CONVEX VMEbus DAT/3480 Tape Subsystem (*dev_vscsit*) Diagnostics Manual

Document No. 760-003530-001

First Edition, Rev. 1
September 1991

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX VMEbus DAT/3480 Tape Subsystem
(dev_vscsit) Diagnostics Manual
Order No. DHW-247
First Edition, Rev. 1

Printed in the United States of America

Replacement Instructions for *CONVEX VMEbus DAT/3480 Tape Subsystem (dev_vscsit) Diagnostics Manual*

This addendum updates the *CONVEX VMEbus DAT/3480 Tape Subsystem (dev_vscsit) Diagnostics Manual, First Edition*, Document No. 760-003530-000, as follows:

- It includes information on the optional automatic cartridge loader (ACL) mechanism.
- The *CONVEXVMEbus DAT/3480 Tape Subsystem (dev_vscsit) Diagnostics Manual* document number changes to Document No. 760-003530-001.
- The *CONVEX DAT/3480 Tape Subsystem (dev_vscsit) Diagnostics Manual* changes from First Edition to First Edition, Rev. 1.

Please update your document as indicated in the following table.

Remove Pages	**Insert Pages**
title and copyright pages	title and copyright pages
Revision/Update Information	Revision Information
Table of Contents	Table of Contents
1-1 through 1-4	1-1 through 1-4
4-1 through 4-2	4-1 through 4-2
4-11 through 4-56	4-11 through 4-60
Appendix A	(deleted)
Index	Index
Reader's Forum	(deleted)

Place this sheet after the title and copyright pages for future reference.

Revision Sheet

CONVEX VMEbus DAT/3480 Tape Subsystem
(dev_vscsit) Diagnostics Manual

Edition	Document No.	Date	Description
First, Rev. 1	760-003530-001	September 1991	Added subtests to test the auto cartridge loader. Removed Appendix A, "Reporting Problems."
First	760-003530-000	May 1991	

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Tables

List of Figures

Preface

Purpose and Intended Audience

This manual explains how to run the *dev_vscsit* diagnostic, which checks the CONVEX VMEbus 3480-Compatible Tape system, CONVEX Digital Audio Tape (DAT) Drive system and the VMEbus Small Computer System Interface (SCSI) host adapter. This document is not a tutorial, but rather a reference for the users of the *dev_vscsit* diagnostics, including field service and manufacturing test personnel, as well as the diagnostics sustaining staff. In addition, CONVEX customers can use this manual to execute the *dev_vscsit* diagnostic.

Scope

This manual applies to all CONVEX computers.

Organization

This document consists of the following:

- **Chapter 1. Diagnostics Environment**—Introduces the theories and concepts that underlie I/O diagnostics on CONVEX machines as well as the basic overview, philosophy, and structure of I/O diagnostics.
- **Chapter 2. EGOS Overview**—Provides a brief overview of the Event Governed Operating System (EGOS) and how it relates to device and peripheral diagnostics testing.
- **Chapter 3. Dshell Overview**—Provides a brief overview of and a general introduction to the *dshell* utility.
- **Chapter 4. VMEbus DAT/3480 Tape Subsystem Test (*dev_vscsit*)**—Describes how to operate the diagnostic, including prerequisites, test invocation, hardware initialization sequence, and class descriptions. It also describes the interactive debugger commands, explains how to use a test script, and provides SCSI host adapter and tape system status and error information and examples.
- **Appendix A. Reporting Problems**—Provides an example of the CONVEX *contact* utility for reporting minor software and hardware problems.

Notational Conventions

The notational conventions used in this text are listed below:

- Bit numbering is left to right, `N-1` through `0`. The most significant numerical bit is `N-1`, the least significant `0`. The bit numbering represents the binary weight of a position.
- Bit fields are specified using the following convention: *name*<*x..y*> where the bit field is *name* from bits *x* through *y*.
- Individual bit positions within a register are denoted by specific positions separated by commas. For example, REG<15,4,0> denotes bits 15, 4, and 0 of REG.
- Byte numbering is from left to right
- A *bit* is a single binary value or entity
- A *nibble* is 4 bits
- A *byte* is 8 bits
- A *halfword* is 16 bits
- A *word* is 32 bits
- A *longword* is 64 bits
- *Single precision* is a 32-bit floating point word
- *Double precision* is a 64-bit floating point longword
- An *instruction* is a multihalfword operand
- A bit is *set* when it contains a binary value of `1`.
- A bit is *clear* when it contains a binary value of `0`.
- All memory and I/O addresses are written in hexadecimal notation unless explicitly stated otherwise.
- All register contents are written in hexadecimal notation unless explicitly stated otherwise.
- A *register* is a programmer-visible hardware storage element internal to the processor
- *Physical memory* is the physical storage installed in the processor
- *Virtual memory* is the perceived amount of physical memory as seen by the application programmer
- The symbol *K* is an abbreviation for *kilo* or 1,024
- The symbol *M* is an abbreviation for *mega* or 1,048,576
- The symbol *G* is an abbreviation for *giga* or 1,073,741,824
- A *stack* is a linked-list group of words useful for dynamic allocation and deallocation of memory
- A *return block* is a collection of registers that is pushed or popped from a context stack in response to an instruction or other event
- *Reserved* or *undefined* convey what to expect, if anything, from unused fields in registers, reserved memory, or reserved I/O space. Algorithm implementation based on the use of undefined or reserved fields is not recommended.

Warnings

The following are examples of warnings, cautions, and notes and their typical content as used in CONVEX documents:

WARNING

Warnings highlight procedures or information necessary to avoid injury to personnel. A warning immediately precedes the critical information and includes a description of the hazard.

CAUTION

Cautions highlight procedures or information necessary to avoid damage to equipment, loss of data, or invalid test results. A caution immediately precedes the critical information and includes a description of the possible damage.

NOTE

Notes highlight useful information that is supplemental in nature. A note may immediately precede or follow the information that is being highlighted.

Associated Documents

The following is a partial list of other manuals or books that may provide more detailed information on the topics presented in this manual:

- *CONVEX Processor Diagnostics Manual (C1, C120)*, Order No. DHW-071
- *CONVEX Processor Diagnostics Manual (C200 Series)*, Order No. DHW-081
- *CONVEX Architecture Reference*, Order No. DHW-005
- *CONVEX SPU UNIX Utilities Manual*, Order No. DHW-021
- *CONVEX Processor Operation Guide (C100 Series, C200 Series)*, Order No. DHW-015
- *CONVEX Diagnostic Utilities Manual (C1, C120)*, Order No. DHW-072
- *CONVEX Diagnostic Utilities Manual (C200 Series)*, Order No. DHW-082
- *CONVEX UNIX Tutorial Papers*, Order No. DSW-002
- *The C Programming Language*, Kernighan & Ritchie, Order No. DSW-046

Ordering Documentation

To order the most current version of this or any other CONVEX document, use the product number. If the product number is not known, order by the exact title. In some situations, the most current version may not be desired. To receive a specific version of a manual, order the manual by its document number, or part number, which can be obtained by contacting the local CONVEX office or by calling the Technical Assistance Center.

The product number for this manual is DHW-247.
The document number for this manual is 760-003530-000.

CONVEX documents can be ordered by mail by sending a request to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

Technical Assistance

Hardware and software support can be obtained through the CONVEX Technical Assistance Center (TAC):

- From all locations in the continental United States, call 1(800)952-0379.
- From locations in Alaska, Hawaii, and Canada, call 1(214)497-4379.
- From all other locations, contact the nearest CONVEX office.

Reader's Forum

If you wish to mail your comments to us, please use the form at the end of this manual and list the document page number with your questions and comments. Thank you.

Chapter 1
Diagnostics Environment

1.1 Overview

CONVEX system diagnostics consist of a suite of test programs designed (except where noted) to execute under the Service Processor operating system, SPU UNIX. These programs utilize the capabilities of the Service Processor to test the operation of one or more of the functions of the system and report any errors detected. All of the diagnostics in this manual are intended to be executed "off-line"; that is, while ConvexOS is not being executed by any of the Central Processing Units (CPUs) in the system.

The Service Processor, together with SPU UNIX, various diagnostic utilities, and the test programs, themselves, comprise the CONVEX diagnostic environment. This chapter describes the hardware and software components of this environment and is intended to provide the background necessary to fully utilize the capabilities of the CONVEX processor diagnostics.

For more information about the diagnostic environment refer to the Diagnostic Environment chapter in the *CONVEX Processor Diagnostics Manual (C200 Series)* or the *CONVEX Processor Diagnostics Manual (C1, C120)* depending on the architecture of the machine under test.

1.2 Test Program Naming Conventions

Test program names are in the form *cattypedevnn.suffix* where:

- *cat* is the subsystem being tested
- *type* is the type of test being performed, e.g., standalone, self-test, or offline functional test
- *dev* is the device being tested, e.g., disk, tape, or printer. This segment of the test program name is used *only* if the category is a device.
- *nn* is a CONVEX code used for distinguishing between test programs
- *suffix* is one of three program identifiers:
  - *.t* are programs that execute on SPU
  - *.x00* and *.rnn* are object files for different target processors other than the SPU. The target processor depends on the subject of the test. The test program name must have the test program category (*cat*) at the beginning of the name to determine the target processor.

1.2.1 Test Program Categories

Test program categories include those tests for the CPU, peripheral devices, I/O system, memory system, SPU, and entire system. For example, *cpu4041* is a CPU vector instruction test while *mem4000* is a memory system functional test. The following table lists test program categories:

Table 1–1, Test Program Categories

Test Category (*cat*)	Description
cpu	CPU subsystem related test
dev	Peripheral device test
io, idc, tli, hpi	I/O subsystem related test
mem	Memory subsystem related test
spu	SPU subsystem related test

1.2.2 Test Program Types

A test program type describes whether a test is a standalone test, self-test, kernel hardware test, or an offline or online functional test. See the following table for the numbering system and description of test program types:

Table 1–2, Test Program Types

Number (*type*)	Description
0	Standalone test
1	Self-test
2	Kernel hardware test
4, 5	Offline functional test

1.2.3 Test Program Device Types

Test programs will test disks, tapes, terminals, printers, and networks. See the following table for the numbering scheme and a description of the test program device types:

Table 1–3, Test Program Device Types

Number (*dev*)	Description
1	Disk
2	Tape
3	Terminal
4	Printer
5	Network

1.2.4 Examples of Test Program Names

The following table presents several examples using the naming conventions as described in Section 1.2, "Test Program Naming Conventions."

> **NOTE**
>
> In the following table, SOFF stands for Standard Object File Format.

Table 1–4, Example Test Program Names

Test Program Name	Description
cpu4041.t	SPU object code in *b.out* format for *cpu4041*
cpu4041.rnn	C210 or C220 machine object code in SOFF format (relocatable)
cpu4041.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
mem4000.t	SPU object code in *b.out* format for *mem4000*
mem4000.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
dev4100.t	SPU object code in *b.out* format for *dev4100*
dev4100.x00	IOP object code in *b.out* format

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2
EGOS Overview

2.1 Overview

This chapter provides an overview of the Event Governed Operating System (EGOS) and how it relates to device and peripheral diagnostics testing. There are three basic types of EGOS systems, one for each type of CCU. There is one for the Multibus interface, one for the VME interface, and one for the HIA interface. This chapter will explain the three types of EGOS systems and how EGOS is positioned within the overall operating system environment.

2.2 Purpose of EGOS for Diagnostic Testing

EGOS is basically a simple operating system that the device tests use to handle interrupts, schedule processes, and generally allocate and control IOP/VIOP resources. The diagnostics code uses both EGOS and the Message Based System (MBS) to manipulate test program control over to the CCU side of the test program. MBS is not a part of EGOS but rather a system that allows a common section of memory to be used as a message area between multiple processors. For more information on MBS, refer to the *CONVEX Guide to Writing Device Drivers.*

EGOS initially sets up interrupt tables, determines how many chassis there are, and initializes its windows and resource allocation tables.

2.3 EGOS for the Multibus Interface

EGOS for the Multibus interface supports event driven device drivers. The Multibus version of EGOS takes interrupts that are local to a CCU and channels those errors to the proper piece of code to handle the error. It basically supplies the error interrupt handlers for the CCU error interrupts. It also contains support routines to control allocation of the various CCU-related resources.

2.4 EGOS for HSP Interface, HSP EGOS

EGOS for the HSP interface supports event driven device drivers. The HSP version of EGOS is like the Multibus version. It takes interrupts that are local to a CCU and channels those errors to the proper piece of code to handle the error. It basically supplies the error interrupt handlers for the CCU error interrupts. It also contains support routines to control allocation of the various CCU-related resources.

2.5 EGOS for VME Interface, VIOP EGOS

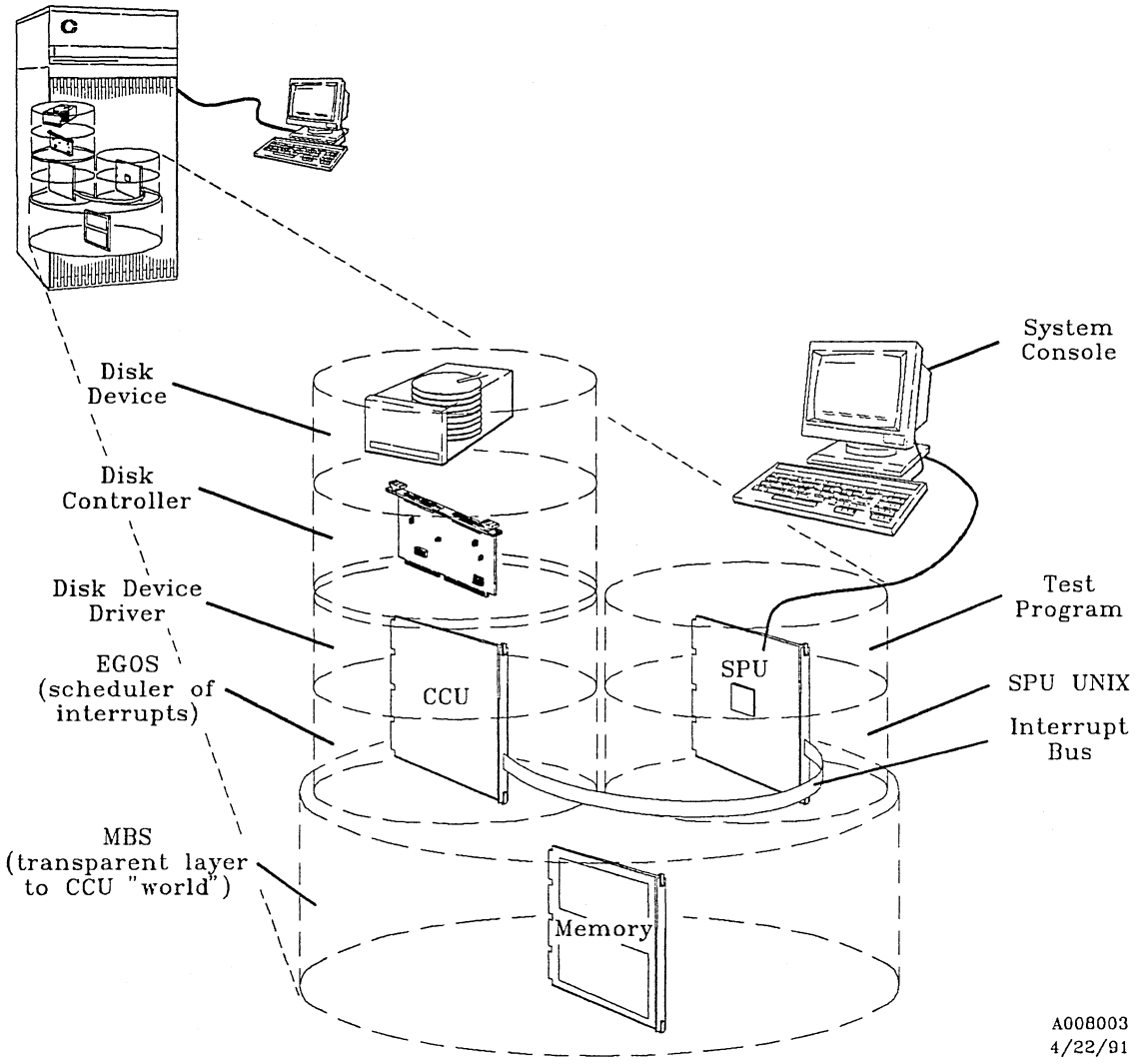
The VME interface version of EGOS is designed with a scheduler for the VIOP and is called VIOP EGOS. VIOP EGOS supports event driven device drivers as well as process type device drivers. VIOP EGOS utilizes a *sleep/wakeup* type of process control that improves efficiency of the device driver and makes it less complicated to create user written device drivers. Each process device driver has a priority level that can be defined relative to other processes. The scheduler supports 32 process priorities and is preemptive for higher priority processes. The VIOP hardware supports 14 device events for event driven device drivers. The 14 levels actually share 2 68020 interrupt levels. Therefore, two is the maximum number of processes at any given time.

2.6 EGOS Position in the Environment

EGOS is positioned in the operating environment between the actual device driver and MBS. MBS is a transparent layer that bridges the CCU and its resources to SPU UNIX. SPU UNIX handles many of the message manipulations that occur during testing. Many error messages that occur during diagnostics testing come from the device driver. When the device driver detects an error from the controller, it calls a routine in EGOS that places a message in the MBS system. This causes SPU UNIX to be interrupted and it retrieves the message from MBS. SPU UNIX then passes a signal to the test program. The test program then prints an error message to the console based on the code that it received.

The following figure illustrates the position of EGOS in the operating system environment.

Figure 2–1, EGOS' Position in the Environment



THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3
Dshell Overview

3.1 Overview

This chapter provides a brief overview of the *dshell* utility. Included in this overview is an overall explanation of the utility and a list of the utility's commands. For a complete description of this utility, refer to the Dshell chapter of the *CONVEX Diagnostic Utilities Manual (C200 Series)* or the *CONVEX Diagnostic Utilities Manual (C1, C120)* depending on the architecture of the machine under test.

3.2 Diagnostic Shell (*dshell*) Overview

The Diagnostic Shell (*dshell*) is a command interface program that runs on the Service Processor. Most of the diagnostics available for the CONVEX machines are interfaced through the *dshell*. Certain peripheral diagnostics are run as standalone tests. To determine whether a test can be run under the *dshell*, consult the appropriate chapter in this manual.

The *dshell* has two basic functions:

- Selecting diagnostics for execution
- Selecting test options
  - Pause on a failure or at the beginning or end of any specific subtest
  - Loop on a specific type of subtest or on a given set of subtests
  - Select subtest execution order
  - Direct test output to a file or to the screen (or both) to monitor the test as it runs or to analyze test results later
  - Select long or short error messages, or turn messages off
  - Execute either user-created or predefined command scripts

The following table list the various *dshell* commands and their functions.

Table 3–1, *dshell* Commands

COMMAND	FUNCTION
! [command]	This command is used to access, or *fork* a UNIX shell to execute the command that follows *!*.
exit	The *exit* command causes immediate termination of the *dshell* process and any test processes that may have been forked.
quit	The *quit* command causes immediate termination of the *dshell* process and any test processes that may have been forked.
^C	Returns user to the *dshell* command level if no subtest is running.
^B	Immediately terminate the *dshell* and any associated active processes. Core is dumped.
help	The *help* command causes a standard *help* menu to be displayed. The menu describes the correct command syntax for each *dshell* command and gives a terse description of what each command does.
status	The *status* command generates a report on the current state of the *dshell* command options. This report gives the name of each flag, its current value, and an explanation of its current effect.
log [options]	The *log* command provides a mechanism for specifying the number of failures that will be allowed to occur before a test or subtest terminates execution.
loop [options]	The *loop* command causes the *dshell* to repeat the execution of a test or subtest.
msgs [options]	The *msgs* command enables or disables different levels of test, class, and subtest result messages.
pause [options]	The *pause* command returns program control to the *dshell* to the beginning, end, or failure of all or specific subtests.
test [options]	The *test* executes specific tests, and displays test, class, and subtest menus.

3.3 Syntax Help for *dshell* Commands

The syntax for each *dshell* command can be obtained by typing the command with no options and pressing <**CR**>. For example, by entering **loop** and pressing <**CR**>, the syntax help in the following figure will be displayed on the screen:

Figure 3–1, Syntax Help for the *loop* Command

```
: loop
Proper syntax is:

loop off (-s) (-t)                    :disables loop modes
loop -s nnn                           :loop on subtest nnn
loop -t                               :loop on test
```

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4
DAT/3480 Tape Subsystem Test (dev_vscsit)

4.1 Overview

The *dev_vscsit* test is a functional test for the CONVEX VMEbus 3480-Compatible Tape system, the CONVEX Digital Audio Tape (DAT) Drive system and the VMEbus Small Computer System Interface (SCSI) host adapter. The test is subdivided into separate classes that verify the functional integrity of the host adapter, the 3480-compatible tape controller and tape drive, the DAT drive; and tape motion and data transfer functions of the tape drive subsystems. In addition, *dev_vscsit* verifies the:

- functional ability of the SCSI host adapter to operate in the CONVEX VMEbus I/O environment, including main memory access and interrupt generation and detection.
- ability of the host adapter to detect anomalous conditions on the SCSI bus.
- operational integrity of the cable interface between the host adapter and the tape subsystems.
- ability of the tape subsystem and the automatic cartridge loader (ACL) to communicate and function properly. The ACL subtests are not supported on C100 Series computers, however.

> **NOTE**
>
> The *dev_vscsit* diagnostic uses the ConvexOS VMEbus Input/Output Processor (VIOP) driver that normally resides in */mnt/os* on the SPU disk drive. This VIOP driver file contains the tape subsystem driver used by the *dev_vscsit* diagnostic. If a VIOP driver exists in the same directory as the *dev_vscsit* diagnostic, the diagnostic will use this driver and not the driver in */mnt/os/viop*.

Channel Control Unit (CCU) communications use the Event Governed Operating System (EGOS) and the Message Based System (MBS) used by ConvexOS. The intent is to test the communication paths that are used in a normal operating environment.

Table 4–1 lists most of the SCSI interface commands, their opcodes, and command names:

Table 4–1, SCSI Interface Commands

Number			
1	**0**	**0x00**	**Test Unit Ready**
2	0	0x01	Rewind
3	0	0x03	Request Sense
4	0	0x08	Read
5	0	0x0a	Write
6	0	0x10	Write Filemarks
7	0	0x11	Space
8	0	0x12	Inquiry
9	0	0x15	Mode Select
10	0	0x19	Erase
11	0	0x1a	Mode Sense
12	0	0x1b	Unload
13	0	0x1c	Receive Diagnostic Results
14	0	0x1d	Send Diagnostic
15	6	0xcf	Load Display
16	6	0xcb	Select Cartridge
17	6	0xcc	Eject
18	6	0xcd	Load Magazine
19	6	0xce	Autoloader Mode Select

4.2 Related Documents

This test description is intended as a reference for users who have a good understanding of the host adapter and VIOP and for those who have a basic understanding of tape drives and magnetic tape recording principles. Specifically, this test description assumes familiarity with SCSI host adapter and tape drive subsystem functionality. Table 4–2 lists several documents that may provide additional information not contained within this test description:

Table 4–2, Related Documents

Document Title	Document Origin
Python DDS DAT TAPE DRIVE PRODUCT DESCRIPTION MANUAL	Archive
Fujitsu Cartridge Tape Drives Customer Engineering Manual	Fujitsu
Fujitsu Cartridge Tape Controller Customer Engineering Manual	Fujitsu
Rimfire 3510 SCSI Host Bus Adapter and Floppy Disk Controller	Ciprico

4.3 Required Equipment

Table 4–3 lists the required hardware depending on the type of machine under test:

Table 4–3, Hardware Requirements

C1, C120	C200 Series
MCU MAU SPU VIOP VBCU	Memory System[1] CPX SP2 VIOP PIA
3480 Drive[2] or DAT Drive[3]	3480 Drive[2] DAT Drive[3]

1 Memory System consists of a minimum of one pair of memory boards (one odd and one even).
2 3480-compatible tape subsystem
3 DAT drive subsystem

NOTE

A maximum of 16 CONVEX 3480-compatible tape drives or 16 DAT drives can be connected to a single VIOP.

4.3.1 Software Requirements

The *dev_vscsit* diagnostic is dependent on the following software revision levels:

- SPU UNIX V5.2 or later
- For C200 Series systems, System Diagnostics V3.4 or later

  For C100 Series systems, System Diagnostics V6.6 or later
- ConvexOS requirements for 3480-compatible tape and DAT drive systems:
  - ConvexOS V8.1 or later is required to operate a CONVEX 3480-compatible cartridge tape subsystem. ConvexOS V8.1 requires a system generation of the software drivers when a cartridge tape subsystem is installed.
  - ConvexOS V9.0 or later contains all the software drivers needed to operate a 3480-compatible subsystem.
  - ConvexOS V8.1 or later is required to operate a CONVEX DAT drive system. ConvexOS V8.1 and V9.0 require a system generation of the software drivers when a CONVEX DAT drive system is installed.

    Versions of ConvexOS later than V9.0 contain all the software drivers needed for a DAT drive system. This means a system generation is not required when a CONVEX DAT drive system is installed.

4.3.2 Setting the 3480-Compatible Formatter Unit Number

Refer to the *CONVEX 3480-Compatible Tape Drive Service Guide* for information on setting the formatter's unit number.

4.3.3 Setting the 3480-Compatible Tape Unit Logical Address

The following procedure changes the logical address of the tape unit from address 0 to address 1:

	Front Panel Operation	Tape Drive Display
1.	Press **RESET**	**NT RDYU**
2.	Press **UNLOAD**	***0**
3.	Remove the tape cartridge	
4.	Press and hold down both **UNLOAD** and **TEST**	**DIAGMODE**
5.	Press **START**	**SETTING**
6.	Press **TEST**	**70: S.L-A**
7.	Press **TEST**	**L-ADR:0**
8.	Press **START**	**L-ADR:1**
9.	Press **TEST**	**70:END**
10.	Press **START** multiple times until the display shows **89:WTROM**	**89:WTROM**
11.	Press **TEST**	**WTROM:Y**
12.	Press **TEST**	**89:END**
13.	Press **RESET** (twice)	**SELFTEST**

After the tape unit number is set, the display on the tape unit will show ***1**.

Refer to the *Fujitsu Cartridge Tape Drives CE Manual* Chapter 5, "Setting Method," for more information.

4.3.4 Setting the DAT Drive Unit Number

Refer to the *CONVEX Digital Audio Tape Drive Service Guide* for information on setting the DAT drive's logical unit number.

4.4 Test Invocation

The *dev_vscsit* test executes under the Diagnostic Shell (*dshell*) and supports all the features of the *dshell*. The *dshell* permits tests to be initiated in any order. To invoke the *dev_vscsit* test, use the procedure shown in Figure 4–1. User input is indicated in **boldface**. The prompts and responses appear sequentially on the screen, one line at a time. Figure 4–1 shows all the prompts and responses:

> **NOTE**
>
> Use the following test invocation sequence for the initial invocation of *dev_vscsit* or when the state of the machine is unknown. Also, the following invocation sequence should be used if any hard errors have occurred since the last system initialization.

Figure 4–1, Initial Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> sysreset RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
:test dev_vscsit [-c [class number(s)]] [-s [subtest number(s)]] [-dV] [-f FILE] [+>filename] RETURN
```

> **NOTE**
>
> After entering **dshell**, specific *dshell* parameters may be changed. Refer to the "Dshell Overview" chapter of this manual for more information.

Entering only **test dev_vscsit** executes all *dev_vscsit* subtests sequentially. Execute a specific class(es) of subtest(s) or one or more individual subtests by using the **-c** or **-s** options, respectively. Refer to chapter 3, "Dshell Overview," for more detailed information on using these options.

The following list defines the remaining options:

-d Enter debugger (no subtests are executed).

-V Print version string (compilation date of test–last modification date).

-f *File* Use *FILE* as the parameter save file. If this option is omitted, */tmp/dev_vscsit.tmp* is used as the parameter file.

The [+>*filename*] option appends the test results to *filename*.

NOTE

The following alternate test invocation procedure is optimal when invoking *dev_vscsit* multiple times. Using this invocation sequence ensures that the test is invoked and executed with all set-up parameters supplied when the test was last executed with the initial invocation sequence.

The only difference in this alternate invocation sequence is the **x** after **dev_vscsit**. When invoking *dev_vscsit* in this manner, no prompts are displayed. The diagnostic obtains all prompt information from the parameter file (default parameter file is */tmp/dev_vscsit.tmp*) created when the initial invocation sequence was performed. Also note that **mminit -s** is only required if the state of the machine is unknown or if hard errors have occurred since the last system initialization. Figure 4–2 shows the alternate test invocation sequence:

Figure 4–2, Alternate Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
:test dev_vscsitx [-c [class number(s)]] [-s [subtest number(s)]] [-dV] [-f FILE] [+>filename] RETURN
```

4.4.1 Test Parameter Menu

Once the test is invoked, a test menu prompt is presented allowing selection of default switches. Figure 4–3 shows the TEST PARAMETER MENU with all prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )):

Figure 4–3, Test Parameter Menu

```
                    ENTER TEST PARAMETERS

        []      Encloses allowed input ranges or values
        ()      Encloses the default value
        ^       Returns to the previous prompt
        :nn     Returns to the prompt # nn
        :       Returns to the first unsatisfied prompt
        :?      Reviews previous entries
        ?       Provides additional help for each question

1: Select ioconfig file [<filepath>,?]           (/ioconfig) -> RETURN

                    PERIPHERAL CONFIGURATION DATA
            CCU    Chassis   Type    CSR    Int Unit   Type
        -------- ------- -------- ------ --- ---- --------
         1) viop  3     1    MTC-202  0xee00  3   0.0 MTD-207
         2) viop  3     1    MTC-202  0xee00  3   0.1 MTD-207
         3) viop  3     1    MTC-202  0xea00  4   0.0 MTD-208
         4) viop  3     1    MTC-202  0xea00  4   1.0 MTD-208

         *** Enter 0 for manual configuration ***

2: Ioconfig File Unit(s) to Test [1-4,0,?]              (1) -> RETURN
3: Use Defaults for Remaining Parameters [y,n,?]        (y) -> RETURN
4: Enter OK, or :NN to return to question NN [OK]      (OK) -> RETURN
```

The prompts and responses in the figure appear sequentially on the screen, one line at a time. The figure illustrates *most* questions that can be displayed during test parameter input. However, some questions may be omitted, depending on answers to previous questions. In all cases, questions are numbered sequentially. The numbers displayed on the screen during testing may not correspond to those shown in the example, as the questions illustrated are examples only.

For help or information during test parameter entry, enter one of the help characters followed by a RETURN. Table 4–4 list the help characters:

Table 4–4, Getting Help During Test Parameter Entry

Character	Description
:?	Reviews previous entries
?	Provides specific help where available

After displaying the desired help information, the system redisplays the last prompt.

If **OK** or RETURN is entered, the test parameter menu terminates and all inputs are no longer changeable.

After all the prompts have been answered, the screen displays a TEST PARAMETER SUMMARY that displays prompts that were answered and their responses. Figure 4–4 illustrates an example of an initial TEST PARAMETER SUMMARY screen, but the actual values and responses vary according to

the input. Figure 4–4 shows a sample `Test Parameter Summary` after a system reset and mminit have been performed. In this case, the CCU drivers (*/mnt/os/viop*) will be loaded, probed, and attached.

> **NOTE**
>
> Refer to Figure 4-1 for the test invocation sequence to produce this TEST PARAMETER SUMMARY.

Figure 4–4, Test Parameter Summary (CCU Never Loaded)

```
                TEST PARAMETER SUMMARY

Select ioconfig file                                                : /ioconfig
Ioconfig File Unit(s) to Test                                       : 1

          PERIPHERAL CONFIGURATION DATA
  CCU    Chassis   Type    CSR   Int Unit   Type
-------- ------- -------- ------ --- ---- --------
viop  3     1    MTC-202  0xee00  3   0.0 MTD-207

Use Defaults for Remaining Parameters                               : y
Enter OK, or :NN to return to question NN                           : OK

Saving options in parameter file ''/tmp/dev_vscsit.tmp'' ... Done
Initializing MBS processor queues ... Done
Loading CCU(s) ... Done
CCU 3/MTC-202 configure/probe completed
Tape unit 0.0 attach and connect completed
Main memory data buffer address starts at 0x00202000
```

If *dev_vscsit* has been executed previously and *sysreset* and *mminit* have not been executed since, the CCU drivers are still loaded and a different TEST PARAMETER SUMMARY will be displayed. Figure 4–5 shows a sample TEST PARAMETER SUMMARY with the CCU drivers already loaded:

> **NOTE**
>
> Refer to Figure 4-2 for the test invocation sequence to produce this TEST PARAMETER SUMMARY.

Figure 4–5, Test Parameter Summary (CCU Previous Loaded)

```
                TEST PARAMETER SUMMARY

Select ioconfig file                                                    : /ioconfig
Ioconfig File Unit(s) to Test                                           : 1

          PERIPHERAL CONFIGURATION DATA
  CCU    Chassis   Type     CSR    Int Unit   Type
-------- ------- -------- ------ --- ---- --------
viop  3     1    MTC-202  0xee00  3   0.0 MTD-207

Use Defaults for Remaining Parameters                                   : y
Enter OK, or :NN to return to question NN                               : OK

Saving options in parameter file ''/tmp/dev_vscsit.tmp'' ... Done
Initializing MBS processor queues ... Done
*** CCU driver already loaded on VIOP 3
*** Tape unit 0.0 already attached and connected
Main memory data buffer address starts at 0x00202000
```

4.5 Initialization Sequence for *dev_vscsit*

After the last prompt is entered and before subtest code execution, the following events occur:

- The largest contiguous main memory space after the first 2 Mbytes is located and reserved for use by *dev_vscsit*.
- The diagnostic checks to see if the CCU is already loaded with the *dev_vscsit* CCU driver. If the driver is not loaded, the CCU is reloaded. After the load completes, the driver is configured for EGOS and the EGOS probe message starts the driver.
- The CCU driver is passed the current test parameters.

> **NOTE**
>
> The file */mnt/boot_db* is used to determine what memory is installed. If this file is nonexistent, it can be created by entering: **scn_util -b > /mnt/boot_db** from the `(spu)>` prompt.

After all the above events have occurred, the test code is started.

4.5.1 Prompt Explanations

The test parameter prompts are repeated and explained in the following paragraphs.

```
1: Select ioconfig file [<filepath>,?]          (/ioconfig) ->
```

This option allows you to specify an alternate */ioconfig* file. The file must still be in the same format as a conventional */ioconfig* file.

```
                PERIPHERAL CONFIGURATION DATA
     CCU     Chassis   Type     CSR    Int Unit   Type
  --------- ------- -------- ------ --- ---- --------
1) viop  3     1    MTC-202  0xee00  3   0.0 MTD-207
2) viop  3     1    MTC-202  0xee00  3   0.1 MTD-207
3) viop  3     1    MTC-202  0xea00  4   0.0 MTD-208
4) viop  3     1    MTC-202  0xea00  4   1.0 MTD-208

The digit to the left of the period under the Unit heading is the SCSI
target identification number and has a value range from 0 to 6. SCSI
identification number 7 is reserved for the host adapter. The digit to the
right is the tape unit logical address number and has a value range from 0
to 3 for 3480-compatible tape drives (DAT drives allow 0 only).

        *** Enter 0 for manual configuration ***
2: Ioconfig File Unit(s) to Test [1-4,0,?]              (1) ->
```

The applicable */ioconfig* file entries are listed (if any). To select a predefined controller configuration entry from the */ioconfig* file, enter the number to the left of the desired entry. Entering a **0** allows each item within the controller configuration entry to be independently specified. If you want to use most of the items associated with a given entry in the */ioconfig* file, select the number for that entry, back up to this prompt (via the **CTRL** command), and then enter **0** the second time. The default values for the independent items will then be the same as the originally selected entry.

```
3: Use Defaults for Remaining Parameters [y,n,?]        (y) ->
```

This option lets you use the default values for the remaining questions.

```
4: Enter OK, or :NN to return to question NN [OK]      (OK) ->
```

This option lets you return to a specified question number and change the answer.

Figure 4–6 shows all the options for the TEST PARAMETER SUMMARY:

Figure 4–6, Test Parameter Menu (with All Options)

```
                    ENTER TEST PARAMETERS

        []      Encloses allowed input ranges or values
        ()      Encloses the default value
        ^       Returns to the previous prompt
        :nn     Returns to the prompt # nn
        :       Returns to the first unsatisfied prompt
        :?      Reviews previous entries
        ?       Provides additional help for each question

 1: Select ioconfig file [<filepath>,?]            (/ioconfig) -> RETURN

                    PERIPHERAL CONFIGURATION DATA
             CCU    Chassis   Type    CSR   Int Unit   Type
          -------- ------- -------- ------ --- ---- --------
          1) viop  3     1    MTC-202  0xee00  3   0.0 MTD-207
          2) viop  3     1    MTC-202  0xee00  3   0.1 MTD-207
          3) viop  3     1    MTC-202  0xea00  4   0.0 MTD-208
          4) viop  3     1    MTC-202  0xea00  4   1.0 MTD-208

          *** Enter 0 for manual configuration ***

 2: Ioconfig File Unit(s) to Test [1-4,0,?]                (1) -> RETURN
 3: Use Defaults for Remaining Parameters [y,n,?]          (y) -> n
 4: Enable Debug Monitor [y,n,?]                           (n) -> y
 5: Run Tape Tests to EOT [y,n,?]                          (n) -> RETURN
 6: Variable record subtest pattern [<hexadecimal pattern>,?]
                                                      (0x6db6) -> RETURN
 7: Variable record miscompare Line Dump Count  [1-100, ?](8) -> RETURN
 8: Use Autoloader for Single Tape Subtests [y,n,?]        (n) -> RETURN
 9: Maximum Cartridges Used in Autoloader Tests [1,10,?]   (2) -> RETURN

  **  Printer On/Off Enable/Disable Options (bit mapped)  **
    0x0001:  Enable printer ON string before error
    0x0002   Enable printer OFF string after error

10: Select Printer On/Off Mode [0x0-0x3,?]              (0x0) -> 3
11: Printer On Character Sequence (before error)
    [<printer on string>,?]                       (\033[?5i) -> RETURN
12: Printer Off Character Sequence (after error)
    [<printer off string>,?]                      (\033[?4i) -> RETURN
13: Enter OK, or  :NN to return to question NN [OK]      (OK) -> RETURN
```

Figure 4–7 shows the TEST PARAMETER SUMMARY menu with online help for each option:

Figure 4–7, Test Parameter Menu (with Help Fields)

```
                        ENTER TEST PARAMETERS

        []      Encloses allowed input ranges or values
        ()      Encloses the default value
        ^       Returns to the previous prompt
        :nn     Returns to the prompt # nn
        :       Returns to the first unsatisfied prompt
        :?      Reviews previous entries
        ?       Provides additional help for each question

 1: Select ioconfig file [<filepath>,?]            (/ioconfig) -> ?
_____________________________ HELP FOLLOWS _____________________________
    Specify an alternate "/ioconfig" file. The file must still be in
    the same format as a conventional "/ioconfig" file
------------------------------ END OF HELP ------------------------------

                    PERIPHERAL CONFIGURATION DATA
              CCU    Chassis   Type    CSR    Int Unit   Type
            -------- ------- -------- ------ --- ---- --------
          1) viop  3     1    MTC-202  0xee00  3   0.0 MTD-207
          2) viop  3     1    MTC-202  0xee00  3   0.1 MTD-207
          3) viop  3     1    MTC-202  0xea00  4   0.0 MTD-208
          4) viop  3     1    MTC-202  0xea00  4   1.0 MTD-208

          *** Enter 0 for manual configuration ***

 2: Ioconfig File Units to Test [1-4,0,?]                  (1) -> ?
_____________________________ HELP FOLLOWS _____________________________
    Above, the applicable "/ioconfig" file entries are listed (if any).
    To select a predefined controller configuration entry from the
    "/ioconfig" file, enter the number which is found to the left of
    the desired entry. Entering a 0 allows each item within the
    controller configuration entry to be independently specified.  If
    you desire to use most (but not all) of the items associated with a
    given entry in "/ioconfig" file, select the number for that entry,
    backup to this prompt (via the ^ command), and then specify 0 the
    second time.  This will cause the default values for the
    independent items to be the same as the originally selected entry.
------------------------------ END OF HELP ------------------------------
 3: Use Defaults for Remaining Parameters [y,n,?]          (y) -> ?
_____________________________ HELP FOLLOWS _____________________________
    A ''Yes'' answer here will result in the selection of the default
    value for all remaining prompts.  This allows the user to avoid
    having to explicitly enter a <C/R> in response to every remaining
    prompt.
------------------------------ END OF HELP ------------------------------
 4: Enable Debug Monitor [y,n,?]                           (n) -> ?
_____________________________ HELP FOLLOWS _____________________________
    This option allows the user to automatically enter the interactive
    debugger any time an error is detected within a subset.  The
    debugger may also be invoked by specifying the "-d" option at test
    invocation time (i.e., dev_vscsit[x].t -d).
------------------------------ END OF HELP ------------------------------
```

Figure 4–7, Test Parameter Menu (with Help Fields) (continued)

```
 5: Run Tape Tests to EOT [y,n,?]                          (n) -> ?
_______________________________ HELP FOLLOWS ________________________________
    If this flag is asserted, tests that attempt to write to
    end-of-tape (EOT) will be executed. For DAT devices, this could
    take as long as 2 hours.
------------------------------ END OF HELP -------------------------------
 6: Variable record subtest pattern [<hexadecimal pattern>,?]
                                                        (0x6db6) -> ?
_______________________________ HELP FOLLOWS ________________________________
    Enter the hexadecimal data pattern to use for variable-record
    read/write subtests.  The pattern may be any length from 1 nibble
    to 256 bytes.  The pattern will be replicated over and over as
    necessary when formatting buffers for tape writes.

Optional pattern files:
    The pattern can be specified via a file on the SPU disk.  For this
    method enter a "+" followed by the optional filename of the pattern
    file.  If the filename is omitted, the test will default to the
    file "dev_vscsit.pat".  The pattern can contain whitespace and can
    be spread across multiple lines if desired.

*** NOTE:
    The buffer fill pattern before reads is "0x55aa55aa". Therefore,
    this pattern should be avoided as the write/read data pattern.
------------------------------ END OF HELP -------------------------------
 7: Variable record miscompare Line Dump Count  [1-100, ?](8) -> ?
_______________________________ HELP FOLLOWS ________________________________
    Enter the MAXIMUM number of display lines to print when a data
    miscompare occurs.
------------------------------ END OF HELP -------------------------------
8: Use Autoloader for Single Tape Subtests [y,n,?]            (n) -> ?
_____________________________HELP FOLLOWS_____________________________________
    If "yes", the autoloader will be requested to load a tape for the
    tape read/write subtests.  The first tape in the magazine will
    be used for these subtests.
--------------------------------- END OF HELP ----------------------------------
9: Maximum Cartridges Used in Autoloader Tests [1,10,?]       (2) -> ?
_____________________________HELP FOLLOWS_____________________________________
    This is the number of tapes that will be used to run the autoloader
    tests.  The default is 2 to shorten test times.
--------------------------------- END OF HELP ----------------------------------
```

Figure 4–7, Test Parameter Menu (with Help Fields) (continued)

```
  **  Printer On/Off Enable/Disable Options (bit mapped)  **
    0x0001:  Enable printer ON string before error
    0x0002   Enable printer OFF string after error

10: Select Printer On/Off Mode [0x0-0x3,?]              (0x0) -> ?
_____________________________ HELP FOLLOWS _____________________________
    This option allows the ability to send a user specified character
    string to the display before and/or after an error message has been
    displayed.  Although any string up to 64 characters may be used,
    its intended use is to turn a printer on before an error is
    displayed and turn it off after the error has been displayed.
    This is a paper-saving feature when running the test over and over
    for several hours.  Only errors will be printed when both the on
    and off strings are defined.  This assumes a printer is connected
    to the auxiliary port on the display terminal where the test is
    running and that the auxiliary port can be turned on and off via an
    escape character sequence.
------------------------------ END OF HELP ------------------------------
11: Printer On Character Sequence (before error)
    [<printer on string>,?]                          (\033[?5i) -> ?
_____________________________ HELP FOLLOWS _____________________________
    Enter the character string to be displayed before an error message
    is printed.  The default value is the "Enter Auto Print Mode"
    sequence for terminals that support vt100 terminal protocol.  This
    will turn on the echo of all displayed data to the terminal's
    auxiliary port.

    Control characters may be specified by using standard C programming
    style escape sequences.  For example:

            \033    -   Octal value of the ASCII escape character  (0x1b)
            \x1b    -   Same as above but specified in hex.
            \r      -   Carriage Return
            \n      -   Line Feed
            \t      -   Tab Character
            \b      -   Backspace character
            \f      -   Form Feed character
------------------------------ END OF HELP ------------------------------
```

Figure 4–7, Test Parameter Menu (with Help Fields) (continued)

```
12: Printer Off Character Sequence (after error)
    [<printer off string>,?]                          (\033[?4i) -> ?
_____________________________ HELP FOLLOWS _____________________________
    Enter the character string to be displayed after an error message
    has been printed.  The default value is the "Exit Auto Print Mode"
    sequence for terminals that support vt100 terminal protocol.  This
    will turn off the echo of all displayed data to the terminal's
    auxiliary port.

    Control characters may be specified by using standard C programming
    style escape sequences.  For example:

            \033    -   Octal value of the ASCII escape character  (0x1b)
            \x1b    -   Same as above but specified in hex.
            \r      -   Carriage Return
            \n      -   Line Feed
            \t      -   Tab Character
            \b      -   Backspace character
            \f      -   Form Feed character
------------------------------ END OF HELP ------------------------------
13: Enter OK, or  :NN to return to question NN [OK]       (OK) -> RETURN
```

4.6 Class Descriptions

The *dev_vscsit* test contains four classes of subtests as listed in Table 4–5:

Table 4–5, *dev_vscsit* Test Classes

Class	Description
1	SCSI host adapter tests
2	Tape controller and logical unit self-tests
3	Tape motion and read/write tests
4	Tape drive exception tests
5	Stacker mechanism tests

NOTE

When a subtest is executing, the elapsed time counter is not updated until the subtest has been completed.

4.7 Class 1 Subtests, SCSI Host Adapter Tests

Class 1 subtests only test the VMEbus SCSI host adapter. A target device does *not* need to be connected to the host adapter to run any of the class 1 subtests. Class 1 subtests verify the following functionalities:

- The ability to communicate with the VIOP and the VMEbus Control Unit (VBCU)
- The ability of the host adapter to interrupt and to mask interrupts
- Accessibility to main memory

Table 4–6 lists all Class 1 subtests, their descriptions, and the approximate times required to execute each subtest:

Table 4–6, Class 1 Subtests

Subtest	Description	Time (min:sec)[1]
100	Host Adapter Identify Test	00:01
101	Host Adapter RAM Test	00:10
102	Host Adapter PROM Test	00:30

[1] The times presented are approximated and are the same for 3480-Compatible subsystem and DAT subsystem.

4.7.1 Subtest 100, Host Adapter Identify Test

Subtest 100 issues an *IDENTIFY* command to the host adapter from a diagnostic parameter block that contains only a Target ID field (0xff) and a command field (0x05). The *IDENTIFY* command returns a specially-formatted status block that identifies the firmware version, engineering revision level, and the day, month, and year when the firmware in the Programmable Read Only Memory (PROM) was generated. The test formats the information returned from the host adapter and then displays it on the terminal. The following is an example of the formatted output from the Host Adapter Identify test:

```
          Subtest 100      0:00:00
Firmware rev = 07, Engr rev = 57, Prom generation date (mm/dd/yy) 04/02/90
                           0:00:00    passed
```

4.7.2 Subtest 101, Host Adapter RAM Test

Subtest 101 issues a *BOARD TEST* command to the host adapter from a diagnostic parameter block that contains the Target ID field (0xff), the command field (0x09), and the Test Flag field with the Static RAM Test (SRT) bit set. When the *BOARD TEST* command is completed, the results of the diagnostic are returned in a status block. This test is repeated 500 times. If the error byte in the status block equals 0x61, the memory test has failed. The following is an example of a failed Host Adapter RAM test:

```
Failed:  VME host adapter ram test
Error: (flag byte) Command completed with error status.
Error: (error byte) Static ram error.
     : Error addr xxxx expected yy found zz
```

NOTE

To avoid writing test patterns over data, the *BOARD TEST* command will not execute until all preceding commands have finished. While this command is running, the adapter will not accept other commands. At the end of the command, any pending channel attentions will be serviced and execution will resume.

4.7.3 Subtest 102, Host Adapter PROM Test

Subtest 102 issues a *BOARD TEST* command to the host adapter from a diagnostic parameter block that contains the Target ID field (0xff), the command (0x09), and the Test Flag field with the PROM Checksum Test (PCS) bit set. When the *BOARD TEST* command is completed, the results of the diagnostic are returned in a status block. This test is repeated 50 times. The PCS has failed when the error byte in the status block equals 0x62. The following is an example of a failed Host Adapter PROM test:

```
Failed:  VME host adapter prom test
Error: (flag byte) Command completed with error status.
Error: (error byte) Prom Checksum error.
```

4.8 Class 2 Subtests, Tape Controller and Logical Unit Self-Test

Class 2 subtests invoke the self-test capabilities of the 3480-compatible tape controller (formatter) and cartridge tape drive and the DAT drive. In this test class, self-test diagnostic commands are sent to the controller (formatter) and tape drives. The receive diagnostic results are then invoked. A tape controller (formatter), a 3480-compatible cartridge tape drive, and a 3480 scratch tape (or a DAT drive and a DAT scratch tape) are required for this class of tests.

CAUTION

During Class 2 subtests, do *not* load, unload, or reset the selected tape drive. Doing so will invalidate the test results.

Any data existing on the scratch tape will be lost during the execution of these subtests.

> **NOTE**
>
> If Subtest 201 is to be run, a tape cartridge needs to be loaded prior to starting Class 2 subtests.

Table 4–7 lists all Class 2 subtests, their descriptions, and the approximate times required to execute each subtest:

Table 4–7, Class 2 Subtests

Subtest	Description	3480 Time (min:sec)[1]	DAT Time (min:sec)[1]
200	Tape Controller Self-Test	00:56[2]	00:30
201	Tape Logical Unit Self-Test	15:00	n/a[3]

[1] The times presented are approximated.
[2] This test can run 00:56, 01:49, or 03:36 depending on the number of tape drives connected to the controller.
[3] This test does not run on the DAT drive subsystem.

4.8.1 Subtest 200, Tape Controller Self-Test

Subtest 200 issues a *SEND DIAGNOSTIC* command to request the tape controller (formatter) or DAT drive to perform diagnostic tests on itself. The SCSI command block containing the *SEND DIAGNOSTIC* command is sent to the host adapter using a standard parameter block. The SCSI command block is a group 0 type, and the opcode is 0x1d. The self-test (SLFTST), device offline (DEVOFL), and unit offline (UNITOFL) bits are set to 1, 1, and 1 for 3480-compatible tape controllers (formatters) and 1,0, and 0 for DAT drives, respectively, to direct the targeted tape controller (formatter) or DAT drive to perform a self-test. If there is no error, the command is terminated with good status. If an error is encountered in the test, the command is terminated with check condition status and the sense key is set to indicate a hardware error. The following functions are tested in the 3480-compatible tape controller (formatter):

- **Processor-to-processor communications**—The processor-to-processor information transfer FIFO is diagnosed for failure conditions.
- **Buffer test**—The buffer RAM, microprocessor interface, and the CRC generation circuitry are diagnosed.
- **Device digital logic test**—The digital portion of the device logic is diagnosed.

For 3480-compatible systems only, a *RECEIVE DIAGNOSTIC RESULTS* command (opcode 0x1c) is issued after the *SEND DIAGNOSTIC* command has completed. Refer to "Subtest 201, Tape Logical Unit Self-Test Test," for more information. The following is an example of a failed Tape Controller Self-Test:

```
        Subtest 200    0:00:00
  0:00:01   failed

*****  Mon Apr 16 19:07:03 1990  *****
Test:    dev_vscsit.t 1.1   Class: 2   Subtest: 200 1.1    Count: 1    Error: 0
Failed:  Controller self test


Error: (flag byte) Command completed with error status.

Error: (error byte) Bad status from scsi device.

Error: (SCSI stat byte) Check condition - error or exception event occurred.

Sense bytes:

[ 0-9 ] = 70 00 04 00 00 00 00 24 00 00
[10-19] = 00 00 44 00 23 00 00 00 02 2c

Sense information interpretation:

 70             Residual-length field is NOT valid
 00             Segment number = 0
 04             Sense Key = 4 (DRIVE HARDWARE ERROR)
 00000000       Residual length = 0 (NOT valid)
 24             36 additional sense bytes
 00 00 00 00    (Command-specific bytes)
 44 00          Internal target failure
 23             Primary FRU: FM PCB (formatter)
                Secondary FRU: CF PCB (core function)
 00 00 00       (Sense-key-specific bytes - NOT valid)
 02             Sense bytes 20-43 have contents of controller hardware regs
 2c             ERPA: Permanent equipment check
```

From the example above, the following are items of interest:

Request sense bytes[0-9] = 70 00 **04** 00 00 00 00 24 00 00

Request sense bytes[10-19] = 00 00 **44 00 23** 00 00 00 02 **2c**

- Sense key definitions are listed in Table dev_vscsit-18, "Sense Key Codes," on page 45 of this chapter.
- Request sense bytes 12 and 13—Additional sense code and sense code qualifiers (44 00 = internal target failure)

  Additional sense code and sense code qualifiers definitions are listed in Table dev_vscsit-19, "Additional Sense Codes and Descriptions," on pages 46 through 49 of this chapter.
- FRU code definitions are listed in "Table dev_vscsit-20, FRU Codes for Byte 14 Nibbles," on page 50 of this chapter.
- ERPA code definitions are listed on pages 50 through 54 of this chapter.

4.8.2 Subtest 201, Tape Logical Unit Self-Test Test

> **NOTE**
>
> This test does not run on the DAT drive subsystem.

Subtest 201 invokes a self-test on a selected cartridge tape logical unit. The *SEND DIAGNOSTIC* command is used to begin the cartridge self-test. The SCSI command block is a group 0 type and the opcode is 0x1d. The SLFTST, DEVOFL, and the UNITOFL bits are set to 0, 1, and 1, respectively, to direct the targeted tape controller (formatter) to perform a self-test on the tape logical address unit matching the Logical Unit Number (LUN) field of the SCSI command block.

The following are online diagnostic routine descriptions for the tape logical unit self-test:

- Routine 50 (LOOP WRITE TO READ level 1)

  Data is written into the data buffer and passed from the buffer through the tape controller (formatter) digital detection circuitry.
- Routine 51 (LOOP WRITE TO READ level 2)

  Data is written into the data buffer and passed from the data buffer to the tape drive. The tape drive returns the data to the tape controller (formatter) through both the analog and the digital check circuitry. No tape motion is required.
- Routine 52 (Write Data)

  The tape is positioned at the Load Point and 25 blocks of 100 bytes, 1 Kbyte, 32 Kbytes, and 64 Kbytes each are written to tape.
- Routine 53 (Read Data)

  The tape is positioned at the Load Point and 25 blocks of 100 bytes, 1 Kbyte, 32 Kbytes, and 64 Kbytes each are written to tape. All data blocks are then read in the reverse and forward directions.
- Routine 54 (Combination test 1)

  Various combinations of the *WRITE, READ, REVERSE, WRITE FILEMARK, ERASE,* and *SPACE* commands are tested.

- Routine 55 (Lifter test)

  The tape drive is set into special diagnostic mode and a write operation is performed on the medium. The tape drive diagnoses the tape drive lifter solenoid, which controls air pressure between the medium and the magnetic tape head.

- Routine 56 (Reserved)

  Routine 56 is reserved by the manufacturer and is not available for diagnostic use.

- Routine 57 (Combination test 2)

  An all-zeros pattern is replicated in the data buffer and blocks are written to tape until Logical End-of-Media (EOM) is detected. The first block written is 255 bytes in length. Each succeeding block length is incremented by one byte. All data is read in both forward and reverse directions.

If an error is encountered during the execution of a routine, a diagnostic result file is generated at that time and no further routine executions occur. This diagnostic result file is then retrieved by the *RECEIVE DIAGNOSTIC RESULTS* command.

A *RECEIVE DIAGNOSTIC* command (opcode 0x1c) is issued following the completion of the *SEND DIAGNOSTIC* command. Table 4–8 lists the format of the diagnostic data returned:

Table 4–8, Receive Diagnostic Command Data Return Format

Byte	Bit <7>	Bit <6>	Bit <5>	Bit <4>	Bit <3>	Bit <2>	Bit <1>	Bit <0>
0	Routine in error (00,50,51,52,53,54,55,57)							
1	Pass count							
2	FRU code							
3	Reserved byte							
4	(MSB)	First Symptom Code						
5								(LSB)
6	(MSB)	Second Symptom Code						
7								(LSB)
8	(MSB)	Third Symptom Code						
9								(LSB)

The following are field descriptions for the *RECEIVE DIAGNOSTIC RESULTS* command:

- **Routine in Error**

  This field contains the Routine ID of the failed routine. If this field contains 0x00, there were no errors detected during the last execution of a *SEND DIAGNOSTIC* command for the tape logical unit self-test.

  If this field contains 0xff, the diagnostic results of data are not generated as the results of a *SEND DIAGNOSTIC* command. This field is set to 0xff on completion of a successful *RECEIVE DIAGNOSTIC RESULTS* command for the tape controller (formatter) self-test.

- **Pass Count**

  This field contains the number of passes attempted before an error was detected. If an error is detected on the first pass, this field contains a 1. This field is reset each time a new routine is started. For example: if the *SEND DIAGNOSTIC* command parameter list contained a pass count of 7 for Routine 51 and an error was detected on the third attempt to execute Routine 51, this field would contain a 3.

- **FRU code**

  The FRU (Field Replaceable Unit) code attempts to identify the most likely failure at the board level. For example, if the FRU code is a 1, the most likely failure is the SCSI Interface (SI) board in the tape controller (formatter). The farthest left four bits contain the least probable FRU and the farthest right four bits contain the most likely FRU. Refer to byte 14 (FRU code) sense data error information in the "Tape System Status and Error Information" section of this chapter.

- **Symptom Codes**

  The symptom code attempts to identify the hardware condition that caused the failure. For example, if the first symptom code is 0x0009, the second is 0x918a, and the third is 0x0073, this would mean the machine reel tachometer fluctuated, the RECA dropped at DID write, and the ready signal went off. If the MSB of the symptom code is 00 (for example, 00xx), refer to the *Fujitsu Cartridge Tape Drive CE Manual*, Table 9.1, "Troubleshooting by check codes," for more information. Otherwise, refer to the *Fujitsu Cartridge Tape Controller Customer Engineering Manual*, Appendix A, "FSC DIRECTORY," for more information.

The following is an example of failure output from Tape Logical Unit Self-Test when the tape unit input power was turned off:

```
Failed:  Tape logical unit self test

Error: Formatter LUN self test detected error executing inline routine.
     : from controller_lun_selftest
Routine in error= 50, Pass count= 01, FRU CODE= 00
1st symptom code= 8520, 2nd symptom code= 8580, 3rd symptom code= 8520
```

- Symptom code 8520 = TSTBB on time-out at data transfer out sequence
- Symptom code 8580 = MTU offline

NOTE

Symptom code definitions can be found in the *Fujitsu Cartridge Tape Controller Customer Engineering Manual*, Appendix A, "FSC DIRECTORY," and in the *Fujitsu Cartridge Tape Drives CE Manual*, Chapter 9, "MAINTENANCE," Table 9.1, "Troubleshooting by check codes."

The following is an example of failure output from the Tape Logical Unit Self-Test with no tape cartridge installed:

```
          Subtest 201    0:00:43   failed

*****  Wed May  2 11:39:59 1990  *****
Test:    dev_vscsit.t  1.1  Class: 2     Subtest: 201 1.1   Count: 1    Error: 0
Failed:  Tape logical unit self test


Error: Formatter lun self test detected error executing inline routine.
     : from controller_lun_selftest
     Routine in error= 52, Pass count= 01, Fru Code= 00
     1st symptom code = 00a3, 2nd symptom code = 00a3, 3rd symptom code = 8510
```

- Symptom code 00a3 = Motion command received when *not ready* was set
- Symptom code 8510 = TSTI on time-out at ending sequence

The following is an example of a tape unit input power interrupt error:

NOTE

Refer to sections "SCSI Host Adapter Status and Error Information," and "Tape System Status and Error Information" for more information on this error.

```
         Subtest 201    0:00:45   failed

*****  Mon Apr 16 18:34:07 1990  *****
Test:    dev_vscsit.t  1.1       Class: 2    Subtest: 201 1.1    Count: 1 Error: 0
Failed:  Tape logical unit self test


Error: (flag byte) Command completed with error status.

Error: (error byte) Bad status from scsi device

Error: (SCSI stat byte) Check condition - error or exception event occurred.

Sense bytes:

[ 0-9 ] = 70 00 02 00 00 00 00 24 00 00
[10-19] = 00 00 04 00 00 00 00 00 00 3b

Sense information interpretation:

 70             Residual-length field is NOT valid
 00             Segment number = 0
 02             Sense Key = 2 (DRIVE NOT READY)
 00000000       Residual length = 0 (NOT valid)
 24             36 additional sense bytes
 00 00 00 00    (Command-specific bytes)
 04 00          Logical unit not ready, cause not reported
 00             No FRU identified
 00 00 00       (Sense-key-specific bytes - NOT valid)
 00             No additional info in sense bytes 20-43
 3b             ERPA: Volume removed by operator
```

From the example above, the following are request sense bytes of interest:

Request sense bytes[0-9] = 70 00 **02** 00 00 00 00 24 00 00

Request sense bytes[10-19] = 00 00 **04 00** 00 00 00 00 00 **3b**

- Sense key definitions are listed in "Table dev_vscsit-18, Sense Key Codes," on page 45 of this chapter.
- Request sense bytes 12 and 13—Additional sense code/qualifier (04 00 = Logical unit not ready, cause not reportable)

  Additional sense code and sense code qualifiers definitions are listed in "Table dev_vscsit-19, Additional Sense Codes and Descriptions," on pages 46 through 49 of this chapter.
- ERPA code definitions are listed on pages 50 through 54 of this chapter.

4.9 Class 3 Subtests, Tape Drive Tests

Class 3 subtests verify tape motion and that the drive can read and write data to the tape. Class 3 subtests use the command set from the standard Common Message Interface (CMI) set.

CAUTION

Any data existing on the scratch tape will be lost during the execution of these subtests.

The first four subtests do not perform any data verification checks. A status check is made prior to all commands to ensure that the tape is online and ready. Before any command using forward motion is issued, a check is made to ensure that the physical end of the tape has not been reached. For all write commands, a check is made to ensure that the write protect has not been set on the tape media. A Beginning-of-Tape (BOT) status bit set check is done for every *REWIND* command to check completion status. The *REWIND* command will not be sent to the tape driver if the tape media is already at BOT.

Each subtest has no dependency on prior execution of other subtests. The execution of the subtests in sequential order will check the tape subsystem in order of increasing complexity. Table 4–9 lists all Class 3 subtests, their descriptions, and the approximate time required to execute each one:

Table 4–9, Class 3 Subtests

Subtest	Description	3480 Time (min:sec)[1]	Dat Time (min:sec)[1]
300	Tape Mark Test	01:17	01:00
301	Space Record Test	02:06	02:00
302	Erase Test	00:48	n/a[2]
303	Long Block Read Test	00:15	01:00
304	Fixed Record Size Read/Write Test	04:40	06:00
305	Write File Unix Style Test	01:12	04:00
306	Variable Size Records Test	00:42	01:00
307	Long Space Test	00:59	02:43
308	Tape Release Test	n/a[3]	07:28

[1] The times presented are approximated.
[2] This test does not run on the DAT drive subsystem.
[3] This test does not run on the 3480 cartridge tape subsystem.

4.9.1 Subtest 300, Tape Mark Test

Subtest 300 verifies that a tape mark can be written to the tape and detected. The subtest first verifies that a tape mark that has been written can be detected in the forward direction. Next, 100 tape records, each followed by a tape mark, are written. *SPACE FILE* commands are then issued in both the forward and reverse directions. Tape position is then verified by testing for the presence or absence of a tape mark.

The following is an example of a failed Tape Mark Test:

```
Error: Expected tape mark status did not occur
     : while fwd space file
```

4.9.2 Subtest 301, Space Record Test

Subtest 301 verifies the *SPACE FORWARD RECORD* and the *SPACE REVERSE RECORD* commands. The subtest begins by rewinding the tape and writing two records followed by a tape mark. The subtest then writes 20 records of decreasing size to the tape, the biggest of these records being 1,000 bytes and the smallest being 50 bytes. A tape mark is then written to the tape. A space record test is then performed between the end points denoted by the tape marks. The subtest then issues a varying series of *FORWARD SPACE RECORD* commands. The subtest verifies the success of the space record test by checking for the presence of a filemark. The testing sequence is repeated using the *BACK SPACE RECORD* command. The subtest then repeats the testing sequence with a combination of *FORWARD SPACE* and *BACK SPACE* commands.

The following is an example of a failed Space Record Test:

```
Error: Unexpected tape mark status sensed
     : while fwd space rec
```

4.9.3 Subtest 302, Erase Test

NOTE

This test does not run on the DAT drive subsystem.

Subtest 302 verifies that an *ERASE* command used in a write recovery, due to write errors, does not leave glitches on the tape. The subtest writes 10 records of 4,096 bytes each to the tape. Each record, in turn, is erased and the remaining records are reconstituted to make a total of 10 records in the file again. The integrity of the file is checked by spacing over the records.

The following is an example of a failed Erase Test:

```
Error: Expected tape mark status did not occur
     : while backspace rec
```

4.9.4 Subtest 303, Long Block Read Test

Subtest 303 verifies that the long-block status bit indicates correctly whether a long block is present. This test begins by writing a series of records correctly on the tape, starting with a record of four bytes and each subsequent record being twice the size of the previous record. The sixteenth and last record written to the tape is 128 Kbytes long. Next, Four read passes are used to read the records, test the long-block status bit, and verify that the number of bytes transferred was correct.

The first read pass has the number of bytes to be transferred equal to the record size on the tape. A check is then made to ensure that the long-block status bit is not set.

The second pass reads each record with the requested transfer size as two bytes less than the record size on the tape. A check is made to ensure that the long-block status bit is set and the number of bytes transferred is equal to the number requested.

The third pass is the same as the second pass except that the requested transfer size is one byte less than the record size on the tape. The results of this pass are odd byte counts in the data transfer size requests.

The fourth and final pass reads each record except the 128-Kbyte record, with the requested transfer size equal to the maximum number of bytes allowed by the 3480/DAT drive (0x20000). A check is made to ensure that the long-block status bit is not set and that the number of bytes transferred is not equal to the number of bytes requested. The following is an example of a failed Long Block Read Test:

```
Error: Expected long block read status did not occur
     :from long_xblk_test
```

4.9.5 Subtest 304, Fixed Record Size Read/Write Test

Subtest 304 writes and reads 160 records of a fixed size. The first write pass writes twenty 16-Kbyte records to the tape. Each of these records contains a different data pattern. On subsequent write passes, the record size is increased by 16 Kbytes until the record size reaches the maximum 128 Kbytes. The tape is then rewound and the records are read and the data is verified. Table 4–10 lists the data pattern used:

Table 4–10, Subtest 304 Data Pattern

Hexadecimal Test Pattern			
00000000	ffffffff	a5a5a5a5	5a5a5a5a
f0f0f0f0	0f0f0f0f	cc33c3c3	99669696
01010101	02020202	04040400	8080808
10101010	20202020	40404040	80808080
fefdfbf7	efdfbf7f	0fa5c396	12487edb

The following is an example of a failed Fixed Record Size Read/Write Test:

```
Error: data miscompare
     : buf adr 00000000 exp 00000000 act 12487edb
     : while reading from tape
```

4.9.6 Subtest 305, Write File UNIX Style Test

Subtest 305 simulates the sequence of commands that are given to the tape driver by the ConvexOS driver. A file of 8 records of 32 Kbytes each are written to tape. Two filemarks are written to the tape, and a *BACKSPACE FILEMARK* command is issued. Seven more files are written in the same manner. The data in each record is set equal to the record number. The end result is 64 numbered records on the tape with a filemark after every 8 records. The last record ends with two filemarks. The tape is rewound. All records are read and verified and all filemarks are verified to be in the right locations on the tape.

The following is an example of a failed Write File UNIX Style Test:

```
Error: Expected tape mark status did not occur
     : while verifying last tapemark on tape
```

4.9.7 Subtest 306, Variable Size Records Test

Subtest 306 verifies that variable-length records on a tape can be written, read, and verified. First, records of 1 byte to 258 bytes (each record size is incremented by 1 byte) are written to the tape. Next, sets of 5 records each are written to the tape. The sizes of these records are designed to cross base 2 boundaries. With the exception of the 1-byte record, the first 2 bytes of each record contain the record number. Byte 3 and greater all contain a data pattern that is either defaulted or is user-specified from the keyboard or from a file. The default pattern is 0x6db6. The default number of error printouts is 8 but can be set up to 100 error printouts by the user. The total number of miscompares in a record are totaled for the printout. There are a total of 298 records written and verified with the size of the records ranging from 1 byte to 65,538 bytes in this subtest. Table 4–11 lists the record sizes for Subtest 306:

Table 4–11, Subtest 306 Record Sizes

Starting Size	Number of Records	Record Sizes	Record Number
1	258	1 to 258	1 to 258
510	5	510 to 514	259 to 263
1,022	5	1,022 to 1,026	264 to 268
2,046	5	2,046 to 2,050	269 to 273
4,094	5	4,094 to 4,098	274 to 278
8,190	5	8,190 to 8,194	279 to 283
16,382	5	16,382 to 16,386	284 to 288
32,766	5	32,766 to 32,770	289 to 293
65,534	5	65,534 to 65,538	294 to 298

The following is an example of a failed Variable Records test:

```
Error in record 292, record length=32769 bytes
Error: data byte miscompared
     : buf adr = 00000008  exp = 6d  act = dd
     : buf adr = 00000009  exp = b6  act = bb
     : buf adr = 00000f01  exp = b6  act = bb
     : buf adr = 00000f02  exp = 6d  act = 6f
     : buf adr = 00002012  exp = 6d  act = 66
     : buf adr = 00002013  exp = b6  act = b7
     : buf adr = 00002014  exp = 6d  act = 7d
     : buf adr = 00004022  exp = 6d  act = 6f
     : Total miscompares = 10
     : while comparing data read from tape
  0:00:30     failed
*****  Sat May 12 15:23:30 1990  *****
Test:    dev_vscsit.t  1.1  Class: 3     Subtest: 306 1.1  Count: 1   Error: 0
Failed:  Variable size records read/write test
```

4.9.8 Subtest 307, Long Space Test

Subtest 307 verifies that the tape drive can space across long regions of the tape. The subtest begins by rewinding the tape and writing two records followed by a tape mark. The subtest then writes 3 records of 100 bytes, 80 records of 128 Kbytes, 3 records of 100 bytes, and a tape mark. *SPACE FORWARD RECORD* and *SPACE REVERSE RECORD* commands are then used to move back and forth across these records.

4.9.9 Subtest 308, Tape Release Test

> **NOTE**
>
> This test does not run on the 3480-compatible cartridge tape subsystem.

Subtest 308 verifies the ability of the DAT drive to automatically release the capstan, pinch roller, and cylinder without losing its position on the tape or corrupting data stored on the tape. The DAT drive will release the capstan and pinch roller when it is idle for more than 30 seconds, and will pull the tape medium away from the cylinder when it is idle for more than 90 seconds. This subtest verifies these functions by first rewinding the tape, writing 10 records of 64 Kbytes each, and waiting for 45 seconds for the drive to release the capstan and pinch roller. It then writes 10 more records and waits 120 seconds for the drive to pull the tape away from the cylinder. The subtest then writes 10 more records, rewinds the tape, verifies that the first 5 records are correct, and with the drive in "read" mode waits 120 seconds for the tape to be pulled away from the cylinder again. The subtest then verifies the next 20 records and waits 45 seconds for the capstan and pinch roller to be released again. Finally, the last 5 records are verified, the tape is rewound, and all 30 records are verified without interruption.

4.10 Class 4 Subtests, Tape Driver Exception Tests

Class 4 subtests test various conditions that do not usually occur during normal tape drive operations. These exception subtests test the tape driver and controller stability and integrity when such operations such as *ZERO BYTES READ* or *WRITE* commands are submitted to the tape driver.

> **CAUTION**
>
> Any data existing on the scratch tape will be lost during the execution of these subtests.

Table 4–12 lists all Class 4 subtests, their descriptions, and the approximate time required to execute each subtest:

Table 4–12, Class 4 Subtests

Subtest	Description	3480 Time (min:sec)[1]	DAT Time (min:sec)[1]
400	Beginning-of-Tape (BOT) Exception Test	00:07	00:30
401	Zero Length Operations Exception Test	00:09	00:30
402	End-of-Data (EOD) Test	00:17	01:30
403	End-of-Tape (EOT) Exception Test	04:56	02:00:00[2]

[1] The times presented are approximated.

[2] Because of the time required to complete, this test does not run as default option from the Test Parameter Menu.

4.10.1 Subtest 400, Beginning-of-Tape (BOT) Status Exception Test

This subtest rewinds the tape then writes a tape filemark. Three consecutive *BACKSPACE FILE* commands are issued. After the first *BACKSPACE FILE*, the BOT status bit should not be set, but the tape mark status bit should be set. At the completion of the second and third *BACKSPACE FILE*, the BOT status bit should be set and a check condition should occur. The request sense bytes obtained after the check condition are checked to see whether the EOM bit is set. A *FORWARD SPACE* command is then issued. The BOT status bit should be reset and the tape mark status bit should be set.

The tape is rewound and a 256-byte record is written. Three *Backspace Record* commands are issued consecutively. After the first *BACKSPACE RECORD* command, the BOT status bit should not be set. After the second and third *BACKSPACE RECORD* commands are complete, the BOT status bit should be set and a check condition should occur. The request sense bytes obtained after the check condition are checked to see whether the EOM bit is set. A *FORWARD SPACE RECORD* command is then issued. The BOT status bit should be reset.

4.10.2 Subtest 401, Zero-Length Operations Exception Test

Subtest 401 rewinds the tape and writes two 4-Kbyte records with known patterns. The zero-length operations are commands formatted to write 0 bytes, read 0 bytes, forward space 0 records, forward space 0 files, backspace 0 records, and backspace 0 files. Each of these zero-length operations is preceded by a backspace record of count = 1. The second record is then read and verified. No tape motion is expected for any of the zero-length commands sent to the tape driver.

4.10.3 Subtest 402, End-of-Data (EOD) Status Exception Test

Subtest 402 rewinds the tape and writes two records of known patterns of 128 Kbytes. A *BACKSPACE RECORD* command is issued followed by a command to write a 4-Kbyte record of a known pattern. This effectively replaces the second record on tape with a shorter record. The tape is rewound and the two records are read and verified. A third *READ* command is issued to attempt to read past the second record on tape. A check condition from the controller is expected with the request sense bytes indicating End-of-Data (EOD).

The tape is rewound and the first record is read and verified. An *ERASE* command with a default byte length is issued. The tape is rewound again. The first record is read and verified, and an attempt is made to read the erased second record. A check condition from the controller is expected with the request sense bytes indicating EOD.

4.10.4 Subtest 403, End-of-Tape (EOT) Status Exception Test

Subtest 403 rewinds the tape and writes multiple 128 Kbyte records until EOT is sensed in the status byte. A *BACKSPACE RECORD* command is issued. Then a 128 Kbyte record with a known pattern is written. An EOT status condition is expected as a result of the *WRITE* command. The tape is backspaced one record and the record just written is verified. A *REWIND* command is then issued, which completes the test.

4.11 Class 5 Subtests, Automatic Cartridge Loader (ACL) Tests

Class 5 subtests test the functionality of the magazine Automatic Cartridge Loader (ACL) mechanism.

> **CAUTION**
>
> Any data existing on the scratch tape will be lost during the execution of these subtests.

Table 4–13 lists all Class 5 subtests, their descriptions, and the approximate time required to execute each subtest when using two tapes in the magazine:

Table 4–13, Class 5 Subtests

Subtest	Description	3480 Time (min:sec)[1]	DAT Time (min:sec)[2]
500	Stacker Auto Test:Stacker automatic mode test	2:58	N/A
501	Stack System Test:Stacker system mode test	2:05	N/A

[1] The times presented are approximated.
[2] Class 5 subtests are not supported for DAT.

4.11.1 Subtest 500, Stacker Automatic Mode Test

Subtest 500 tests the stacker in automatic mode, which loads tape cartridges sequentially. The operator is required to load a magazine containing at least two test tapes. The diagnostic makes two passes through the magazine. A unique pattern is written on each cartridge on the first pass. These patterns are verified on the second pass, to assure that the stacker is progressing through the tape cartridges in proper sequence.

4.11.2 Subtest 501, Stacker System Test

Subtest 501 tests the stacker in system mode, which permits random access loading of tape cartridges. The operator is required to load a magazine containing at least two test tapes. The diagnostic makes two passes through the magazine. On the first pass, the cartridges in the magazine are loaded at random and a unique pattern is written on each cartridge. On the second pass, the cartridges are loaded in the same order and the patterns are verified.

4.12 Interactive Debugger

The diagnostic provides an interactive debugger that supports the ability to execute commands from a script file. This allows more flexibility in debugging. Invoke the debugger with one of the following methods:

- Use the **-d** option when invoking the diagnostic (enter **dev_vscsit**[x] **-d**). No subtests are executed.
- Respond with a `y` to the `Enable Debug Monitor` question in the `TEST PARAMETER SUMMARY`. The debugger will then be invoked if an error is encountered during a test.

Once the interactive debugger is entered, online help commands are available. By entering **help**, the following screen is displayed:

Figure 4–8, Interactive Debugger On-line Help

```
Input base specification:
    0dNN - decimal,  0xNN or NN - hexadecimal,  the default is hexadecimal

Meta-command sequences:
   ![UNIX_CMD]       - execute UNIX_CMD
   !![UNIX_CMD]      - fork a shell and execute UNIX_CMD (allows redirection)
   <FILE             - redirect input from FILE (recursive)
   <<FILE            - end input from current file and change input to FILE

Commands:
    Commands may be abbreviated as long as the abbreviation is unique.

    help    [COMMAND ...]                    - display general or specific help
    cd      [DIRECTORY]                      - change to DIRECTORY
    quit                                     - exit debug mode
    echo    [-n] [arg ...]                   - echo statement to display
    pause   [-n] [seconds]                   - pause for <C/R> or seconds
    mb      begin [end] [step]               - modify/[dump] bytes CCU
    mw      begin [end] [step]               - modify/[dump] words CCU
    ml      begin [end] [step]               - modify/[dump] longs CCU
    mmb     begin [end] [step]               - modify/[dump] bytes in MM
    mmw     begin [end] [step]               - modify/[dump] words in MM
    mml     begin [end] [step]               - modify/[dump] longs in MM
    fb      begin [end] value [incr [step]] - fill bytes CCU
    fw      begin [end] value [incr [step]] - fill words CCU
    fl      begin [end] value [incr [step]] - fill longs CCU
    ffb     begin [end] value [incr [step]] - fill bytes in Main Memory
    ffw     begin [end] value [incr [step]] - fill words in Main Memory
    ffl     begin [end] value [incr [step]] - fill longs in Main Memory
    weof    count                            - write EOF count times
    fsr     count                            - forward space record count times
    bsr     count                            - backward space file count times
    fsf     count                            - forward space file count times
    bsf     count                            - backward space file count times
    erase                                    - erase tape, default length
    wphys   byte_count [pattern]             - write bytes with optional pat
    rphys   byte_count [pattern to verify]   - read bytes with opt pat to verify
    rewind                                   - rewind tape unit
    changeunit  unit subunit                 - change current unit and subunit
    tracemsgs   0 or 1                       - trace mbs messages control
    notimeout   0 or 1                       - mbs message timeout control
    status                                   - read tape status(IO_RDSTATS_PHYS)
    laststat                                 - read last command's sense bytes
    unload                                   - unload tape (IO_UNLOAD)
    unitclr                                  - clear tape error status
    boot                                     - reboot the CCU driver
    reset                                    - reset the host adapter
    dapselftest                              - diagcmd adapter prom self-test
    darselftest                              - diagcmd adapter ram self-test
    daidentify                               - diagcmd adapter identify(rev num)
    dfcselftest                              - diagcmd target cntrl self-test
    dfuselftest                              - diagcmd target lun self-test
    dfrecvdiag                               - diagcmd recv diag results
    dfdispload string                        - diagcmd load display on tape unit
    stkselect                                - select a tape in the stacker
    stkstat                                  - read stacker status
    stkmode                                  - set stacker mode
    stkload                                  - load tape from stacker
    stkunload                                - unload tape to stacker
    stkloadmag                               - load stacker magazine
```

In addition to the help screen in the previous figure, help for each specific command can be obtained by entering:

help *command*

Where *command* is the desired debugger command. Abbreviations of the desired commands may be used. For example, for help with all commands that start with the letter "r," enter:

help r

4.12.1 Interactive Debugger Command Descriptions

4.12.1.1 help

Usage: **help** [COMMAND ...]

Displays general or specific help, where COMMAND is replaced with the desired interactive debugger command. Specific help is displayed for COMMAND. The COMMAND may be an abbreviation. The following example would list help for all commands that start with the letter "r":

help r

4.12.1.2 cd

Usage: **cd** [DIRECTORY]

Changes current directory to desired directory, where DIRECTORY can be any valid directory path. If DIRECTORY is omitted, the default path is $HOME or "/" if $HOME not set.

4.12.1.3 quit

Usage: **quit**

Exits the interactive debugger.

4.12.1.4 echo

Usage: **echo** [-n] [arg ...]

Writes arguments to the display separated by blanks and is terminated by a new line, where:

-n means do not terminate with a new line

arg... is the list of arguments to send to the display

4.12.1.5 pause

Usage: **pause** [-n] [seconds]

Waits for specified amount of time or for a RETURN if the time is omitted, where:

-n means do not echo the pause message

seconds specifies the number of seconds to pause

4.12.1.6 mb, mw, ml

Usage: **mb begin** [end] [step]
mw begin [end] [step]
ml begin [end] [step]

Displays or modifies, or both, CCU address space in byte-at-a-time mode (mb), word-at-a-time mode (mw), or long-word-at-a-time mode (ml), where:

- **begin** is the initial address
- **end** is the ending address
- **step** is the address increment (default is access size)

If the ending address is omitted, this command enters an interactive mode that allows modification of memory. The following list gives the valid responses while in interactive mode:

[<value>]	write optional <value> to current address, advance to next address
[<value>]=	write optional <value> to current address, stay at the present address (re-read)
[<value>]^[N]	write optional <value> to current address, move to address N (address 0 if N is omitted)
[<value>]+[N]	write optional <value> to current address, advance to the next address (N addresses if N is specified)
[<value>]-[N]	write optional <value> to current address, back up to the previous address (N addresses if N is specified)
[<value>]q	write optional <value> to current address, exit interactive mode

Multiple commands may be specified on the same line. A comma or space may be used to separate the commands or value, as shown in the following example:

```
Debug mode -> mb c03fc1
<c03fc1> = 1c 00=ff,1q
```

where **1c 00=ff,1q** is an example of executing multiple commands on the same line. This sequence modifies the byte at address 0xc03fc1 to 0, re-reads and displays the new value, modifies the byte to 0xff, skips to address 0xc03fc2 and modifies it to a 0x1, and then quits interactive mode.

4.12.1.7 mmb, mmw, mml

Usage: **mmb begin** [end] [step]
mmw begin [end] [step]
mml begin [end] [step]

Displays or modifies, or both, main memory address space in byte-at-a-time mode (mmb), word-at-a-time mode (mmw), or long-word-at-a-time mode (mml), where:

- **begin** is the initial address
- **end** is the ending address
- **step** is the address increment (default is access size)

If the ending address is omitted, this command enters an interactive mode that allows modification of memory. The following list gives all valid responses while in interactive mode:

[<value>]	write optional <value> to current address, advance to next address
[<value>]=	write optional <value> to current address, stay at the present address (re-read)
[<value>]^[N]	write optional <value> to current address, move to address N (address 0 if N is omitted)
[<value>]+[N]	write optional <value> to current address, advance to the next address (N addresses if N is specified)
[<value>]-[N]	write optional <value> to current address, back up to the previous address (N addresses if N is specified)
[<value>]q	write optional <value> to current address, exit interactive mode

Multiple commands may be specified on the same line. A comma or space may be used to separate the commands or values, as shown in the following example:

```
Debug mode -> mmb c03fc1
<c03fc1> = 1c 00=ff,1q
```

where **1c 00=ff,1q** is an example of executing multiple commands on the same line. This sequence modifies the byte at main memory address 0xc03fc1 to 0, re-reads and displays the new value, modifies the byte to 0xff, skips to address 0xc03fc2 and modifies it to a 0x1, and then quits interactive mode.

4.12.1.8 fb, fw, fl

Usage: **fb begin** [end] **value** [incr [step]]
fw begin [end] **value** [incr [step]]
fl begin [end] **value** [incr [step]]

Fills CCU memory with a specified pattern in byte-at-a-time mode (fb), word-at-a-time mode (fw), or longword-at-a-time mode (fl), where:

- **begin** is the starting address
- **end** is the ending address
- **value** is the initial fill value
- **incr** is the fill value increment
- **step** is the address increment

4.12.1.9 ffb, ffw, ffl

Usage: **ffb begin** [end] **value** [incr [step]]
ffw begin [end] **value** [incr [step]]
ffl begin [end] **value** [incr [step]]

Fills CCU memory with a specified pattern in byte-at-a-time mode (ffb), word-at-a-time mode (ffw), or longword-at-a-time mode (ffl), where:

- **begin** is the starting address
- **end** is the ending address
- **value** is the initial fill value
- **incr** is the fill value increment
- **step** is the address increment

4.12.1.10 weof

Usage: **weof count**

Writes a number of End-of-File (EOF) marks on the tape medium, where:

count is the number of EOF marks to be written

4.12.1.11 fsr, bsr, fsf, bsf

Usage: **fsr count**
bsr count
fsf count
bsf count

Spaces forward by *count* records (fsr), backward by *count* records (bsr), forward by *count* files (fsf), and backward by *count* files (bsf).

4.12.1.12 erase

Usage: **erase**

Erases a default length of tape on the selected tape drive. For DAT drives, this command erases the rest of the tape by placing an EOD mark at the current tape position.

4.12.1.13 wphys

Usage: **wphys byte_count** [32-bit pattern]

Writes *byte_count* bytes to the physical device with optional *pattern* of bytes.

4.12.1.14 rphys

Usage: **rphys byte_count** [pattern to verify]

Reads from the physical device *byte_count* number of bytes with optional *pattern* to verify.

4.12.1.15 rewind

Usage: **rewind**

Rewinds tape unit.

4.12.1.16 changeunit

Usage: **changeunit unit subunit**

Selects the next tape unit to be tested in the debug mode. Where **unit** is the SCSI ID of the 3480-compatible formatter or DAT drive and **subunit** is the subunit number of the tape drive.

4.12.1.17 notimeout

Usage: **notimeout {0 | 1}**

Enables (**0**) or disables (**1**) the timeout of the MBS return message. It is useful during debugging of the driver when timeout should be disabled.

4.12.1.18 tracemsgs

Usage: **tracemasgs {0 | 1}**

Enables (**1**) or disables (**0**) the listing of the mbs send and receive messages sent to and received from the driver. When an error is encountered in the test, the debugger mode may be entered and the message tracing should be turned on.

4.12.1.19 status

Usage: **status**

Reads and returns current tape unit status.

4.12.1.20 laststatus

Usage: **laststatus**

Retrieves the sense bytes from the last request sense command performed by the CCU driver.

4.12.1.21 unload

Usage: **unload**

Unloads tape medium from tape unit.

4.12.1.22 unitclr

Usage: **unitclr**

Clears tape error status.

4.12.1.23 boot

Usage: **boot**

Reboots the CCU driver.

4.12.1.24 reset

Usage: **reset**

Resets the host adapter using EGOS to write to the host adapter reset port. An EGOS read from the status port follows the write to read status from the host adapter. A bad status signal will result in an error printout. After this *RESET* command, the *BOOT* command in debugger mode should be invoked to re-configure and re-probe the driver.

4.12.1.25 dapselftest

Usage: **dapselftest**

Tells the host adapter to run its PROM self-test.

4.12.1.26 darselftest

Usage: **darselftest**

Tells the host adapter to run its RAM self-test.

4.12.1.27 daidentify

Usage: **daidentify**

Tells the host adapter to report its revision number.

4.12.1.28 dfcselftest

Usage: **dfcselftest**

Tells the 3480-compatible tape controller (formatter) or DAT drive to run its self-test.

4.12.1.29 dfuselftest

> **NOTE**
>
> **dfuselftest** is not supported on the DAT drive subsystem.

Usage: **dfuselftest**

Tells the target 3480-compatible tape unit to run its self-test.

4.12.1.30 dfrecvdiag

> **NOTE**
>
> **dfrecvdiag** is not supported on the DAT drive subsystem.

Usage: **dfrecvdiag**

Tells the target 3480-compatible device to report its diagnostics results.

4.12.1.31 dfdispload

> **NOTE**
>
> **dfdispload** is not supported on the DAT drive subsystem.

Usage: **dfdispload** *string*

Tells the 3480-compatible target device to load a *string* on its display.

4.12.1.32 stkselect

Usage: **select** *n*

Selects tape #n in the stacker.

4.12.1.33 stkstat

Usage: **stkstat**

Obtains the status from the stacker. The status is returned in the CMI message. It must be viewed there by using *trace_msgs*.

4.12.1.34 stkloadmag

Usage: **stkloadmag**

Loads the magazine into the stacker

4.12.1.35 stkload

Usage: **stkload**

Causes the stacker to load the cartridge into the tape drive.

4.12.1.36 stkunload

Usage: **stkunload**

Causes the tape drive to unload the cartridge into the stacker.

4.12.1.37 stkmode

Usage: **stkmode** *[4|8]*

Sets the stacker mode to auto (4) or system (8).

4.12.2 Using a Test Script

The following procedure describes how to run a test script and includes an example test script:

1. Create two files. For this example, the script files are named *scr3* and *scr3a*. Figures 4–9 and 4-10 show examples of *scr3* and *scr3a* script files:

Figure 4–9, Example Script File, *scr3*

```
echo ++
echo ************Tape test starting************
echo status command
status
echo ...rewinding tape
rew
echo + write 1 filemark at bot
weof 1
<<scr3a
```

Figure 4–10, Example Script File, *scr3a*

```
echo ++
echo ************Scr3a tape test script re-starting************
echo + backspace 1 file
bsf 1
echo + erase the old tape mark
erase
echo + weof 1 (write new tape mark)
weof 1
echo + write 10000 bytes with 11111111
wphys 10000 11111111
echo + write 16000 bytes with 22222222
wphys 16000 22222222
echo + write 1f000 bytes with 33333333
wphys 1f000 33333333
echo + write 20000 bytes with 44444444
wphys 20000 44444444
echo + back space 1 record
bsr 1
echo + back space 3 records
bsr 3
echo + back space 1 file
bsf 1
echo + forward space 1 file
fsf 1
echo + read 10000 11111111
rphys 10000 11111111
echo + read 16000 22222222
rphys 16000 22222222
echo + read 1f000 33333333
rphys 1f000 33333333
echo + read 20000 44444444
rphys 20000 44444444
<<scr3a
```

2. Invoke the *dev_vscsit* diagnostic using the debug option (**dev_vscsit.t -d**).
3. Respond to the questions asked in the diagnostic normally.
4. At the `Debug Mode->` prompt, type *<scriptname*. Figure 4–11 shows the output of the example test script:

Figure 4–11, Example Test Script Output

```
Debug Mode-> <scr3
++
************Tape test starting************
status command

The current tape unit status is = 0x00000043
+ write 1 filemark at bot
++
************Scr3a tape test script re-starting************
+ backspace 1 file
+ erase the old tape mark
+ weof 1 (write new tape mark)
+ write 10000 bytes with 11111111
+ write 16000 bytes with 22222222
+ write 1f000 bytes with 33333333
+ write 20000 bytes with 44444444
+ back space 1 record
+ back space 3 records
+ back space 1 file
+ forward space 1 file
+ read 10000 11111111
+ read 16000 22222222
+ read 1f000 33333333
+ read 20000 44444444
++
************Scr3a tape test script re-starting************
+ backspace 1 file
+ erase the old tape mark
+ weof 1 (write new tape mark)
+ write 10000 bytes with 11111111
+ write 16000 bytes with 22222222
+ write 1f000 bytes with 33333333
+ write 20000 bytes with 44444444
+ back space 1 record
+ back space 3 records
+ back space 1 file
+ forward space 1 file
+ read 10000 11111111
+ read 16000 22222222
+ read 1f000 33333333
+ read 20000 44444444
++
```

5. The script will continue to run until the EOT is reached. To exit the program before the EOT is reached, press CTRL **c**.

4.13 SCSI Host Adapter Status and Error Information

The following sections include information on status and error reporting for the SCSI host adapter. Table 4–14 lists the contents of the SCSI host adapter status block:

Table 4–14, Status Block Contents

Bits <31..24>	Bits <23..16>	Bits <15..8>	Bits <7..0>
Command Identifier			
Reserved	SCSI Status	Error	Flags
Class/Code	Segment	SCSI Flags	Info Byte 3
Info Byte 4	Info Byte 5	Info Byte 6	EX Length

4.13.1 Flags Byte

This byte contains flags indicating the status of the host adapter. Table 4–15 lists each flag bit in the host adapter status block flags byte:

Table 4–15, Status Block Flags Byte

Bit <7>	Bit <6>	Bit <5>	Bit <4>	Bit <3>	Bit <2>	Bit <1>	Bit <0>
CC	ERR	RTY	DTT	TAR	CSB	SE	0

The following list contains a brief explanation of each flag bit:

CC Command complete

ERR Error status (this command had an error)

RTY Command required one or more retries

DTT Data transfer truncated (SCSI command completed, but fewer bytes were transferred than requested)

TAR Target mode enabled in SCSI host adapter

CSB Continued status block (for additional sense data)

SE Soft error

4.13.2 Error Byte

The following list contains a brief explanation of each error byte code:

0x01	Invalid Board Command
0x02	Bad Unit or ID Number
0x03	Floppy Disk Option Not Installed
0x0b	Reserved Field Not Zero
0x0e	Command List Stopped
0x0f	Bad Command List Size Field
0x11	List Already Active
0x14	Bus Time-out
0x15	Bus Error
0x16	Scatter/Gather Descriptor Block Read Error
0x1e	SCSI Select Time-out
0x1f	SCSI Disconnect Time-out
0x20	SCSI Parity Error
0x21	Unexpected SCSI Disconnect
0x22	General SCSI Bus Error
0x23	SCSI Device Returned Bad Status
0x24	Unexpected SCSI Phase Encountered
0x25	Bad Byte Seen by SCSI Controller Chip
0x26	Error in Synchronous Transfer Negotiation
0x27	SCSI Bus Reset During Operation
0x28	Target Command not Found
0x29	This command must be issued with a command list
0x2a	Drive is Write Protected
0x2b	Vendor Unique Command Set Up Improperly, Modifier Field Zero
0x2c	Bad SCSI Chip Condition
0x61	Static RAM Error
0x62	PROM Checksum Error
0x63	Undefined Diagnostic Specified
0x80	Firmware Error (0x80 and above)

4.13.3 Host Adapter Status Byte

A status byte is sent from the target to the initiator during the STATUS phase at the termination of each command unless the command is cleared by:

- An ABORT message
- A BUS DEVICE RESET message
- A "hard" RESET condition
- An unexpected BUS FREE condition

Table 4–16 lists the bits in a host adapter status byte:

Table 4–16, Host Adapter Status Byte

Byte	Bit <7>	Bit <6>	Bit <5>	Bit <4>	Bit <3>	Bit <2>	Bit <1>	Bit <0>
0	Rsvd		Status byte code					Rsvd

Table 4–17 lists the status byte codes and their descriptions:

Table 4–17, Bit Values for SCSI Status Byte Code

Status Byte Bits								Status Represented
<7>	<6>	<5>	<4>	<3>	<2>	<1>	<0>	
R	R	0	0	0	0	0	R	Good
R	R	0	0	0	0	1	R	Check condition
R	R	0	0	0	1	0	R	Condition met/good
R	R	0	0	1	0	0	R	Busy
R	R	0	1	0	0	0	R	Intermediate/condition met/good
R	R	0	1	1	0	0	R	Reservation conflict
R	R	1	0	1	0	0	R	Queue full

R = Reserved bit
All other codes are reserved

4.14 Tape System Status and Error Information

The following sections include information on status and error reporting for the 3480-compatible formatter and tape drive(s) in a cartridge tape system or the DAT drive subsystem.

4.14.1 Sense Data

The sense bytes contained in the 3480-compatible controller (formatter) or DAT drive indicate error, status, and statistical information about the controller to the drive or about the device. Error information is set in a sense byte when the check condition status is reported as a completion status. The sense byte is transmitted to an initiator by the *REQUEST SENSE* or the *REQUEST LOG* command. Table 4–18 lists the sense byte format for error code 70:

Table 4–18, Sense Bytes Format

Byte	Bit <7>	Bit <6>	Bit <5>	Bit <4>	Bit <3>	Bit <2>	Bit <1>	Bit <0>
0	Valid	1	1	1	0	0	0	0
1	Segment Number (0x00)							
2	FMark	EOM	ILI	Rsvd	Sense Key			
3 to 6	(MSB) Residual Length (LSB)							
7	Additional Sense Length							
8 to 11	(MSB) (SCSI-2 Command Specific Information Bytes) (LSB)							
12	Additional Sense Code							
13	Additional Sense Code Qualifier							
14	FRU Code (*3480 Only*)							
15 to 17	SKSV (MSB) (SCSI-2 Sense-Key Specific Information Bytes) (LSB)							
18	Format of additional sense (*3480 Only*)							
19	Host ERPA (*3480 Only*)							
20 to 43	(MSB) Additional Sense Bytes as defined by the format indicated in byte 18. (*3480 Only*) (LSB)							

4.14.2 Sense Bytes Description

This section describes all bits in the sense format bytes.

Byte 0 (valid)

When the valid bit is a one, sense bytes 3 to 6 indicate the difference between the number of bytes, blocks, or filemarks requested by a command and the number of bytes, blocks, or filemarks actually executed.

Byte 1 (segment number)

Byte 2

Bit <7> fmark (filemark)

Bit <6> EOM (End-of-Medium)

Bit <5> ILI (Incorrect Length Indicator)

Bit <4> Ignored

Bits <3..0> Sense key, listed in Table 4–19

Table 4–19, Sense Key Codes

Code	Description
0x0	No Sense
0x1	Recovered Error
0x2	Not Ready
0x3	Medium Error
0x4	Hardware Error
0x5	Illegal Request
0x6	Unit Attention
0x7	Data Project
0x8	Blank Check
0x9	Busy/Not Busy
0xa	Copy Aborted
0xb	Aborted Command
0xd	Volume Overflow
0xe	Miscompare

Bytes 3 – 6 (residual length)

Byte 7 (additional sense length)

Bytes 8 – 11 (command specific information)

Bytes 12 – 13 (Additional sense code and sense code qualifiers as listed in Table 4–20)

Table 4–20, Additional Sense Codes and Descriptions

Byte 12	Byte 13	Description
00	00	No additional sense information
00	01	Filemark detected
00	02	End-of-Medium (EOM) detected
00	03	Beginning-of-Data (BOD) detected (3480 only)
00	03	Setmark detected (DAT only)
00	04	Beginning-of-Tape (BOT) detected
00	05	End-of-Medium (EOM) detected
03	00	Peripheral device write fault
03	01	No write current (*3480 only*)
03	02	Excessive write errors
04	00	Logical unit not ready, cause not reported
04	01	Logical unit not ready, manual intervention required (*3480 only*) Logical unit is in process of becoming ready (*DAT only*)
04	02	Logical unit not ready, initializing command required
04	03	Logical unit is in process of becoming ready (*3480 only*) Logical unit not ready, manual intervention required (*DAT only*)
04	04	Logical unit not ready, format in progress
05	00	Logical unit does not respond to selection
07	00	Multiple peripheral devices selected
08	00	Logical unit communication failure
08	01	Logical unit communication time-out
08	02	Logical unit communication parity error
09	00	Track following error
0a	00	Error log overflow
0c	00	Write error Sense Key says whether recovered
11	00	Unrecovered read error
11	01	Read retries exhausted
11	02	Error too long to correct
11	03	Multiple read errors
11	04	Physical EOT encountered (*3480 only*)
11	08	Incomplete block read (postamble not found)
11	09	No gap found
11	0a	Miscorrected error
14	00	Recorded entity not found
14	01	Record not found
14	02	Filemark not found
14	03	End-of-Data not found
14	04	Block sequence error

Table 4–20, Additional Sense Codes and Descriptions (continued)

Byte 12	Byte 13	Description
15	00	Random positioning error
15	01	Mechanical positioning error
15	02	Positioning error detected by read of medium
17	00	Recovered read data with no error correction applied
17	01	Recovered read data with retries
17	02	Recovered read data with positive head offset (*3480 only*)
17	03	Recovered read data with negative head offset (*3480 only*)
18	00	Recovered read data with error correction applied
1a	00	Parameter list length error
1b	00	Synchronous data transfer error
20	00	Invalid command operation code
21	00	Logical block address out of range
24	00	Invalid field in CDB (check field pointer)
25	00	Unsupported logical unit
26	00	Invalid field in parameter list (check field pointer)
26	01	Parameter not supported
26	02	Parameter value not supported
26	03	Threshold parameters not supported
27	00	Write protected
28	00	Not ready to ready transition (medium may have changed)
29	00	Power on, reset, or BUS DEVICE RESET occurred
2a	00	MODE SELECT parameters changed by another initiator
2a	01	Mode parameters changed
2a	02	Log parameters changed
2b	00	COPY cannot execute since host cannot disconnect
2c	00	Command sequence error
2d	00	Overwrite error on update in place
2f	00	Tagged commands cleared by another initiator

Table 4–20, Additional Sense Codes and Descriptions (continued)

Byte 12	Byte 13	Description
30	00	Incompatible medium installed
30	01	Cannot read medium—unknown format
30	02	Cannot read medium—incomplete format
30	03	Cleaning cartridge installed
31	00	Medium format corrupted
33	00	Tape length error
37	00	Rounded parameter
39	00	Saving parameters not supported
3a	00	Medium not present
3b	00	Sequential positioning error
3b	01	Tape position error at Beginning-of-Tape
3b	02	Tape position error at End-of-Tape
3b	08	Reposition error
3d	00	Invalid bits in IDENTIFY message
3e	00	Logical unit has not self-configured yet
3f	00	Target operating conditions have changed
3f	01	Microcode has been changed
3f	02	Changed operating definition
3f	03	INQUIRY data has changed
40	nn	Diagnostic failure on component nn (0x80-0xff)
43	00	Message error
44	00	Internal target failure
45	00	Select/reselect failure
46	00	Unsuccessful soft reset
47	00	SCSI parity error
48	00	Initiator detected error message received
49	00	Invalid message error

Table 4–20, Additional Sense Codes and Descriptions (continued)

Byte 12	Byte 13	Description
4a	00	Command phase error
4b	00	Data phase error
4c	00	Logical unit failed self-configuration
4e	00	Overlapped commands attempted
50	00	Write append error
50	01	Write append position error
50	02	Timer position error
51	00	Erase fault
52	00	Cartridge fault
53	00	Media load/eject failed
53	01	Unload tape failure
53	02	Medium removal prevented (*DAT only*)
5a	00	Operator request or state change input unspecified (*3480 only*)
5a	01	Operator medium removal request
5a	02	Operator selected write protect (*3480 only*)
5a	03	Operator selected write permit (*3480 only*)
5b	00	Log execution
5b	01	Threshold condition met
5b	02	Log counter at maximum
5b	03	Log list codes exhausted

NOTE

Byte 14 (FRU codes) applies only to the 3480-compatible subsystem.

Byte 14 (FRU codes) Nonzero values in the FRU field are used to define a specific FRU or FRU-pair that has failed. The FRU byte contains two nibbles of information. The low-order nibble indicates the highest probability FRU. The high-order nibble indicates a secondary FRU that may also be responsible for the reported failure. Table 4–21 lists the FRU codes for the two nibbles:

Table 4–21, FRU Codes for Byte 14 Nibbles

Code	FRU	Description
1	SI or DI PCB	The SI or DI PCB contains the SCSI interface processor (68000), associated ROM/RAM, SCSI interface drivers or Receivers, and the SCSI interface chip. The SI card is used in single-ended controllers. The DI card is used in differential controllers.
2	CF PCB	The CF PCB contains the formatter microprocessor (MPU), associated ROM/RAM, data buffer, and the Buffer Function Chip (BFC).
3	FM PCB	The FM PCB contains the formatter digital read/write interface, the MTU serial interface, and the MTU parallel interface.
4	RA PCB	The RA PCB contains the formatter analog read circuitry.
5	XL PCB	The XL PCB contains the compression hardware.
6	OP Panel	The OP Panel is not diagnosed.
7	Power Supply	The power supplies are not diagnosed.
8-D	Fans	The fans are not reported in Sense.

Bytes 15-18 Not applicable for this diagnostic.

NOTE

Byte 19 (ERPA code) applies only to the 3480-compatible subsystem.

Byte 19 (ERPA code)

Byte 19 identifies the Error Recovery Procedure Action (ERPA) code. A list of the codes and their meanings follows:

ERPA code 0x22

Path equipment check.

One or more of the following errors cause this error code:

- Drive adapter error occurred.
- System could not recover from a buffer error on the lower interface.
- System could not use internal path (sense byte 2 identifies the path in error).

ERPA code 0x23

Read data check.

A permanent read error has occurred, or a temporary read error occurred with one of the following conditions:

- The controlling computer had inhibited control-unit error recovery with Mode Set bit <7>.
- Tape synchronous mode was in effect.

ERPA code 0x24

Load display check.

A *LOAD DISPLAY* command is received by a drive while a cartridge is being loaded.

ERPA code 0x25

Write data check. One or more of the following errors can cause this error code:

- Buffered data could not be written on the tape successfully. ERP has tried to erase gaps and rewrites but could not complete the write operation.
- A permanent error occurred when trying to write data, an IBG, or a tape mark on the tape. All attempts to retry the operation have been completed but were unsuccessful.
- A temporary write error occurred with one of the following conditions: The controlling computer had inhibited control-unit error recovery by Mode Set bit <1>, or Tape synchronous mode was in effect.

ERPA code 0x26

Data check (read opposite).

A read recovery is in progress, and a *READ* command (in the opposite direction) must be issued to the subsystem before the data can be recovered.

If the command at CCW address pointer -8 is 0x02 (Read), issue a 0x0c (Read Backward) chained to a 0x37 (Forward Space Block).

If the command at CCW address pointer -8 is 0x0c (Read Backward), issue a 0x02 (Read) chained to a 0x27 (Backspace Block).

If the controlling computer cannot issue a command to the subsystem to read the block in the opposite direction, a permanent OBR record is entered. If the subsystem cannot complete the command to read the record in the opposite direction, a unit check is issued and the associated sense information contains the ERPA code.

ERPA code 0x27

Command reject.

ERPA code 0x28

Write ID mark check.

The ID mark could not be written successfully at the Beginning-of-Tape (BOT). Any data to be written to the drive is still in the buffer.

ERPA code 0x2c

Permanent equipment check.

Either the control unit cannot recover because an error occurred in the subsystem hardware or microprogram, or the control unit recovery action was unsuccessful.

ERPA code 0x2d

Data security *ERASE* command failure.

The drive became "not ready" after the command was issued, or an error occurred while the command was processing.

ERPA code 0x2e

Not capable (BOT error).

Either a density mark could not be read correctly or the block ID read by the control unit is invalid (bit <0> or bits <8..11> are not zero).

If a density could not be read correctly, likely causes are:

- A void occurred at BOT.
- A time-out occurred before the density separator was detected.

ERPA code 0x30

File protected.

A write type operation was attempted on a tape cartridge that is file protected.

ERPA code 0x31
Tape void.

No patterns or data were found on the tape during a read operation. The tape could be positioned after the last data block or tape mark that was written on the tape.

ERPA code 0x32
Load assistance.

An error caused the drive to lose tape tension.

ERPA code 0x33
Load failure.

The cartridge is not inserted or threaded correctly.

ERPA code 0x34
Manual unload.

The drive cannot maintain tape tension and control tape movement during an unload operation.

ERPA code 0x35
Drive equipment check.

One of the following has occurred:

- The control unit cannot recover from a drive-detected error.
- A check code message is displayed on the drive message display and a *LOAD DISPLAY* command is issued (drive display is busy).
- A failure occurred during an index/load or unload cycle. The tape cartridge is not manually retrievable by the operator.

ERPA code 0x37
Tape length error.

The tape length in the cartridge is too short. This error could occur when the leader block was replaced (the length of tape ahead of the BOT has been trimmed).

ERPA code 0x38
Physical EOT.

A read or write operation was in process when the physical EOT pattern was reached. The drive does not pull the tape out of the cartridge.

ERPA code 0x39
Backward at Beginning-of-Tape (BOT).

While the tape was moving backwards, the BOT pattern was reached.

ERPA code 0x3a
Drive reset by operator.

The drive reset switch was activated, and the drive is not ready.

ERPA code 0x3b
Volume removed by operator.

The Rewind Unload switch on the drive has been activated and the cartridge is unloaded.

ERPA code 0x41
Block ID sequence error.

The control unit detected an incorrect block ID sequence.

ERPA code 0x43
Intervention required.

A Start I/O or Start Subchannel instruction was received by a drive that is not ready.

ERPA code 0x44
Locate block unsuccessful.

The control unit cannot find the block preceding the desired block.

ERPA code 0x46
Drive not online.

A command was issued to a drive that is not online. One of the following has occurred:

- The drive is switched offline.
- The drive power is switched off.
- The drive address is set incorrectly.

ERPA code 0x47
Control unit error.

The control unit developed an error that caused it to initialize itself again and continue.

ERPA code 0x49
Bus out parity.

The bus parity error was detected on the command or parameter transfer.

ERPA code 0x4a
Control unit ERP failed.

The control unit could not recover from a data handling failure.

Bytes 20 – 43
Not applicable for this diagnostic.

4.15 Error Examples

This section gives various examples of errors that may be encountered.

Figure 4–12 shows the error message received when a SCSI host adapter board is not present in the VMEbus chassis or there is no power to the VMEbus chassis:

Figure 4–12, Host Adapter Missing or Unpowered VMEbus Error

```
Loading CCU(s) ... Done
Bus error

Error: Egos write to adapter reset port failed - adapter installed correctly?
     :  from boot_ccu1
dev_vscsit: ccu_init failed!
```

Figure 4–13 shows an example of an error message received when a command does not complete in the allotted time or the CCU (VIOP) is not responding to the command:

Figure 4–13, Command Not Completed Error Example

```
Error: (0x6) timeout waiting for message
     : cmi operation 0x201e IO_RDSTATS_PHYSICAL: read statistics
     : cmd specific modifier 0x01  cmd common modifier 0x32
     : from sc_wrt_eof
```

Figure 4–14 show an example of an error message received when the wrong driver file is loaded:

Figure 4–14, Wrong Driver File Loaded Error Example

```
Error: cmi code 0x0006 device driver not found
     : error count 0x01  status modifier 0x05  extended status 0x00000000
     : cmi operation 0x0111 IO_INIT: probe, attach, etc.
     : cmd specific modifier 0x01  cmd common modifier 0x30
     : from xinit_driver module(b)
dev_vscsit: ccu_init failed!
```

Index