tapes and disks, erasing and rewinding tapes, and sending data over serial port and network connections. Refer to Figure 7-1 for a block diagram of the system peripherals and their connections.

Because **periph_diag** does more substantial testing of each component than the standalone peripherals diagnostics and transfers large amounts of data back and forth, the **periph_diag** tests are correspondingly longer, sometimes up to one hour each.

Like the standalone peripherals test, **periph_diag** is designed to assist in the diagnosis of failed devices. Failures are localized to the smallest possible component. (In some cases the tests may only determine that a device or controller board has failed.) Information from these tests enables field service personnel to pinpoint the necessary Field Replaceable Unit (FRU) and replace it.

In addition to performing substantive testing of the peripherals, **periph_diag** checks that all peripheral-to-system connections are operational by means of its system configuration test.

8.2 Typical Symptoms

If a test fails, the error message is often obvious. If the error message is obvious (like 'tape door open' or 'is tape loaded?'), fix the problem and re-run the test.

If the message is not obvious, follow the instructions in the section describing the test. If these approaches fail, call BBN ACI for assistance.

The typical symptoms are listed under each peripheral subsection.

8.3 How to Use the periph_diag Diagnostics

After selecting a terminal type, the system will displays the main menu. Run the tests on each peripheral. If you have problems, but the device passes all of these test, try Stress test and call BBN ACI. To exit from the diagnostics, select the **Quit** menu item. This brings up the nX OS prompt.

8.3.1 Run Times

The run times are listed under each peripheral subsection.

8.4 TC2000 Peripherals

The TC2000 peripherals consist of controller cards and the actual devices attached to these cards. Both controller cards and peripherals are tested. Refer to Figure 8–1 for the list of peripherals tested by **periph_diag**. Refer to Figure 7–1 for a block diagram of these devices and controller cards.

Figure 8–1 TC2000 Peripheral Subsystems

1/4" SCSI Tape Subsystem	Interphase 4810 VMEbus SCSI controller Tandberg 3640 1/4" Tape Drive
1/2" Tape Drive Subsystem	Interphase 4810 VMEbus SCSI controller Cipher M990 1/2" Tape Drive
Terminal Controller Subsystem	Systech 6840 Host Adapter Systech 7088 Cluster Controller Various CRT's, Printers, etc.
Ethernet Interface Subsystem	Excelan 302 Ethernet Controller (VMEbus)
SMD Fixed Disk Subsystem	Xylogics 451 Disk Controller NT 8414 Disk

8.5 Starting the Tests

The following conditions are required for running **periph_diag:**

- The nX operating system must be running.
- TCS must be running (refer to Chapter 2 for more information).

If the nX OS has been booted successfully on your TC2000, you may run the nX based diagnostics. If you are unable to boot the nX OS, you must run the standalone diagnostics first to determine if a peripheral is preventing successful boot up of the operating system. Chapter 7 describes the procedure for running standalone diagnostics.

Start the nX based diagnostics by completing the following steps at the system prompt:

```
nX (challenger.bbn.com) (console)

login: root ↲                                   Log in to the system as superuser ("root").
Password: password ↲                            Enter password for "root".
Last login: Thu May  4 18:29:15 on console


*******************************************************************************
For systems with post-sales hardware/software support, the
BBN Butterfly Hotline number is:
        1-800-4AC-BFLY          (in US)
        1-617-873-8660          (outside US)
*******************************************************************************
# cd /stand ↲                                   Change to the appropriate directory.
# ./periph_diag ↲                               Execute the diagnostics.
```

When the **periph_diag** Diagnostics have started, a terminal type menu is displayed.

Select the appropriate terminal type or item **6** if you intend print the output. After making your selection, the main menu will be displayed.

8.6 periph_diag Main Menu

At this level, you can go to the peripheral submenu, start the automatic tests, change your terminal type, or change the verbose mode:

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS
    1. Peripheral-Diagnostics          TC2000, nX based
    2. Automatic-Mode
    3. Terminal-Type                   Hardcopy, Menu-On
    4. Verbose-Mode-On-Off             On
    5. Help
    6. Quit
                                ver x.xx
nX BASED-PERIPHERAL-DIAGNOSTICS -> 1
```

Entering a **1** at the menu above displays the peripheral diagnostics submenu:

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS (TC2000)
    1. 1/4"SCSI-Tape-Cartridge
    2. Hard-Disk-Drive                 SMD only, not SCSI
    3. Ethernet-Controller
    4. CRT-Terminal-Driver
    5. 1/2"-Tape-Drive
    6. System-configuration
    7. Quit

(TC2000) ->
```

You can test the peripherals in any order. In the following sections, the tests are described in the order they appear in the menus.

The System configuration menu item displays the current state of the peripherals and some peripheral specific information.

8.7 1/4" Tape Drive Tests

The 1/4" tape drive diagnostic has eight tests, as shown in the menu:

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS (TC2000) 1/4"-Tape-Drive
    1. Raw-Device-R/W-Test
    2. Block-Device-R/W-Test
    3. Short-Tar-Test
    4. Long-Tar-Test
    5. Convert-&-Copy-Test
    6. Retension-Test
    7. Write-Protect-Test
    8. Erase-Rewind-Test
    9. Quit

Jaguar    = Interphase 4210 SCSI Controller
TANDBERG  = SCSI Tape Drive

1/4"-Tape-Drive ->
```

To run the tests, type the appropriate number followed by a <Return>. The following subsections display the results of a successful run.

Error Message Interpretation

Possible non-obvious error messages are described in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI.

If Kernel Driver error message:

1. Verify that the tape cable is not plugged in backwards; if it is, reorient cable and re-run the test.
2. Replace cable and re-run test.
3. Open tape door, close, verify that tape winds to the beginning, re-run the test.
4. Halt the nX OS and run the standalone tests on the tape drive; if they fail, replace drive and/or Controller board; if they pass, reboot the nX OS and re-run tests; if the nX OS test still fails, call BBN ACI.

If Open/Write/Read error:

1. Verify that the tape cable is not plugged in backwards; if it is, reorient cable and re-run test.
2. Make sure that the device (e.g., 'rsmt1') exists in the /**dev** directory and is of the proper type (char, block, etc.); if it doesn't exist, use MAKEDEV to create it and reboot the machine.
3. If test still fails, try opening another device by running the 'system configuration' test; if this fails on all devices, there is a either an nX problem (call BBN ACI) or an HVE repeater problem: replace both repeaters and re-run tests (after halting nX).
4. If system configuration passes on all devices but the SCSI drive, replace the cable first, then the tape drive.

If Data miscompare error:

1. Try a new tape; tape must be of type 3M/600A , 3M/600 XTD or DEI Series II Gold (***not*** 'Gold Plus') or equivalent.
2. If 1) doesn't fix the problem, clean tape heads and re-run test.
3. If 1) and 2) don't fix the problem, replace tape cable and re-run test.
4. If all else fails, replace tape drive.

If Kernel panic:

1. Do crashdump analysis (see Section 3), if possible.
2. Run standalone diagnostics; replace controller/tape drive if either fail.
3. Reboot the nX OS; verify that the correct version of the nX OS is installed.
4. Reboot the nX OS and re-run test; if panic re-occurs, first try a different king node, then a different tape drive. If it repeats, Call BBN ACI.

If timeout/hang error:

1. Quit out of the diagnostics (<Control>C). Type 'reset' at the console to restore the console to its original state.
2. Type **ps** to find the PID of the running process; use 'kill' to kill it. If it remains alive, halt the nX OS; else re-run the test.
3. If test still hangs or times out: run standalone diagnostics; if they pass, reboot the nX OS and retry the nX OS tests.
4. If test still hangs/times out, try a different king node, and if it still fails, call BBN ACI.

Failures

Possible non-obvious error messages are described in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI. Many of the error messages are the same as for Block-Device-Read/Write and Long-tar tests (Subsections 8.7.2 and NO TAG), with the following differences:

1. If you forgot to enable the SAFE write-protect mechanism, do so and re-run the test.
2. If the test still fails, try a new tape.
3. If the test still fails, replace the tape drive cable, then the tape drive, if necessary.

Failures

Possible non-obvious error messages are described in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI. Many of the error messages are the same as for Block-Device-Read/Write tests (Section 8.7.2), with the following differences:

1. First, quit out of the diagnostics on any failures and type the following at the shell level prompt: **tar cvf /dev/***xxxx* <*filename*>, where *xxxx* is the name of the drive and <*filename*> is a file of your choice. Note the pass/fail status of this shell command. If this shell level command passes, but the diagnostics still fail, try a new tape; if this doesn't fix the problem, run the diagnostics on the disk; if they pass, most likely there is a software bug in the diagnostics; report it.
2. If the shell-level command fails, run the disk diagnostics; if they pass, try a new tape drive; if the test still fails, there is either an nX problem or a diagnostic software problem; report it.
3. Otherwise, make sure the **/usr/tmp** directory exists; if not create it and re-run the test.

C&C test Failures

1. First, quit out of the diagnostics on any failures and type the following at the shell level prompt: **dd if=/dev/rxy0b of=/dev/***xxxx* where *xxxx* is the name of the drive. Note the pass/fail status of this shell command. If this shell level command passes, but the diagnostics still fail, try a new tape; if this doesn't fix the problem, run the diagnostics on the disk; if they pass, most likely there is a software bug in the diagnostics; report it.
2. If the shell-level command fails, run the disk diagnostics; if they pass, try a new tape drive. If the test still fails, there is either an nX problem or a diagnostic software problem; report it.

Failures

If all other 1/4" tests pass but this one fails, most likely the tape cartridge is at fault. Try a different tape.

8.7.1 Tape Drive Raw Device Read/Write Test

This test writes data to the tape drive and requires the use of a scratch tape. It takes about five minutes to finish, starting at buffer #1 and ending with buffer #1000.

A successful run displays a test passed message.

If the test fails, try to run the rest of the tape tests, recording the results. Then call BBN ACI.

8.7.2 Tape Drive Block Device Read/Write Test

This test writes blocks of data to the tape drive and requires the use of a scratch tape. Press <Return> after **smt1** to start the test. Buffer #1–#1000

```
Enter 1/4" Block Tape Device <String> ->  smt1 ↵
```

A successful run displays a test passed message.

8.7.3 Tape Drive Short tar Test

This test briefly (about ten minutes long) exercises the **tar** command. The default device is smt1; press <Return> after the prompt to start the test. It requires the use of a scratch tape.

A successful run displays a test passed message.

8.7.4 Tape Drive Long tar Test

This test, which takes about thirty minutes transferring 100 megabytes, exercises the **tar** command. The default device is smt1; press <Return> after the prompt to start the test. It requires the use of a scratch tape.

A successful run displays a test passed message.

8.7.5 Tape Drive Convert and Copy Test

This test dumps data to the tape drive and requires the use of a scratch tape. It takes 15 to 20 minutes, and the tape must rewind after the test is completed. The default device is **rsmt1**; press <Return> after the prompt to start the test.

NOTE

This test may fail if run while booted from a SCSI disk.

```
Enter 1/4" Raw Tape Device <String> -> (rsmt1)↲
-> Convert-&-Copy-Test rsmt1
```

A successful run displays a test passed message.

8.7.6 Tape Drive Retension Test

This test exercises the **retension** command and the tape drive's response. It requires the use of a scratch tape. The default device is **smt1**; press <Return> after the prompt to start the test.

```
-> Retention-Test
Enter 1/4" Raw Tape Device <String> -> (smt1)↲
```

A successful run displays a test passed message.

8.7.7 Tape Drive Write-Protect Test

This test requires the use of a scratch tape. You must put the Write protect ring on the tape reel before starting. The default device is **smt1**; press <Return> after the prompt to start the test.

```
-> Write-Protect-Test
Enter 1/4" Block Tape Device <String> -> (smt1)↲
```

A successful run displays a test passed message.

8.7.8 Tape Drive Erase/Rewind test

This test erases ***all*** data on the tape and requires the use of a scratch tape. The default device is **smt9**; press <Return> after the prompt to start the test. The test takes about five minutes.

```
Enter 1/4" Tape Device (no rewind)  <String> -> (smt9)↲
```

A successful run displays a test passed message.

8.8 Hard Disk Drive Diagnostics

The disk drive diagnostic tests perform the following operations:

1. Seeks to ten random sectors on every head on every tenth cylinder and does a quick read on raw disk device.
2. Seeks to ten random sectors on EVERY cylinder and EVERY head and does a quick read on raw device.
3. Reads defect list off disk and controller address.
4. Reads every tenth sector on every tenth cylinder on block device.
5. Reads EVERY sector on block device.
6. Reads and displays disk label.
7. Reads and displays data from user-specified sector.
8. Reads and displays track header info from user specified track.
9. Copies a large directory to scratch area of disk; compares original files with copies.

CAUTION

The write-protect switch on the disk should ***never*** be enabled while the nX OS is running multi-user, since that will crash the system when it writes to the disk.

The disk drive diagnostic main menu displays the following:

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS (TC2000) Hard-Disk-Drive
    1. Fast-Seek-Test
    2. Extensive-Seek-Test
    3. Raw-Device-Test
    4. Short-Block-Device-Test
    5. Long-Block-Device-Test
    6. Label-Read-Test
    7. Read-Sector-Test
    8. Track-Hdr-Test
    9. File-Copy-Test
    10. Quit

Allowable Drives:  (#Heads  #Cylinders  #Sectors)
  NT 8408 (500)          8        1496        65
  NT 8414 (850)         14        1511        65
Hard-Disk-Drive ->
```

Failures

Possible non-obvious error messages are described in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI.

1. Halt the nX OS, run standalone diagnostics, and replace either the disk or the controller if either is suspect; reboot the nX OS and re-run nX disk diagnostics.
2. Otherwise, halt the nX OS, try a new disk cable, then reboot the nX OS.
3. Else, if non-fatal error occurs on a small subset of the sectors, check those sectors against the bad sector table (defect list) using **diskutil**, and ignore those sectors that match; if a sector fails in the seek test that doesn't show up in the defect list, you have a bad drive; replace it.

Possible non-obvious error messages are described in Subsection 8.8.2 (seek tests) and in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI.

1. Replace disk if the same sectors keep failing, if those sectors are not found in the defect list.
2. Or, use **diskutil** to add the sectors to the defect list, and re-install and reboot the nX OS.

Kernel Driver error message:

1. Try a different king node.
2. Halt the nX OS, re-seat controller board and try different cables.
3. Reformat disk or try a new disk.
4. Replace controller board.
5. Call BBN ACI.

Open/Write/Read error:

If the disk boots without problems, but you get open/read/write/close errors during disk tests, there probably is a diagnostic software bug; report it.

Timeout or hang error:

If the nX OS boots ok, but you get timeout errors during disk tests, there probably is a diagnostic software bug; report problem.

Kernel panic:

1. Replace king board and re-try test.
2. Re-format disk and reinstall the nX OS.
3. Try a new disk.
4. Try new cables.
5. Replace Controller board.
6. Call BBN ACI.

8.8.1 Disk Drive Seek Test (Short Version)

This test takes about 15 minutes to finish, going from cylinder 0 to 1510.

To run the disk drive fast seek test, type **1** at the main menu prompt and type in the drive number at the prompt:

```
-> Fast-Seek-Test
Enter Disk Drive Unit Number -> (0) ↲      Press <Return> if 0.

Enter Disk Drive Type -> (NT-8414) ↲   Press <Return> if NT8414.
```

A successful run displays a test passed message.

8.8.2 Disk Drive Seek Test (Long Version)

The long version of the disk drive seek test takes about one and one half hours. 0 - 1510.

```
Disk Drive Unit Number <string> 0
 Disk Drive Type NT-8414
-> Extensive-Seek-Test
Enter Disk Drive Unit Number
 -> (0) ↲                                  Press <Return> if 0.
-> Extensive-Seek-Test 0
Enter Disk Drive Type
-> (NT-8414) ↲                            Press <Return> if NT8414.
-> Extensive-Seek-Test 0 NT-8414
```

A successful run displays a test passed message.

8.8.3 Disk Drive Raw Device Test

This test will read and display the disk controller's address and the disk-resident defect list. At the prompt type the disk drive number or <Return> if 0.

```
Disk Drive Unit Number <number>  (0) ↲
```

A successful run displays a table of information and a test passed message.

Possible non-obvious error messages are described below. If you get a non-obvious message that is not described, call BBN ACI.

1. Run the 'system configuration' test: if disk doesn't respond, replace controller (after halting nX).
2. Halt the nX OS and run standalone tests; if they fail, replace controller.
3. Call BBN ACI.

8.8.4 Block Device Test (Short Version)

The short version of the Block Device test takes about 15 minutes. Certain sectors may fail; you should declare those sectors bad using the **diskutil** program. A successful pass displays the following:

```
Disk Drive Unit Number <number>  (0) ↲          Press <Return> if 0.
Disk Drive Type          NT-8414
Reading every tenth sector on cylinder 0
                    .
                    .       more messages.
                    .
Reading every tenth sector on cylinder 1510
SHORT BLOCK DEVICE TEST PASSED.
Short-Block-Device-Test: Test Passed
```

8.8.5 Block Device Test (Long Version)

This test reads every sector on the disk; it takes about five hours. Certain sectors may fail; these sectors should be mapped into the bad sector table by the **diskutil** program. Press <Return> if you want to test disk #0:

```
Disk Drive Unit Number <number>  (0) ↲
Disk Drive Type       NT-8414
```

A successful run displays a test passed message.

8.8.6 Label Read Test

This test will read and display the disk-resident label. At the prompt type the disk drive number or <Return> if 0.

```
Disk Drive Unit Number <number>  (0) ↲
```

A successful run displays a table of information and a test passed message.

Possible non-obvious error messages are described in Subsection 8.8.2 (seek tests) and below. If you get a non-obvious message that is not described, call BBN ACI.

1. Check to see if cylinder 0, track 0, sector 0 is in defect list; if so, don't run this test.
2. Re-format disk and re-install the nX OS.
3. There probably is a diagnostic bug: report failure symptoms.

8.8.7 Read Sector Test

This test will read and display the contents of a specific sector. Use the prompt responses below as an example.

```
Enter Disk Drive Unit Number  -> <number>  (0) ↲
Enter Disk Drive Type  ->  <string> (NT-8414) ↲
Enter Cylinder Number  -> <number> (0) 1 ↲
Enter Head Number  -> <number> (0) 1 ↲
Enter Sector Number  -> <number> (0) 5 ↲
-> Read-Sector-Test 0 NT-8414 1 1 5
```

A successful run displays a table of information and a test passed message.

Possible non-obvious error messages are described in Subsection 8.8.2 (seek tests) and below. If you get a non-obvious message that is not described, call BBN ACI.

1. Sector may be a 'slipped' sector: check against **diskutil** defect list and ignore error if sector is in list.
2. Run Short-Block-Device and Quick-Seek tests: if they pass, reformat your disk.
3. Replace cable.
4. Call BBN ACI.

8.8.8 Track Header Test

A successful pass displays the following:

```
-> Track-Hdr-Test 0 NT-8414 2 3

Disk Drive Read Track Header Test

Disk Drive Unit Number <number>  (0) ↲      Press <Return> if 0.
Disk Drive Type          NT-8414

Cylinder: 2
Head    : 3

Track Header Data:

0200033f    02400300    dddddddd    02000300
02000301    02000302    02000303    02000304
      .
      .     more messages.
      .
02000339    0200033a    0200033b    0200033c
0200033d    0200033e

READ TRACK HEADER TEST PASSED.
Track-Hdr-Test: Test Passed
```

If all other tests pass but this one fails, there probably is a diagnostic software bug: report symptoms of failure.

8.8.9 File Copy Test

A successful run displays a test passed message.

Possible non–obvious error messages are described in Subsection 8.8.2 (seek tests) and in the following paragraphs. If you get a non–obvious message that is not described, call BBN ACI.

1. Make sure **/usr/tmp** directory exists; if not create it and re–run test.
2. Run standalone diagnostics (after halting nX).
3. Re–format or replace disk; re–run test.
4. Replace cables and/or controller.

8.9 Ethernet Controller Test

The Ethernet controller test performs the following:

1. Ping test: **pings** (echoes back and forth) blocks of data to user specified host computer on network.
2. Statistics: prints collection of communication statistics on ethernet device. Not implemented yet.
3. Data transfer test: transfers data to specified system on network.

The Ethernet controller menu displays the following:

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS (TC2000) Ethernet-Controller
    1. Ping-Test
    2. Statistics-Test
    3. Data-Transfer-Test
    4. Quit

Excelan 302 Card

Ethernet-Controller ->
```

Possible non-obvious error messages are described in the following paragraphs. If you get a non-obvious message that is not described, call BBN ACI.

1. Check cable; replace if suspicious.
2. Try a different ping host.
3. Reboot the nX OS and retry.
4. Quit out of the diagnostics and attempt to **ftp** to known slave host; then re-run test.
5. If tests on other devices fail, replace HVE repeaters.
6. Replace Ethernet board.

8.9.1 Ethernet Controller Ping Test

This test takes about three minutes to complete. Enter the name of a system on **your** network at the prompt.

A successful pass displays a table of information with no errors listed.

8.9.2 Ethernet Controller Statistics Display Test

The kernel Excelan statistics are only updated every 25 minutes. You must wait at least this long after booting the kernel before the first statistics are available.

A successful pass displays a table of information with no errors listed.

8.9.3 Ethernet Data Transfer Test

Select the Ethernet data transfer test from the Ethernet-Controller menu. This test transfers a file from a host computer to the TC2000 using **ftp**.

A successful pass displays the following:

```
Ethernet Data Transfer.
    Remote Host             <String>        bfly-vax.bbn.com
    Remote login            <String>        guest
    Path name               <String>        /vmunix
-> Data-Transfer-Test
Enter Remote Host     <String> -> (bfly-vax.bbn..com)   wheat.bbn.com ↲
-> Data-Transfer-Test  wheat.bbn.com
Enter Remote login    <String> ->   (guest)   rpreston ↲
-> Data-Transfer-Test  wheat.bbn.com  rpreston
Enter Path name     <String> -> (/vmunix)   /nfs/wheat/u1/rpreston/temp/zndx.addr ↲
-> Data-Transfer-Test wheat.bbn.com rpreston /nfs/wheat/u1/rpreston/zndx.addr

Ethernet-Controller Data-Transfer-Test...
Enter remote host password:   ↲                    < = Note: echo disabled here!
Connected to wheat.bbn.com
220 wheat.bbn.com FTP server (Version 4.12 Thu May 11 16:52:09 EDT 1989) ready
Password required for login rpreston

        .   More messages.
        .

Comparing /usr/tmp/dummy10 with /usr/tmp/dummy1
Ethernet Data Transfer Test Passed
```

8.10 CRT Terminal Driver Diagnostics

The CRT Terminal Driver diagnostics perform the following:

1. Tests the host adapter card using the on-board diagnostic
2. Tests the cluster controller connection.
3. Displays the adapter log or the remote controller log.
4. Exercises a single or multiple terminal lines.
5. Dumps all displayable characters to a terminal for a specified time.

```
Menu:  nX-BASED-PERIPHERAL-DIAGNOSTICS (TC2000) CRT-Terminal-Controller
    1. Host-Adapter-Loopback-Test
    2. Cluster-Controller-Loopback-Test
    3. Adapter-Log-Test
    4. Remote-Cntrlr-Log-Test
    5. Tty-Line-Loopback-Test
    6. Multiple-Tty-Loopback-Test
    7. Character-Screen-Dump
    8. Quit
```

8.10.1 Host Adapter Loopback Test

The sixteen self-tests are comprehensive and are intended to catch and diagnose almost all errors short of catastrophic ones such as a no clock or no power to the Host Adapter board. In addition to executing the previous self-test this test performs a "handshake" with the host computer to check the Multibus data bus interface. The test also performs a `DATA LOOPBACK TEST` and an `INTERRUPT LOOPBACK TEST`.

Check the jumpers with the instructions for correct position. For only this test to successfully complete, the E22 jumper pins **3** and **4** must be **out**.

If the test passes it displays a table with no errors.

8.10.2 Cluster Controller Loopback Test

This tests the connection from the host adapter to the cluster controller. It prompts for the controller number:

```
-> Cluster-Controller-Loopback-Test 1
```

If the test passes it displays a table with no errors.

8.10.3 Host Adapter Log Display Test

This test displays a table of information from the host adapter card. It prompts for the controller number:

```
-> Cluster-Controller-Loopback-Test 1 ↲
```

8.10.4 Remote Controller Log Test

This test displays a table of information from the cluster controller. It prompts for the controller number:

```
-> Cluster-Controller-Loopback-Test 1 ↲
```

8.10.5 TTY Line Loopback Test

This tests a single TTY line and requires a loopback connector on the line under test. It prompts for the TTY line (default tty00):

```
Enter Tty Line. <String> -> /dev/tty00 ↲
```

If the test passes it displays a test passed message.

8.10.6 Multiple TTY Loopback Test

This tests multiple TTY lines and requires a loopback connector on each of the lines under test. It prompts for the first and last TTY line:

```
Enter first tty Line. <String> -> /dev/tty00 ↲
Enter second tty Line. <String> -> /dev/tty07 ↲
```

If the test passes it displays a test passed message.

8.10.7 Character Screen Dump

This test dumps all displayable characters to a terminal for a specified time. It prompts for the tty line and duration:

```
Enter first tty Line. <String> -> /dev/tty00 ↲
Enter Burn-in time (seconds) <Number> -> 20 ↲
```

If the test passes it displays the pattern on the selected terminal.

8.11 1/2" Tape Drive Diagnostics

The 1/2" tape drive diagnostics perform the following:

1. Tests the drive as a raw device: writes out 1000 buffers, reads them back, compares data.
2. Tests the drive as a block device (buffered I/O).
3. Does a short or a long **tar** test.
4. Does a 'convert and copy' test.
5. Writes 20 filemarks to tape, backspaces over each one of them one at a time.
6. Tests the write protect detect mechanism on the tape drive.
7. Rewinds tape.

Most of the 1/2" tape drive tests are the same as the tests for the 1/4" SCSI tape (in Section 8.7). If you encounter an error, check the descriptions in the SCSI tape section. Only differences are described in this Section. The 1/4" tape is **rsmt1** while the 1/2" tape is **rmt0**, or **rmt8**, for low or high density respectively.

```
Menu:  nX BASED-PERIPHERAL-DIAGNOSTICS (TC2000) 1/2"-Tape-Drive
    1. Raw-Device-RW-Test
    2. Block-Device-RW-Test              Not implemented
    3. Short-Tar-Test
    4. Long-Tar-Test
    5. Convert-&-Copy-Test
    6. EOF-Search-Test
    7. Write-Protect-Test
    8. Rewind-Test
    9. Quit

Jaguar = Interphase 4210 SCSI Controller
CIPHER = SCSI Tape Drive
```

NOTE

To run the SCSI or 1/2-inch tape **periph_diag** tar tests, be sure to change (**cd**) to the /stand directory. Otherwise the tests will fail.

Failures

If a test is unsuccessful:

1. Try a different tape.
2. Halt the nX OS; run the standalone diagnostics—specifically, the 1/2" Magnetic Tape diagnostic that leads one through the front-panel driven self-tests on the tape drive itself.
3. Try a different tape drive.
4. Try a different controller.

8.11.1 Raw Device Read/Write Test

This test writes data to the tape drive and requires the use of a scratch tape. It will take 15–20 minutes to finish, starting at buffer #1 and ending at buffer #1000. Press <Return> at the prompt:

```
Enter 1/2" Raw Tape Device   <String> -> (rmt0) ↲
```

A successful run displays a test passed message.

8.11.2 Block Device Read/Write Test

This test writes data to the tape drive and requires the use of a scratch tape. It takes about five minutes to complete.

NOTE

This test has not been fully implemented and should **not** be used.

A successful run displays a test passed message.

8.11.3 Short or Long tar Test

This test writes data to the tape drive and requires the use of a scratch tape. The short tar takes about five minutes and the long tar takes about 30 minutes to complete. Press <Return> at the prompt:

```
Enter 1/2" Raw Tape Device   <String> -> (rmt0) ↲
```

A successful run displays a test passed message.

8.11.4 Convert and Copy Test

This test writes data to the tape drive and requires the use of a scratch tape. This test requires a 2400' tape if it is run at 1600 bpi (rmt0). A 1200' tape is fine if using 6250 bpi (rmt8).

If you run out of tape (*e.g.* using rmt0 and a 1200' tape), you will get a controller error **3** (end–of–tape), but the test will still pass since nothing was wrong.

```
Enter 1/2" Raw Tape Device <String> -> (rmt0) ↵
```

A successful run displays a test passed message.

8.11.5 Tape Drive End–of–File (EOF) Search Test

This test writes data to the tape drive and requires the use of a scratch tape.

NOTE

This test has not been fully implemented and should **not** be used.

```
Enter 1/2" Raw Tape Device <String> -> (rmt0) ↵
```

A successful run displays a test passed message.

8.11.6 Tape Drive Write–Protect Test

This test has the potential to destroy the magnetic tape data and requires the use of a scratch tape.

```
Enter 1/2" Block Tape Device <String> -> (mt0) ↵
```

If you run this test with a write ring, the test fails as it should with the following message:

```
WRITE PROTECT MECHANISM FAILED
```

A successful run displays a test passed message.

8.11.7 Tape Drive Rewind test

Typing an **8** executes the Rewind test. This test requires a scratch tape to be loaded in the tape drive.

```
Enter 1/2" Tape (no rewind) <String> -> (rmt8) ↵
```

A successful run displays a test passed message.

9

Stress Test

9.1 Overview

Execution of the TC2000 I/O Stress Test can be entirely automatic or defined by the user (to test particular parameters or possible failure modes). While this is not intended to diagnose problems it might indicate problems in particular subsystems.

Stress Test is designed to place a heavy burden on the I/O subsystems. It exercises I/O in ways that a typical customer in the field may not encounter.

If stressing a marginal peripheral subsystem causes a system crash, loss of disk data could occur. Avoid running the stress test while users are on the system, since their work will be jeopardized.

Ethernet stress testing during normal working hours is likely to place a load on the network that may affect other users and may therefore alarm people responsible for network maintenance. We advise that Ethernet stress testing proceed only after consulting with network maintenance personnel. Even on a perfectly fine system, the stress test will make system response unacceptably slow for users.

Just prior to starting, the Stress Test attempts to add all available free nodes to the Public cluster, in order to fork processes on remote nodes later on. In most instances, not every node is available. This results in a non–fatal error message, and Stress Test continues with the available nodes.

9.2 How to use Stress Test

Use Stress Test in automatic mode. Once started, errors may checked from the command prompt using the **kill ?** command. To exit Stress test type **exit**.

9.2.1 Run Times

The typical run time is dependent on how many subsystems tested, but it should be run for at least one half hour.

9.3 Automatic Mode

When the Stress Test is first invoked (Figure 9–1), it prompts to select automatic or command mode. Invoking automatic mode starts a standard suite of stress test functions involving all five I/O subsystems (disk, Ethernet, 1/4" tape, 1/2" tape, and terminal). Several other I/O activities are also started which will loop until explicitly killed. To exercise the tape drives, load the drives with scratch tapes and put the half-inch tape drive online before starting.

Figure 9–1 Stress Test Automatic Startup

```
# cd /stress ↲                                  After logging in as 'root', change directories
# ./stress ↲                                    to /stress and start Stress Test.
ADDING ALL NODES TO PUBLIC CLUSTER; PLEASE WAIT...
.
.
.
Enter choice (by number):
  1. Start standard stress test without user intervention
  2. Start stress test 'command enter' mode
1 ↲                                             Type 1 for automatic mode.
Test disk?       [y/n] y ↲
Test 1/4" tape?  [y/n] y ↲
Test tty?        [y/n] y ↲
Test 1/2" tape?  [y/n] y ↲
Test ethernet?   [y/n] n ↲                      It is not recommended to use the Ethernet
Wait...                                         exercises.
Execution of standard Stress Test has begun...
ENTER 'kill ?' FOR A LIST OF ALL STRESS TEST PROCESSES RUNNING IN AUTO-MODE...
```

After the tests are started, the command prompt is displayed.

You may also use any of the commands listed below:

quit or **exit**	Leave Stress Test
help [*command*]	Displays a help list or specific help
kill [?]	Displays a list of all known child processes
sst	Runs a standard stress test on all five I/O systems
ver	Displays the version number

9.4 Command Mode

The various Stress Test commands are described below with optional arguments in brackets:

'cc' command

`cc [-f] [-r]`

The **cc** command starts a child process that invokes the C compiler to compile and link a moderately-sized C program. C compilation involves a moderate amount of disk I/O, depending upon the size of the program being compiled.

-f run in a loop forever (can be stopped with the **kill** command)
-r run remotely on a node other than the Master Node

'combine' command

```
combine [-rf] disk_io scsi_io
combine [-rf] disk_io scsi_io ethr_io
combine [-rf] disk_io scsi_io ethr_io tape_io
```

The **combine** command starts one or more child process that exercises two, three, or four I/O subsystems in isolation of the other I/O subsystems. The allowable subsystems are: "disk_io", "scsi_io", "tape_io", "ethr_io", and "term_io" (tty I/O). Thus, for example, **combine disk_io ethr_io scsi_io** will start I/O activity on the disk, the ethernet, and the scsi tape drive all at the same time. The particular I/O subsystems can be specified in any order.

-f run in a loop forever (can be stopped with the 'kill' command)
-r run remotely on a node other than the Master Node

'do' command

`do` *filename*

The **do** command executes a series of stress test commands that are located in a file, as if the commands were typed in at the keyboard. This allows custom Stress Test command scripts to be stored and duplicated.

'exec' command

`exec [-w]` *nX-command*

The **exec** command executes any valid nX Operating System shell-level command. This feature allows you to write your own I/O intensive commands and execute them from stress test, so that a your custom commands and standard stress test commands can be executed simultaneously.

-w Makes stress test wait for the command to completed. Without the **-w** flag, the command will be queued, but the command parser will return immediately allowing several commands to be active at once.

'fork' command

```
fork [-rf] disk_io
fork [-rf] scsi_io
fork [-rf] ethr_io
fork [-rf] tape_io
fork [-axf] term_io
```

The **fork** command creates a child process that exercises a specific I/O subsystem by itself. The allowable I/O subsystems are: **disk_io**, **scsi_io**, **tape_io**, **ethr_io**, and **term_io** (tty I/O). For example, the command **fork disk_io** starts intensive disk I/O activity. Depending upon the I/O device, many forks can be issued, one after the other: **fork -f disk_io** commands can be issued ad infinitum, until either the number of open files allowed by one process is exceeded or the allowable number of child processes that can be forked by one processes has been exceeded. This will not work with other devices, such as the SCSI tape drive, which will return a 'device busy' error upon the issuance of a second **fork scsi** command. In this case, the second fork must await the completion of the first.

-f run in a loop forever (can be stopped with the **kill** command)
-r run remotely on a node other than the Master Node
-a (term_io only) run tty I/O on 8 terminals instead of just /dev/tty00 (/dev/tty00 through /dev/tty07)
-x (term_io only) run tty I/O assuming getty is 'off' on each tty; without the **-x** flag, getty is assumed 'on';

'help' command

```
help [command]
```

The **help** command displays help on a particular command (syntax, usage) or displays a list of all available commands. If no command name is given as an argument to the **help** command, a list of all available commands is given. If a valid stress test command is given as an argument, a more extensive help message on that particular command is displayed. Typing a **?** is also accepted.

'kill' command

```
kill [?] [pid] [all]
```

The **kill** command will stop a child process that was invoked by a previous stress test command which used the **-f** loop forever flag.

?	displays a list of all known child processes for 'kill'ing. A process id number (pid) will kill that particular process.
all	kills all known child processes that have been issued with the **-f** loop forever flag set.

'nroff' command

```
nroff [-f] [-r]
```

The **nroff** command starts a child process that executes an nroff text formatter session on a moderately-sized nroff text document.

-f	run in a loop forever (can be stopped with the **kill** command)
-r	run remotely on a node other than the Master Node

'quit' command

```
quit
```

The **quit** or **exit** commands exits the stress test and returns the user to the shell command level.

'sst' command

```
sst
```

The **sst** command runs a standard stress test designed to activate all five TC2000 I/O subsystems simultaneously, as well as a large amount of switch traffic. The same effect can be achieved by entering **do sst_script** or by bypassing 'command enter' mode and entering 'automatic mode' at the time of Stress Test invocation.

'sys_call' command

```
sys_call [system-call]
```

The **sys_call** command invokes nX system calls found in the *nX Programmers Reference Volume II*. Some of the calls are invoked more than once with different parameters. Since the stress test is first and foremost an I/O stress test, only those system calls that are expected to engage in a lot of I/O activity are invoked.

With no arguments, all major system calls are invoked sequentially. With a valid system call argument, only that system call is invoked.

'mm' command

```
mm [-f]
```

The **mm** command is designed to create lots of switch traffic by exercising Uniform System Matrix Multiplies.

-f run in a loop forever (can be stopped with the **kill** command)

'ver' command

```
ver
```

The **ver** command displays the stress test version number.

9.4.1 Notes on use of Commands

Usage

Commands can be abbreviated, as long as it remains unique enought to identify the command. Arguments to commands generally cannot be abbreviated.

Although terminal I/O stress testing will work without actual CRT's connected to ports tty00 through tty07, you will be able to actually see the I/O only if the CRT's are connected (obviously). The cluster controller(s) must be hooked up or errors will result.

The stress test is used most effectively when many activities are ongoing at the same time. Issuing a single **fork disk_io** is not very interesting. On the other hand, issuing five or six **fork -f disk_ios**, several compiles (loop forever), several **nroffs** (loop forever), several forks on other I/O subsystems, a couple of **combines**, and several matrix multiplies, will generally stress the system to its limit or beyond. 'Automatic mode' (standard Stress Test) represents a middle ground.

The **help** command, followed by a valid stress test command as an argument, displays (on-line) all a user needs to know to correctly invoke a stress test command.

Kill is to be used only on commands that have been (previously) invoked with the **-f** (loop forever) flag. It cannot be used on commands that have been issued without the -f flag, since these commands will die of their own accord when they have completed. When issuing a **kill** command, the process to be killed may loop one more time after issuance of the **kill** command before it dies.

The filename containing commands for the **do** command must contain valid stress test commands, not nX shell script commands.

10

SCSI Disk Tool

10.1 Overview

DISKTOOL is a utility for formatting, labelling, partitioning, and trouble-shooting SCSI disks. It has been designed for use with all SCSI disks. There are two versions of DISKTOOL; a standalone version located on the TCS that runs under TEX and another version that runs under the nX OS. This chapter describes the standalone version.

This chapter only covers those functions that might be used in testing or diagnosing problems. For a more complete description of the functions in DISKTOOL refer to the *TC2000 System Administration Guide*.

10.2 How to Use DISKTOOL

DISKTOOL is useful for checking the some of the basic SCSI disk functions. Before doing anything you must attach to the disk drive, which if successful indicates some functionality of the subsystem. Typical use for testing might be to check the label, the geometry, the partitions and try some of items on the **Exercise-disk** menu. To exit DISKTOOL test type **quit** at the main menu.

10.2.1 Run Times

The typical run times for the suggested functions are mostly under one minute. Depending on the disk drive vendor, the time some functions take may vary.

10.2.2 Attaching and Specifying

A disk must be **attached** or **specified** before any meaningful function can be used. A disk is **specified** if DISKTOOL has been able to find the device. A disk is **attached** if DISKTOOL has been able to send an inquiry, mode sense.

and capacity commands to the disk. These commands are used to indicate vendor type and disk geometry. DISKTOOL then tries to read and verify the label on an attached disk.

All DISKTOOL functions can be performed on an attached disk. If the disk is **specified**, but not **attached**, then you will not be able to format or label the disk. This is because the disk or disk controller has not been able to respond to the SCSI inquiry, mode sense, or get capacity commands. DISKTOOL is unable to label a disk without this information. The disk or disk controller probably also has a serious hardware problem. However, you will be able to run some simple disk tests on a specified disk (**self-test**, **spin-up**, **spin-down**, and other tests in the **Exercise** menu). These may help you find the problem.

The **Attach-to-disk** menu item uses two arguments; the SCSI bus number (0 or 1) and the device number on the SCSI bus (0–6). After you enter the disk's bus and device number, DISKTOOL prompts for confirmation. It then sends four SCSI commands: inquiry, mode sense (once for format and once for geometry), and read capacity. If all of the commands succeed, the disk is considered **attached**, and DISKTOOL tries to read the label from the disk. If it finds the disk but one or more of the SCSI commands fail, then the disk is considered **specified**, not **attached**.

If **Attach-to-disk** fails to even specify the disk, try using the **Specify-disk** function on the **Fix-disk-problems** menu (Figure 10–1).

The **Display-label** menu reads the label from the disk. If the read succeeds, and if there is a label on the disk, the contents of the label are displayed. The label contains the disk name, disk partitioning, and disk geometry.

If DISKTOOL failed to verify the label geometry it means that the geometry on the label does not match the geometry reported by the disk. This is usually an indication that the label is corrupted or wrong, although it may mean that a disk DIP switch setting has changed. If the geometry is correct, you can fix the label be writing it back to the disk. If it is the geometry that is incorrect, refer to the *TC2000 System Administration Guide* to make the changes.

```
Menu:  SCSI disktool
       1. Attach-to-disk                <scsi bus (0-1), dev (0-6)>
       2. Display-label
       3. Show-disk-geometry
       4. Change-partitioning-on-label/
       5. Name-disk (changes label)
       6. Fix-disk-problems/
       7. Exercise-disk/
       8. Terminal-type/                Hardcopy, Menu-On, More-Off
       9. Quit

SCSI disktool ->
```

Figure 10–1 Fix Disk Problems Menu

```
Menu:  SCSI disktool Fix disk problems
     1. Show-defects-on-disk/          <log|phys>
     2. Reassign-block-on-disk         <logical block number>
     3. Format-disk/
     4. Verify-format                  <optional starting block>
     5. Drive-ready-test
     6. Quit-to-prev-menu

Fix disk problems ->
```

10.2.3 Exercising the Disk

There are a few simple tests in DISKTOOL that can be used in troubleshooting disk subsystems. These tests are available from the **Exercise–disk** menu:

```
Menu:  SCSI disktool Exercise disk
     1. Specify-disk                   <scsi bus (0-1), dev (0-6)>
     2. Spin-disk-up-or-down/          <up | down>
     3. Drive-ready-test
     4. Self-test
     5. Quit-to-prev-menu

Exercise disk ->
```

Specify–disk

This command has two arguments; the bus number and the device number. See the section on Attaching and Specifying Disks for more details.

Spin–disk–up–or–down

This command has one argument; **up** or **down**. It sends a SCSI command to the disk to spin it up or down. This command will only take effect if the drive supports the command, and if the DIP switches are set enable it. It is common for the DIP switches to be set to disable this command.

Drive-ready-test

This command sends a SCSI drive ready command to the disk, and displays the results of the test. Do not assume that the disk is spun up if this command completes successfully. The precise meaning of this command is vendor-dependent.

Self-test

This command sends a SCSI self-test command to the disk and displays the results of the test. It will probably take between 2 and 10 minutes to complete, although the complexity of the self-test, and thus the time it takes, is vendor-dependent.

11

Modem Testing

11.1 Overview

This program provides a menu based interface to the configuration data and internal diagnostics of the UDS FasTalk 2400 PC modem card for the TCS Master. It provides the ability to check and set modem configuration data as well as running the diagnostic tests. For the diagnostics it also interprets the return codes and displays the information in a useful format.

This chapter provides information on the configuration parameters and tests available from the menu. It provides explanations of the various configuration parameters, and descriptions of the tests that can be selected (including the significance of the tests, how to invoke them, and more detailed explanations of the result messages).

11.2 How to use the B2MDM Diagnostics

Once it is loaded and the terminal type has been selected, initialize the modem from the main menu (2). Go to the Modem-Diags menu. Check the commonly changed parameters on the System-Level-Diagnostics menu. Finally run the tests on the Board-Level-Diagnostics menu. There are more parameters on the Commands menu.

11.2.1 Run Times

The modem interface is a simple unit and therefore the tests are very brief. To complete all of the test should take less than five minutes.

11.2.2 Initializing the System and Loading Diagnostics

The B2MDM diagnostics are started from the DOS prompt (see Chapter 3). From DOS by type:

`C:\ CD \DIAG\B2MDM ↲`	Changes to the B2MDM directory.
`C:\DIAG\B2MDM> START ↲`	Initializes the system.

This brings up the initial (terminal type) menu. Select your terminal type. After you make your selection the B2MDM diagnostic main menu is displayed.

11.3 Modem Diagnostics Menu System

The main menu provides access to all of the sub-menus for the modem diagnostics. The main menu is shown below:

```
Menu:  TC/MOD-D
    1. Port-Modem                Port 1
    2. Init-Modem                MODEM MUST BE INITIALIZED!
    3. Modem-Diags
    4. Logging-On/Off            Logging Off
    5. Terminal-Type             Hardcopy, Menu-On
    6. Quit
                            ver  x.xxx
TC/MOD-D ->
```

After initializing (item **2**) select item **3** to change settings or run tests.

```
Menu: TC/MOD-D Modem-Diags

    1. Board Level Diagnostics
    2. System Level Diagnostics
    3. Commands
    4. Quit

   Modem Diags ->
```

11.4 Board Level Diagnostics

Selecting item 1 from the Modem-Diags menu displays the Board-Level-Diagnostics menu. This menu has all of the internal tests available on the FasTalk 2400 PC Modem Card, and interprets and displays the results.

```
Menu: TCS Modem-Diags Board-Tests
    1. Internal-Memory-Test
    2. Analog-Loopback-Test
    3. All-Internal-Tests
    4. Verify-Phone-Line
    5. Dial-Out-Test
    6. Quit

Board-Tests ->
```

Internal Memory Test

This test verifies that all of the memory on the modem card is good. It returns either `Good` or `Failed`. It is possible that the modem will still operate after some of its memory tests as failed, but it cannot be regarded ar reliable or safe to use.

Analog Loopback Test

This test generates a test pattern and then loops it back from the modem's transmitter to it's receiver instead of transmitting the signal through the telephone line. This test will only report failure after several attempts. If it does report failure, there is a hardware problem associated with the modem card.

When this test fails, the user should determine if the modem card is secure in its slot, and that it is addressed as COM2. If the card is properly installed, then the problem is with the card itself, and should be replaced.

If the modem passes this test, all of the hardware on the card is good, and any problem must be with the connectors, cable, or further along the phone circuit.

? Digital Loopback Test?

This test must first establish a link to a distant modem. Once it does, it then requests that the remote modem loop back the signal for line testing purposes. This test can fail in one of three ways. If it is impossible to connect to a remote modem, the test cannot be run. If the remote modem is not configured to grant the loopback, the test cannot be run. Finally, if the loopback is set up but fails, the line quality is inadequate for reliable data transmission. The test will actually return one of `No Connection`, `No Loopback Privilege`, `Fail`, or `Good`.

All Internal Tests.

This menu item runs the memory and analog loopback tests, and will return one of `Memory Bad`, `Loopback Fail`, or `Card Good`.

Verify Phone Line

Dial Out Test

11.5 System Level Diagnostics

This menu permits the user to inspect and modify the most commonly changed parameters of the modem's configuration. It also provides a single step command for resetting the modem to the configuration most commonly need for use with the TCS Master.

```
Menu:  TC/MOD-D Modem-Diags System-Tests
    1. Originate/Autoanswer          Autoanswer
    2. Answer-On-Ring                1
    3. Action-On-DTR-Drop            Ignore
    4. DCD-Setting-Toggle            Constant On
    5. Modem-Speed                   2400 bps
    6. Line-Type-Toggle              Public
    7. Jack-Type-Toggle              RJ11
    8. Quit

System-Tests ->
```

`1. Reset Modem to TCS Defaults`

? Reset Modem to TCS Master Defaults

This menu selection will force all of the configuration information of the modem to the default values for use with the TCS Master, store these values in the default profile, and set it for use on power up initialization.

Originate/Autoanswer

This parameter will be set to either `Originate` or `Autoanswer`. When the modem is in originate mode it initiates connections to other machines. When it is in autoanswer mode it answers the phone line and connects to machines attempting to contact it. The default value for this parameter for use with the TCS Master is `Autoanswer`.

Answer on Ring

When the modem is in autoanswer mode, this parameter determines the number of rings it will wait before answering the phone line. The default value for this parameter for use with the TCS Master is `1`.

Action on DTR Drop

This parameter can be set to one of four values: `Ignore`, `Command Mode`, `Hang Up`, `Reset`. When it is set to `Ignore`, a drop in the DTR signal will not have any effect on the modem's operation. When it is set to `Command Mode`, the modem will return to command mode, but without hanging up the phone line. When it is set to `Hang Up`, the modem hangs up the phone line and returns to command mode. It also disables auto-connection until DTR comes back on. When it is set to reset, the modem hangs up the line, and returns to its initial power-up state. The default value of this parameter for use with the TCS Master is `Ignore`.

DCD Setting Toggle

This parameter has two possible values, `Constant On` and `Carrier`. When it is set to `Constant On`, the modem will hold DCD on all of the time. When it is set to `Carrier`, DCD will be on when a carrier is present on the line, and off at all other times. The default value of the parameter for use with the TCS Master is `Constant On`.

Modem Speed

This parameter is the speed, in bits per second, at which the modem sends and receives data. It can be either `2400`, `1200`, `600`, or `0-300`. The last value is a range because, due to the way data is encoded fo lower speeds, all speeds of 300 bps or less are handle in the same way. The default value of this parameter for use with the TCS Master is `2400`. This parameter will almost always be inspected but not set, since the modem adapts to the speed and data format of whatever is being sent to it.

Line Type Toggle

This parameter can either be `Public` or `Leased`. It is set to `Public` for ordinary, dial-up access, telephone lines; and Leased for dedicated, permanently connected circuits leased from the telephone company. The default value of this parameter for use with the TCS Master is `Public`.

Jack Type Toggle

The modem has software support for two types of jacks; `RJ11`, which is an ordinary phone, and `RJ12` which is typical of PBX systems. The `RJ11` designation also includes RJ41 and RJ45 jacks, and the `RJ12` designation also includes RJ13 jacks. The default value of this parameter for use with the TCS Master is `RJ11`.

11.6 Commands

This menu permits the user to inspect and set additional parameters for the modem, that are less likely to require modification than those on the system menu for the modem. This menu also makes it possible for a user to save the current profile, along with whatever changes may have been made to it, and to use either this saved profile or the default one for the TCS Master.

```
Menu:  TC/MOD-D Modem-Diags Commands
    1. Local-Echo-On/Off              On
    2. Speaker-Volume                 medium
    3. Speaker-Mode                   On Until Connect
    4. Pause-for-Carrier-Detect       30 Seconds
    5. Carrier-Detect-Interval        .6 Seconds
    6. Carrier-Loss-Disconnect        1.4 Seconds
    7. Grant/Deny-Remote-Loop         Granted
    8. Quit

Commands ->
```

Local Character Echo

This will be set to either `On` or `Off`. When it is `On`, the modem will echo any characters typed in command mode. When it is `Off`, the mode will not. The default value for this parameter for use with the TCS Master is `On`.

Speaker Volume

The modem is equipped with a speaker to permit the user to monitor call progress. The volume of this speaker can be set to one of three levels, `Low`, `Medium`, and `High`. The default value of this parameter for use with the TCS Master is `Medium`.

Speaker Mode

This parameter controls which portions of a call are sent to the speaker for monitoring purposes. It has four possible values: `Off`, `Until Connect`, `Entire Call`, and `Call Progress`. When it is set to `Off`, nothing is ever sent to the speaker. When it is set to `Until Connect`, the user will be able to listen to the progress of the call from the time the line goes off hook until connection to another modem is accomplished. When it is set to `Entire Call`, the speaker is active for the entire duration of the call, from the time the line goes off hook until hang-up. This mode is generally only used for trouble shooting. Finally, when it is set to `Call Progress`, the speaker is active from the time dialing is completed until connection to another modem is accomplished. The default value of this parameter for use with the TCS Master is `Until Connect`.

Pause for Carrier Detection

This parameter determines how long the modem will look for a carrier when the telephone line type is Public. The default value of this parameter for use with the TCS Master is 30 seconds.

Carrier Detect Interval

When DCD is set to follow the carrier, this parameter determines how long the carrier must be present before the modem brings up the DCD signal. The default value of this parameter for use with the TCS Master is .6 seconds.

```
    7. Grant/Deny-Remote-Loop          Granted
    8. Quit

Commands ->
```

Carrier Loss Disconnect Delay

When the modem is set to disconnect on carrier loss, this parameter determines how long the carrier signal must be missing before the modem treats it as lost. This delay helps prevent disconnection due to noise or other line problems. The default value of this parameter for use with the TCS Master is 1.4 seconds.

Grant/Deny Remote Loop

This parameter controls whether or not a remote modem can successfully request that the local modem loop signals back to it for testing purposes. The parameter can be set to either `Granted` or `Denied`. the default value of this parameter for use with the TCS Master is `Granted`.

A

Error Codes for Peripherals

A.1 1/4" SCSI Tape Drive Subsystem

This subsection lists the error codes for the Interphase 4210 Jaguar SCSI controller and the command completion and error codes for the Tandberg drive.

A.1.1 Interphase 4210 Jaguar SCSI Controller Error Codes

```
0x00  "Good Status: no error detected"
0x01  "Queue Full: specified work queue is full"
0x02  "Work Queue initialization error: specified work queue hasn't been init."
0x03  "First Command Error: first command issued must be Init. Cntrlr command"
0x04  "Command Code Error: invalid command in queue entry"
0x05  "Queue Number Error: invalid work queue number specified"
0x06  "Queue Already Initialized: cannot initialize a work queue twice"
0x07  "Queue un-initialized: specified work queue hasn't been initialized"
0x08  "Queue Mode not ready: Init. Cntrlr must be issued before starting queue
      mode"
0x09  "Unavailable Command: unknown IOPB type; check IOPB format"
0x0a  "Priority Error: the priority specified for the work queue is invalid"
0x10  "Reserved Field Error: a reserved IOPB field has a non-zero in it"
0x11  "SCSI RESET IOPB has executed successfully"
0x12  "Jaguar Port 2 was used, but is unavailable"
0x13  "SCSI ID Error: invalid SCSI ID used for TANDBERG Drive; check IOPB"
0x14  "SCSI BUS Stuck in Reset state"
0x15  "SCSI Command Aborted by spurious SCSI Bus Reset"
0x20  "VME Bus Error: bus error occurred during the DMA transfer of VME data"
0x21  "VME Bus Timeout occurred during data transfer"
0x23  "Illegal VME Bus Address (odd boundary) detected"
0x24  "Illegal VME Bus Memory Type specified; check IOPB"
0x25  "Illegal transfer count specified (odd count); check IOPB"
0x30  "SCSI Selection Timeout Error"
0x31  "SCSI Disconnect Timeout Error"
0x32  "SCSI Bus Hardware Error"
0x34  "SCSI Transfer Count Exception"
```

A.1.2 Tandberg Drive Command Completion Codes

```
0x00  "Good Status: the requested operation was completed successfully"
0x02  "Check Condition: an abnormal condition has occurred. Issue Request Sense"
0x08  "Busy Status: TANDBERG Drive is busy executing a previous command"
0x10  "Intermediate Status: TANDBERG Drive executing series of linked commands"
0x18  "Reservation Conflict: Jaguar has attempted to access TANDBERG drive when
       it was reserved by another Initiatior (host)"
```

A.1.3 Tandberg Drive Error Status

```
0x00  "No sense: no info available (Filemark or End-of-Media possibly detected)"
0x01  "Tape Drive is hung"
0x02  "Tape Drive is not ready (cartridge not inserted or loaded)"
0x03  "Unrecoverable media data error occurred"
0x04  "Hrdwr error: power-up selftest failed or parity error detected"
0x05  "Illegal request: bad parameter detected in cmd issued to drive"
0x06  "Unit attention (door opened?)"
0x07  "Write protect error"
0x08  "Logical End-of-Media (end of recorded area) detected"
0x09  "Reserved error code (no info available)"
0x0a  "Copy command aborted because error detected on src/destdevice"
0x0b  "Command aborted by Tape Drive for unspecified reason"
0x0c  "Reserved error code (no info available)"
0x0d  "Volume overflow: attempt to append data after EOM detected"
0x0e  "Verification failure: data miscompare on verify command"
0x0f  "Reserved error code (no info available)"
```

A.1.4 Tandberg Drive Extended Error Codes

```
0x00  "No error: extended error code = 0x00"
0x01  "Append Error"
0x02  "Bad Command Block"
0x03  "Bad Parameter Block"
0x04  "Bus Parity Error"
0x05  "Busy"
0x06  "Capstan Servo Error"
0x07  "Tape Cartridge Removed"
0x08  "Compare Error"
0x09  "Copy Data Error"
0x0A  "Copy Management Error"
0x0B  "File Mark Error"
0x0C  "Head Servo Error"
0x0D  "Illegal Command"
0x0E  "Illegal Copy"
0x0F  "Illegal Length"
0x10  "Inappropriate request"
0x11  "Latch Error"
0x12  "No Cartridge"
0x13  "Not Loaded"
0x14  "Power-on request"
```

```
0x15  "QIC; no data detect."
0x16  "Read after Write Error"
0x17  "Read EOM logical"
0x18  "Read EOM physical"
0x19  "Reserved conflict"
0x1A  "Sensor Error"
0x1B  "Tape runout"
0x1C  "Unit Attention"
0x1D  "Write EOM warning"
0x1E  "Write EOM"
0x1F  "Cartrdige Write Protect"
0x20  "16 rewrite errors"
0x21  "24 rereads; block found"
0x22  "24 rereads; block not found"
0x23  "Illegal Copy Function"
0x24  "Illegal Header"
0x25  "No Header"
0x26  "Too Large Address"
0x27  "Bad ID or LUN"
0x28  "Partial Description"
0x29  "Bad Target Status"
0x2A  "Check Condition"
0x2B  "Data Transfer Error"
0x2C  "Selection Failure"
0x2D  "Sequence Error"
0x2E  "Illegal Block Size"
0x31  "Buffer Size Error"
0x32  "Rereads Outside Limits"
0x33  "Rewrites Outside Limits"
0x34  "Buffer Error Low Nibble"
0x35  "Buffer Error High Nibble"
0x36  "Unspecified Fatal Error"
0x37  "Timeout Error"
0x38  "Buffer RAM Error"
0x39  "Drive contr. Error"
0x3A  "EEPROM Verification"
0x3B  "EEPROM Error"
0x3C  "External RAM Error"
0x3D  "SCSI contr. Error"
0x3E  "Spurious interrupt"
0x3F  "Stack Overflow"
```

A.2 Disk Drive Subsystem

As the disk drive diagnostics execute commands, they poll the status return values from the Xylogics Disk Drive Controller Board. Any anomalies are reported to the user:

A.2.1 Xylogics Controller Error Codes (hex)

```
CODE  DESCRIPTION
01  Interrupt Pending
03  Busy Conflict
04  Operation Timeout
05  Header not Found
06  Hard ECC error
07  Illegal Cylinder Address Error
0A  Illegal Sector Address
0D  Last Sector too small
0E  Slave ACK Error (non-existent memory)
12  Cylinder and Head/Header Error
13  Seek retry required
14  Write protect error
16  Drive not ready
17  Sector count zero
18  Drive Faulted
19  Illegal Sector Size
1A  Self Test A Error
1B  Self Test B Error
1C  Self Test C Error
1E  Soft ECC Error
1F  Soft ECC Error Recovered
20  Illegal Head Error
21  Disk Sequencer Error
25  Seek Error
```

A.3 Ethernet Subsystem

Possible error codes returned by the Excelan Controller running the self-test are listed below:

A.3.1 Excelan Controller Error Codes (hex)

```
0x70   Transceiver fuse failed or not present
0x71   Transceiver loopback test failed
0x72   Negative 12 volt supply failed
0x91   Positive 12 volt supply failed
0xa0   Invalid Address for config message
0xa4   Invalid operation mode parameter
0xa5   Invalid host data format test pattern
0xa7   Invalid configuration message format
```

0xa8	Invalid movable data block parameter
0xa9	Invalid number of processes parameter
0xaa	Invalid number of mailboxes parameter
0xab	Invalid number of address slots parameter
0xac	Invalid number of hosts parameter
0xad	Invalid host queue parameter
0xae	Improper objects allocation
0xaf	Net boot failed
0xb0	Checksum on NX300 EPROM failed
0xb1	Memory test failed for 0-128k
0xb2	Memory test failed for 128k up to the highest address
0xb3	Counter test failed
0xb4	Interrupt test failed
0xb5	Transmission test failed
0xb6	Reciever test failed
0xb7	Local loopback data path test failed
0xb8	CRC test failed
0xb9	Checksum on physical address EPROM failed
0xba	Bus timeout
0xbb	Ethernet chip initialization failed
0xbc	Ethernet chip self-test failed
0xbd	Ethernet chip resource counter failed
0xbe	External loop-back test alignment error
0xbf	iSBX board not in place
0xc0	Specified time exhausted
0xc1	Host memory read/write test failed
0xc8	Parity hardware logic failed
0xc9	NMI interrupt for bus timeout failed
0xca	Host interrupt test failed
0xcb	Command unit test failed
0xcc	Divide error exception
0xcd	Undefined interrupt type
0xce	Command not executed by the CU of the 82586
0xcf	Command block sync failed between HW and SW
0xf0	Crossed 80286 segment boundary

A.4 Terminal Subsystem

The two–digit hex code identifies which part of the test failed. The error codes range in value from 81 hex to B1 hex and can also be found in the Host Adapter Technical Manual no. 80–000136–8–00, Revision A., Appendix D. The special error code value of 01 implies that the self–test never started, indicating a catastrophic Host Adapter error such as power failure, CPU failure, etc., or that Host Adapter/Host communications have failed.

A.4.1 Systech Controller Error Codes (hex)

CODE	DESCRIPTION
0x80	No HPS error
0x81	Rom checksum error
0x82	Stack data test error
0x83	Stack address test error
0x84	Stack checkerboard test error
0x85	Zero stack test error
0x86	Stack addressing conflicts error
0x87	Watchdog timeout test error
0x88	No real time clock present
0x89	Bad clock pulse width
0x8b	Dynamic RAM address test error
0x8c	Dynamic RAM checkerboard test error
0x8d	Zero dynamic RAM test error
0x8e	High-speed serial reset state test error
0x8f	High-speed serial DMA asynchronous lpbk err
0x90	High-speed serial interrupt-driven async. lbpk err
0x92	Network RAM test error
0x93	Network RAM location addressability test error
0x94	Network RAM checkerboard test error
0x95	Zero network RAM test error
0x96	COM 9026 test: status register incorrect value
0x97	COM 9026 test: network ID DIP switch set to zero
0x98	COM 9026 test: network interrupt test error
0x99	68440 DMA chip detected error
0x9f	PANIC error
0xb0	Network interrupt error:an interrupt occurred when not expected
0xb1	A COM 9026 interrupt occurred but the interrupt was not enabled
0xb2	The POR bit of the COM 9026 is not set during an interrupt
0xb6	Dynamic RAM parity error
0xb7	Watchdog timeout error
0xb8	Watchdog timeout occurred too early in the self-test sequence
0xc0	Host to HPS I/F test: data wrap test error
0xc1	Host to HPS I/F test: flag byte interrupt test error
0xc2	Host to HPS I/F test: host interrupt bit will not reset
0xc4	Host to HPS I/F test: data returned != data send
0xc5	Host to HPS I/F test: host not ready for more data error
0xf0	Bus exception error

```
0xf1    Address exception error
0xf2    Illegal instruction error
0xf3    Interrupt exception error
0xf4    Trap exception error
0xf5    Other exception error
0xff    Initialization complete
```

A.4.2 Adaptor Hardware Error Codes

```
80   Local memory error
81   Parameter latch empty error
82   Parameter latch full error
83   PROM checksum error
84   Parameter/FIFO test error
85   FIFO RAM test error
86   Tape side test error
87   Underrun test error A
88   Underrun test error B
89   Overrun  test error A
8A   Overrun  test error B
8B   Tape prescale test error A
8C   Tape prescale test error B
8D   Tape prescale test error C
8E   Bus  prescale test error A
8F   Bus  prescale test error B
90   Bus timeout hardware error
91   Parameter latch hardware error
```

Command Index

Symbols

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

X

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

W

X

Z

Dear Customer:

To do a better job of serving you by providing improved documentation, we are asking you to help us. Please take a few minutes to answer the following questions and return them to us by folding up and taping this questionnaire. We value your comments and suggestions and appreciate the time you take to send them to us. Thank you.

1. What is your position/function? ☐ application user ☐ system manager/administrator ☐ programmer ☐ service engineer ☐ other

2. How many years have you been working with computer systems? _____

3. What system are you currently working on? (please check both **hardware** and *software*) ☐ **GP1000** ☐ **TC2000** ☐ *Mach 1000* ☐ *pSOS 1000* ☐ *nX* ☐ *pSOS+m*

4. What application are you using the product for? _____

5. How much of your time do you spend reading or referring to documents? ☐ 10% ☐ 20% ☐ 30% ☐ 40% ☐ 50%

Please answer the following questions about: TC2000 Diagnostic Guide

6. Is the information accurate? ☐ always ☐ mostly ☐ seldom ☐ never

7. Is the material clear and logical? ☐ always ☐ mostly ☐ seldom ☐ never

8. Is it easy to find the information you need? ☐ always ☐ mostly ☐ seldom ☐ never

9. Does the index contain the words you look up? ☐ always ☐ mostly ☐ seldom ☐ never

10. Is the information located where you expect it in the book? ☐ always ☐ mostly ☐ seldom ☐ never

11. Are the illustrations and examples adequate? ☐ always ☐ mostly ☐ seldom ☐ never

12. Did you need information not available in this book or set of books? ☐ always ☐ mostly ☐ seldom ☐ never

13. What sections of the book did you use the most?

14. What areas need more or better examples? (Please list page numbers.)

_____

May we contact you for more information? If so, please give your name, address, and telephone number.

Do you have additional comments? If so, please write them on the reverse side.

After completing the form, **fold this end up to the dotted line**, fold down and tape the top, stamp, and mail. Thank you.

Additional Comments:

Documentation Department
BBN Advanced Computers Inc.
10 Fawcett St.
Cambridge, MA 02138

Tape here