Release Notice
C1 System Diagnostics V6.6

Document No. 760-001630-000

June 27, 1990

CONVEX Computer Corporation
Richardson, Texas USA

Table of Contents

Appendices

1

Release Notice

1. Introduction

This document describes release V6.6 of C1 Series System Diagnostics. This document is intended to enhance and clarify the existing permanent documentation for this product with information that is up-to-the-minute or was developed too late for inclusion in the permanent documentation. Always refer to this release notice before reporting questions or problems with C1 Series System Diagnostics. Your questions may be answered here. Fixes and workarounds are listed here that may save you time in rediscovering known problems.

The remaining sections in this document describe the contents of this release:

- Section 2 describes the contents of this distribution.
- Section 3 contains notes and warnings about the use of the software.
- Section 4 contains enhancements to the previous functionality.
- Section 5 describes fixes for previously reported problems.
- Section 6 describes known software problems.
- Section 7 contains known documentation problems.
- Section 8 contains information on new documentation.
- Appendix A contains instructions for installing this release on a C1 Service Processor Unit (SPU).
- Appendix B contains a list of the files contained on the release tape.
- Appendix C contains updates to the current documentation.

C1 Series System Diagnostics consists of various functional-level test programs and diagnostic utilities that execute under the UNIX Version 7 operating system of the SPU. All programs MUST execute in the offline diagnostic environment of SPU UNIX and are mutually exclusive with the operation of ConvexOS on the C1. These programs are the property of CONVEX Computer Corporation and are intended for use only by CONVEX Field Service.

2. Contents of This Distribution

The release package for this release of C1 Series System Diagnostics consists of this document, distribution media for the software, and documentation. The software distribution media is dependent upon the type of SPU tape drive installed in the system. If you already have C1 System Diagnostics, the documentation updates are included in the release notice. The specific contents of the software and documentation distribution are described in the following tables:

C1 Series System Diagnostics Software (CT Type)

ITEM	QTY	TYPE	PART NUMBER	DESCRIPTION	FORMAT
1.	1	CT	760-000015-205	C1 Series System Diagnostics, V6.6	Installsw

C1 Series System Diagnostics Software (MT Type)

ITEM	QTY	TYPE	PART NUMBER	DESCRIPTION	FORMAT
1.	1	MT	760-002015-202	C1 Series System Diagnostics, V6.6	Installsw

C1 Series System Diagnostics Documentation

The C1 Series diagnostics documentation is arranged into a kit of four manuals and is named *CONVEX Diagnostics Documentation (C1, C120)*. Each individual manual is listed in the following table with its individual part number. The kit can be ordered in entirety with its product number (DHW-070).

Release Package

ITEM	QTY	TYPE	PART NUMBER	DESCRIPTION
1.	1	Manl.	760-000003-200	CONVEX Diagnostics Documentation Overview(C1, C120), 1st ed.
2.	1	Manl.	760-001050-201	CONVEX Diagnostic Utilities Manual (C1, C120), 2nd ed.
3.	1	Manl.	760-000950-200	CONVEX Processor Diagnostics Manual (C1, C120), V1.0
4.	1	Manl.	760-000730-000	CONVEX PBUS I/O System Diagnostics Manual, 3rd ed.

3. Notes and Warnings

This section contains general useful information or words of caution about the product.

- This release supercedes all previous versions of System Diagnostics.
- **Pre-requisites:**
  - SPU UNIX V5.2 or later.
  - For systems with a 6 mip FSAU, this release requires Diagnostic Database V2.6 (760-001515-201: cipher) or later. **NOTE:** Systems with a 4 mip ASU (400-001129-200) may require Diagnostic Database V1.15 (760-001415-201: cipher) or later. Consult the appropriate C-1 CPU Logic Configurator for more details.
- The installation of this release results in the deletion of the following directories prior to loading the V6.6 files from the release tape:
  - /mnt/bin
  - /mnt/test
- Beginning with this release, the installation procedure has been changed to install only those diagnostics needed by the target system. Previously, each diagnostic was installed, regardless of whether or not the target system was configured with the equipment the diagnostic was designed to test. With the addition of the **dev5300**, **dev_v3480** and **dev_ultra** diagnostics to this release, the 20-megabyte SPU disk is no longer large enough to contain all released diagnostics and still provide required support for online services.
- **cpu4000** subtest 395 fails on systems with ASU 400-001129-200 if the revision of the Diagnostics Database is V1.12 or earlier. This subtest should pass for both ASU 400-001129-200 and 400-001134-200 if used with Diagnostics Database V2.1 or later. The installed version of the Diagnostics Database can be determined by issuing the SPU UNIX command:

```
(spu)> cat /mnt/DIAG_DB_REV
```

- This release requires assembly revision "b" or later for MAU 410-000102-200 and MAU 410-000103-200, if MCU 410-001136-200 (any assembly revision) is used and if a memory interleave factor greater than 4 is desired. Use of assembly revision "a" will not allow an interleave factor greater than 4. Note that any MAU whose *cop* chip contains incorrect information could also prevent higher memory interleave factors.

- This release should only be installed by a qualified CONVEX Field Service representative. Please see Appendix A for installation details.

4. Enhancements

Files

- /mnt/bin/lib/DB_diskfmt was changed to add support for the 780MB RDS drive.

Utilities

- All test programs now permit subunit declarations in the /ioconfig file. These declarations are used by dev_v3480, but the code that parses the /ioconfig file is shared by all diagnostics that read this file, so the capability of interpreting a subunit declaration was added to all of them.

Test Programs

- The **dev5300** diagnostic program is being released for the first time. This is the functional test for the VME Async controller. There are 3 subtest classes in this diagnostic. Class 1 includes the basic controller tests involving EPROM, parity, interrupt, and RAM. Class 2 tests download and execute firmware on the controller. Class 3 tests are the printer FIFO and interface tests.

- The **dev_ultra** diagnostic program is being released for the first time. This is the functional test for the VME Ultranet controller. There are 3 subtest classes in this diagnostic. Class 1 is the controller reset test. Class 2 includes EPROM based tests with internal and external loopback. Class 3 tests the ability of the controller to address the VME bus.

- The **dev_v3480** diagnostic program is being released for the first time. This is the functional test for the VMEbus SCSI, 3480-compatible tape controller. There are 4 subtest classes in this diagnostic. Class 1 includes the host adapter self tests. Class 2 includes the tape controller self tests. Class 3 includes the tape drive tests, and Class 4 includes the tape drive exception tests.

  dev_v3480 uses the ConvexOS VIOP driver that resides in */mnt/os/viop* on the SPU disk. If the 3480 driver has not been incorporated into the VIOP driver, this diagnostic will fail with the error message "device driver not found". For release 8.1 of ConvexOS, the installation procedure for the 3480 software includes both the **dev_v3480** diagnostic and the required update for the */mnt/os/viop* file. For later releases of ConvexOS, the VIOP driver will be compatible with **dev_v3480**.

5. Fixes

Utilities

- None.

Test Programs

- **io4000** now automatically scans for the Multibus chassis in the MBCU voltage test. Some trap handling code was also fixed. Memory errors are now reported to the console before the test exits.

- **io4120** no longer gives a bus error when subtest 4240 is the first test to be run. There was a bug fix to prevent SIGIOT from causing a core dump.

- **io4130** was deleted from this release and will not appear in any future releases of C1 series diagnostics. **io4130** tested the echo64 board in a HSP chassis. The test was deleted because it was outdated by the current version of **io4120**.

- **dev4200** subtests 360 (800bpi Write Chained Mode Data) and 361 (800bpi Read Chained Mode Data) are shorter so they don't run off the end of a 600 foot tape. Also, a bug that caused subtest 430 to fail when used with a 2920 drive has been corrected. Drive type defaults now derive from /ioconfig. Added subtest 570 for read underrun support. Dropped CDC drive from drive type menu.

- **dev4500** now recognizes COV-002 as a controller in /ioconfig.

- Formerly, when a 68000 DTACK timeout (e.g., bus error) occurred on an IOP during an invocation of **dev4600** (the DR11W test), the test would report a SPU UNIX error. Now it correctly reports an IOP bus error.

- **io5000** now passes the line clock test (subtest 230) when used with a 50Hz machine. The parity check test (subtest 261) was enabled. The address increments reported by subtest 101 were corrected. Out of tolerance margin ground and voltage reference readings are now reported. A problem with trap handling that caused some to be missed was corrected. Also, some errors in the user prompts were fixed.

- **dev5130** has been changed to bypass the attempt to read the original manufacturer's defect list from a drive being formatted if defects have already been entered by hand. The attempted read formerly occurred on exiting subtest 300 after a "continue" command. This change prevents a program failure that occurred when two Multibus-formatted drives were connected to one controller and a VME format was attempted.

- **dev5210** includes the following fixes:

  - An End-of-Tape (EOT) detection bug in the CCU driver was fixed. The bug would intermittently cause the test tape to run off the feeder spool.
  - The reported block size in the output of the variable length read/write subtests was corrected. Previously, the reported block size was 1 byte larger than the actual value.
  - The CCU driver definition of a formatter-correctable media error was corrected. In addition, formatter-correctable errors on reads are not retried any longer.
  - The CCU driver was updated to use the same write retry algorithm as used by ConvexOS.
  - The "Real Time Media Error Logging" was changed to display the current

status of the controller registers, as well as the tape position if known.
- Error reports are copied to the log file /tmp/dev5210.err so that error reports from verbose subtests don't scroll off the screen.
- The EOT subtest now attempts to write one record after EOT has been detected. The subtest also now reports which record EOT was detected on.
- A "continue" command was added to the debugger. The use of continue allows the user to preserve the current skip options.
- The "skip" command in the debugger was modified to allow the user to print the current skip options and current debug mask options.
- Certain VBTC register mnemonics were changed: "pmode" became "mode", "emode" became "csb", and "csb" became "esb". This was done to provide similarity with the ConvexOS driver.
- Any internal error in a subtest now causes the subtest to fail.
- Chain mode subtests now print the record size being used.
- Subtests 608 and 609 were added to test chain mode operations that terminate on or near a chain mode boundary. Subtest 608 checks chain mode writes, and subtest 609 checks chain mode reads.
- A new debug flag was added that will cause the **dev5210** to automatically go into debug mode during a pause. In addition, if the file dev5210.auto exists, it is executed as a debugger script.
- The "echo" command in the debugger was fixed to not core dump if no arguments are specified.
- Register dumps now print current density selection, decoded command codes, and counts in both decimal and hex.
- Subtest 700 was modified to print the velocity at which data is being written. Previously, there was no way to determine the tape velocity.
- A bug was fixed which caused intermittent core dumps when the debug option "disable multi-record buffering" was selected.
- Media error summaries are printed at the end of each subtest.
- An option was added to limit the number of recovered media errors during one invocation of the test.
- A "boot" command was added to the debugger. This command allows the CCU to be rebooted from within the debugger.
- Class 6 subtests now run in PE density instead of the least dense mode available. This only affects the STC 1968 drive, since all other drives supported by **dev5210** have PE as their lowest available density.
- Subtest 610 was added to verify that an STC CLR command is no longer required before a chain mode operation. This applies to assembly rev K or higher VBTCs.
- The VBTC assembly rev register was added for assembly rev K or higher VBTCs. This register is now printed out in addition to the VME chassis slot.
- Memory dumps can now be displayed by the following command in the debugger: "mb <start>,<count>".
- Tape block values can now be followed by "k", implying the value multiplied by 1024, or "m", implying the value multiplied by 1024 * 1024.
- Formerly, Subtests 604 and 610 both induced media errors during a tape read operation by rewinding the tape to physical BOT, writing a long block, rewinding the tape to physical BOT again and writing a short block. If a read was performed after the previous write, it would fail (intentionally). Due to the possibility of a large erase gap between phsycial BOT and the first block, the second rewind in the above sequence was changed to a backspace block. This insures the second (short) block starts at the same position as the first (long) block.

- **dev5500** now recognizes COV-003 as a controller in /ioconfig.

- **security_clear** now permits user specified data patterns for clearing CCU local RAMs.

6. Known Software Problems

As of the time of the preparation of this release notice, this section contains the known problems with C1 Series System Diagnostics V6.6 software. Please refer to this list prior to reporting a problem in order to ensure that it has not been previously reported. Serious problems include workarounds if they are known.

Utilities

- There is an IOP hardware problem that causes **boot_iop**, when run with the **-t** option, to fail intermittently at 5 MHz only. This hardware problem does not affect IOP operation at other frequencies.

Test Programs

- This affects all test programs that utilize the user input query-prompt interface to acquire test parameters from the user. When a parameter with a leading "08" is entered in response to a prompt for hexadecimal input, the "08" may be ignored but the rest of the hexadecimal characters are accepted by the user interface. For example, if the prompt is "Controller offset in Multibus [0xf-0xfff] (0xc0)" and "080" is entered, the entry is interpreted as "0" since the "08" is ignored. This problem can be avoided by prefixing the hexadecimal input with "0x" (in this example, enter "0x80") or by omitting the leading zero (enter "80").

- **cpu4000** subtest 395 fails on systems with ASU 400-001129-200 if the revision of the Diagnostics Database is V1.12 or earlier. This subtest should pass for both ASU 400-001129-200 and 400-001134-200 if used with Diagnostics Database V2.1 or later. The installed version of the Diagnostics Database can be determined by issuing the SPU UNIX command:

  (spu)> cat /mnt/DIAG_DB_REV

- **io4000** subtest 302 fails if the first block of main memory is not present.

- There is a known hardware problem with IOPs when run at 5 MHz. **io4000** will intermittently fail due to this problem. This hardware problem does not affect IOP operation at other frequencies.

- **mem4000** Class 5 subtests do not support multiple failures per subtest (as set by the "log -s" command in the **dshell**). Instead, these subtests always return after the first detected failure. Also, these subtests deal with multiple bit errors improperly.

7. Known Documentation Problems

- **dev5300** is new for this release, but since it had been previously released for the C2 series computers, the documentation already exists in the CONVEX PBUS I/O System Diagnostics Manual. Since that documentation was printed, subtests 115 and 116 have been deleted from the diagnostic.

8. New Documentation

- See Appendix C of this release notice for updates to the current documentation.

A

Installing C1 Series System Diagnostics V6.6

Warnings

This section contains general useful information or words of caution about the product.

- This release supercedes all previous versions of System Diagnostics.
- **Pre-requisites:**
  - SPU UNIX V5.2 or later.
  - For systems with a 6 mip FSAU, this release requires Diagnostic Database V2.6 (760-001515-201: cipher) or later. **NOTE:** Systems with a 4 mip ASU (400-001129-200) may require Diagnostic Database V1.15 (760-001415-201: cipher) or later. Consult the appropriate C-1 CPU Logic Configurator for more details.
- The installation of this release results in the deletion of the following directories prior to loading the V6.6 files from the release tape:
  - /mnt/bin
  - /mnt/test
- Beginning with this release, the installation procedure has been changed to install only those diagnostics needed by the target system. Previously, each diagnostic was installed, regardless of whether or not the target system was configured with the equipment the diagnostic was designed to test. With the addition of the **dev5300**, **dev_v3480** and **dev_ultra** diagnostics to this release, the 20-megabyte SPU disk is no longer large enough to contain all released diagnostics and still provide required support for online services.
- **cpu4000** subtest 395 fails on systems with ASU 400-001129-200 if the revision of the Diagnostics Database is V1.12 or earlier. This subtest should pass for both ASU 400-001129-200 and 400-001134-200 if used with Diagnostics Database V2.1 or later. The installed version of the Diagnostics Database can be determined by issuing the SPU UNIX command:

  (spu)> cat /mnt/DIAG_DB_REV

- This release requires assembly revision "b" or later for MAU 410-000102-200 and MAU 410-000103-200, if MCU 410-001136-200 (any assembly revision) is used and if a memory interleave factor greater than 4 is desired. Use of assembly revision "a" will not allow an interleave factor greater than 4. Note that any MAU whose *cop* chip contains incorrect information could also prevent higher memory interleave factors.

- This release should only be installed by a qualified CONVEX Field Service representative. Please see Appendix A for installation details.

Installation Procedure

This section describes how to install the V6.6 release. Unless otherwise noted, these instructions apply to both the Archive (QIC) and Cipher (original) tape drives.

1. Verify SPU UNIX V5.2 or later is installed:

   (spu)> more /UNIX_REV

   If the file */UNIX_REV* is not present or the version number displayed for SPU UNIX is not V5.2 or later, then install SPU UNIX V5.2 (760-001615-200: boot, 760-001715-200: qic) or later in accordance with the SPU UNIX V5.2 Release Notice (760-001218-200). If SPU UNIX V5.2 or later is booted, go to step 5.

2. Place the front panel key switch in the *local* position and depress the system reset button to boot SPU UNIX.

3. The soft front panel menu will be displayed. Change the mode to diagnostics and continue the boot process by entering the following commands at the **(fp)>** prompt:

   (fp)> set mode=diagnostic (or sm=d)
   (fp)> boot (or b)

4. The SPU UNIX bootstrap routine will prompt with:

   SPU UNIX boot
   :

   You should enter a carriage return <CR> in response to the prompt. SPU UNIX will now boot and prompt with **(spu)>** when boot is complete.

   NOTE: A file system check is performed during the boot procedure. If errors are detected in the file system, they will be corrected if possible. If it is not possible to automatically correct the errors, then you will be requested to execute */etc/fsck* manually to correct these errors before proceeding.

5. If steps 2-4 were skipped, then execute */etc/fsck* manually and correct all errors before proceeding:

   (spu)> /etc/fsck -y

   If errors are detected in the root file system, then reboot SPU UNIX via */etc/reboot*:

   (spu)> /etc/reboot -n

6. Place the C1 Series System Diagnostics V6.6 tape in the cartridge tape unit. If the tape is Archive (QIC) format, first enter the following command:

(spu)> mt rew

Then, for either Archive or Cipher format tapes, enter the following command:

(spu)> /etc/installsw -i

The installation script has been changed for this release. Previously, at this point in the procedure, the contents of /mnt/bin and /mnt/test would be removed and all of the diagnostics on the release tape would be loaded onto the spu disk in these directories. The new procedure requires that the operator select the diagnostics to be installed before any files are copied to the spu disk.

After a short wait (which, in the case of the Cipher format tape, will include retensioning time), the installation script will display a prompt that includes the version number and date of the new diagnostics, and a prompt that asks

Do you want to Install or Update? [Install]

The prompts in this script can be selected by pressing the first letter of the desired choice (upper or lower case), or the default (in square brackets) can be selected by simply pressing the return key. Selecting "Install" will result in the removal of the contents of /mnt/test and /mnt/bin after the module selection step described below; this was also done by previous release procedures. If the "Update" option is selected, the contents of these two directories will not be removed; instead, the diagnostic modules selected at the next prompt will be added to the spu disk. "Update" is intended for adding functionality when the same release level has been partially installed at an earlier time, such as when hardware is added to the system. The "Update" option will fail if the revision level of the diagnostics already on the spu disk is older than that of the tape being used for the attempted update.

If "Install" has been selected, the diagnostic module selection prompt first lists all the diagnostics that should be needed by the system, based on the contents of the /ioconfig file on the spu disk. If "Update" was selected, there is no initial list. Selecting "Continue" at this point will cause the displayed modules to be installed, and the installation script will terminate. Other options are "List available", which shows all modules on the release tape; "Add", which prompts for a module to be added to the installation list; "Delete", which prompts for a module to be removed from the installation list; and "Help", which gives a detailed description of the commands that are available. When the desired modules are on the installation list, select "Continue".

7. After installation is complete, remove the tape from the cartridge tape unit.

8. If the desired mode of operation is diagnostic mode, then this step may be skipped. Otherwise, return to the soft front panel via the */etc/reboot* command:

   (spu)> /etc/reboot

   Change the mode of operation setting to the *desired-mode*. Use the soft front panel *help* command if you need assistance.

   (fp)> set mode=*desired-mode*

   Place the front panel key switch in the *secure* position and enter the *boot* command to reboot the system:

   (fp)> boot (or b)

9. This completes the installation of C1 Series System Diagnostics V6.6.

B

Files list

The following is the directory listing output from the V6.6 System Diagnostics release tree.

USER: root, DATE: Thu Jun 28 17:45:41 CDT 1990
Release directory: /c1master/rel/spu_V6.6/spu
make reltape TAPECMD=installsw TAPEDEVICE=ct
installsw -f spu_in_file -d ct

---

a /tmp/install1 1 blocks
a /tmp/install2 33 blocks
a bin/lib/get_defects.x00 111 blocks
a bin/lib/HSPutil 74 blocks
a bin/lib/IOPutil 42 blocks
a bin/lib/DB_diskfmt 6 blocks
a bin/lib/libccu.causes 1 blocks
a bin/lib/controllers 3 blocks
a bin/lib/security_clear/purge_patterns 1 blocks
a bin/lib/security_clear/purge_cpu 33 blocks
a bin/lib/security_clear/mm_purge_ccu 2 blocks
a bin/lib/security_clear/purge_iop 61 blocks
a bin/lib/security_clear/purge_viop 59 blocks
a bin/lib/security_clear/purge_hsp 66 blocks
a bin/lib/security_clear/mm_purge_spu 12 blocks
a bin/lib/vioputil.x00 45 blocks
a bin/lib/dev4100.help 8 blocks
a bin/lib/dev4100.causes 6 blocks
a bin/lib/dev4110.help 7 blocks
a bin/lib/dev4200.help 5 blocks
a bin/lib/dev4300.help 4 blocks
a bin/lib/dev4400.help 3 blocks
a bin/lib/dev4410.help 10 blocks
a bin/lib/dev4500.help 3 blocks
a bin/lib/dev4510.help 5 blocks
a bin/lib/dev4600.help 5 blocks
a bin/lib/dev5130.help 9 blocks
a bin/lib/dev5130.causes 3 blocks
a bin/lib/cop.x00 21 blocks
a bin/lib/mmvinit 33 blocks
a bin/lib/mm.help 2 blocks
a bin/lib/scan.help1 35 blocks
a bin/lib/scan.help2 3 blocks
a bin/lib/lib 4 blocks
a bin/lib/dev4540.help 14 blocks
a bin/lib/dev5210.help 41 blocks
a bin/lib/DBtapefmt 11 blocks
a bin/lib/dev5210.dhelp 20 blocks
a bin/lib/dev5300.help 14 blocks
a bin/lib/dev5300.dhelp 24 blocks

a bin/lib/dev_ultra.help 11 blocks
a bin/lib/dev_v3480.help 10 blocks
a bin/lib/dev_v3480.dhelp 28 blocks
a bin/boot_hsp 124 blocks
a bin/boot_iop 135 blocks
a bin/dshell 96 blocks
a bin/get_defects 215 blocks
a bin/hsputil 84 blocks
a bin/ioputil 41 blocks
a bin/pup 93 blocks
a bin/RC_Xfer 57 blocks
a bin/RC_Talk 25 blocks
a bin/security_clear 322 blocks
a bin/sfpread 103 blocks
a bin/vioputil 31 blocks
a bin/x 27 blocks
a bin/cop 144 blocks
a bin/cpureg 99 blocks
a bin/.diaginit 6 blocks
a bin/errintd 100 blocks
a bin/mm_sniff 68 blocks
a bin/hard_logger 188 blocks
a bin/initall 341 blocks
a bin/epcs link to bin/initall
a bin/icache link to bin/initall
a bin/interleave link to bin/initall
a bin/margin link to bin/initall
a bin/mminit link to bin/initall
a bin/sysreset link to bin/initall
a bin/wcs link to bin/initall
a bin/vcs link to bin/initall
a bin/cs link to bin/initall
a bin/map 107 blocks
a bin/mm 138 blocks
a bin/mmld 124 blocks
a bin/ringrev_chk 24 blocks
a bin/scan 156 blocks
a bin/scanc 83 blocks
a bin/scn_ring 101 blocks
a bin/scn_util 147 blocks
a bin/scnlink 50 blocks
a bin/spuutil 98 blocks
a bin/syshalt 65 blocks
a test/dev4100.t 513 blocks
a test/dev4100.x00 122 blocks
a test/dev4110.t link to test/dev4100.t
a test/dev4200.t 390 blocks
a test/dev4200.x00 85 blocks
a test/dev4300.t 430 blocks
a test/dev4300.x00 79 blocks
a test/dev4400.t 266 blocks
a test/dev4400.x00 69 blocks
a test/dev4410.t 367 blocks
a test/dev4410.x00 75 blocks
a test/dev4500.t 291 blocks
a test/dev4500.x00 71 blocks
a test/dev4510.t 276 blocks
a test/dev4510.x00 87 blocks
a test/dev4600.t 331 blocks

a test/dev4600.x00 76 blocks
a test/dev5130.t 591 blocks
a test/dev5130.x00 119 blocks
a test/dev5500.t 293 blocks
a test/dev5500.x00 72 blocks
a test/io4000.t 301 blocks
a test/io4000.x00 31 blocks
a test/io4120.t 566 blocks
a test/io4120.x00 156 blocks
a test/io5000.t 351 blocks
a test/io5000.x00 50 blocks
a test/dev5210.t 528 blocks
a test/spu4100.t 173 blocks
a test/cpu4000.t 440 blocks
a test/cpu4000.r00 9 blocks
a test/cpu4000.r01 1361 blocks
a test/cpu4000.x10 14 blocks
a test/cpu4000.x11 11 blocks
a test/cpu4000.x12 11 blocks
a test/cpu4000.x13 11 blocks
a test/cpu4000.x14 11 blocks
a test/cpu4000.x20 1 blocks
a test/cpu4000.x21 1 blocks
a test/cpu4000.x22 1 blocks
a test/cpu4000.x23 2 blocks
a test/cpu4000.x24 1 blocks
a test/cpu4000.x25 10 blocks
a test/cpu4010.t 383 blocks
a test/cpu4010.x00 25 blocks
a test/cpu4010.x01 25 blocks
a test/cpu4030.t 395 blocks
a test/cpu4030.x00 190 blocks
a test/cpu4040.t 381 blocks
a test/cpu4040.x00 78 blocks
a test/mem4000.t 472 blocks
a test/mem4000.x00 27 blocks
a test/mem4000.x01 27 blocks
a test/spu4000.t 323 blocks
a test/dev4900.t 411 blocks
a test/online_rtc 299 blocks
a test/dev4900.x00 17 blocks
a test/dev4900.x04 17 blocks
a test/dev4540.t 289 blocks
a test/dev4540x.t link to test/dev4540.t
a test/dev4540.x00 66 blocks
a test/dev5210x.t link to test/dev5210.t
a test/dev5210.x00 108 blocks
a test/dev_v3480.t 382 blocks
a test/dev4540.xx0 19 blocks
a test/dev5300.t 308 blocks
a test/dev5300x.t link to test/dev5300.t
a test/dev5300.x00 113 blocks
a test/dev5300.xx0 29 blocks
a test/dev_ultra.t 286 blocks
a test/dev_ultrax.t link to test/dev_ultra.t
a test/dev_ultra.x00 256 blocks
a test/dev_v3480x.t link to test/dev_v3480.t

** Installsw Header File Copy **

Product: System Diagnostics, Version: V6.6
Release date: June 27, 1990
Directories: /mnt/bin, /mnt/test
SPU tape devices: /dev/rct0b - header, script
/dev/rct0e - data

release: Release tape complete. No errors detected.

C
Documentation Updates

This section contains updates to the current C1 System Diagnostics documentation.

dev_v3480

3480-Compatible Cartridge Tape and SCSI Host Adapter Test

Overview

The *dev_v3480* test is a functional test for the CONVEX VMEbus cartridge tape 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 tape controller, and tape drive; and tape motion and data transfer functions of the 3480-compatible tape subsystem. This functional test is modeled to test the 3480-compatible cartridge tape system and the VMEbus SCSI host adapter. Data is recorded on tape at a recording density of approximately 37,000 bits per inch (bpi). In addition, *dev_v3480* 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 the cable interface between the tape controller and the tape unit.

> **NOTE**
>
> The *dev_v3480* diagnostic uses the ConvexOS (VIOP) driver that normally resides in */mnt/os* on the SPU disk drive. This VIOP driver file contains the 3480-compatible driver used by the *dev_v3480* diagnostic. If a VIOP driver exists in the same directory as the *dev_v3480* 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 dev_v3480–1 lists most of the SCSI interface commands, their opcodes, and descriptions:

Table dev_v3480–1, SCSI Interface Commands

COMMAND DESCRIPTIONS			
Number	**Group Code**	**OP Code**	**Command Name**
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	ox1b	Unload
13	0	ox1c	Receive Diagnostic Results
14	0	0x1d	Send Diagnostic
15	6	0xcf	Load Display

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 Small Computer System Interface (SCSI) and 3480-compatible cartridge tape functionality. Table dev_v3480–2 lists several documents that may provide additional information that is not contained within this test description:

Table dev_v3480–2, Related Documents

Document Title	Document Origin
Fujitsu Cartridge Tape Drives CE Manual	Fujitsu
Fujitsu Cartridge Tape Controller Customer Engineering Manual	Fujitsu
Rimfire 3510 SCSI Host Bus Adapter and Floppy Disk Controller	Ciprico

Required Equipment

Table dev_v3480–3 lists the required hardware depending on the type of machine under test:

Table dev_v3480–3, Hardware Requirements

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

[1] Memory System consists of a minimum of one pair of memory boards (one odd and one even).

[2] 3480-compatible tape drive

NOTE

A maximum of 16 CONVEX 3480-compatible cartridge tape drives can be connected to a single VMEbus Input/Output Processor (VIOP).

Software Requirements

The *dev_v3480* 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

Setting the Formatter Unit Number

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

Setting the 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.

Test Invocation

The *dev_v3480* 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_v3480* test, use the procedure shown in "Figure dev_v3480–1, Initial Test Invocation Sequence." All responses in **boldface** are entered by the user. The prompts and responses appear sequentially on the screen, one line at a time. Figure dev_v3480–1 shows all the prompts and responses:

> **NOTE**
>
> Use the following test invocation sequence for the initial invocation of *dev_v3480* 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 dev_v3480–1, Initial Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> sysreset RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
:test dev_v3480.t [-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_v3480** executes all *dev_v3480* 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_v3480.tmp* is used as the parameter file.

The [+>*filename*] option allows the test results to be appended to *filename*.

NOTE

The following alternate test invocation procedure is optimal when invoking *dev_v3480* 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_v3480**. When invoking *dev_v3480* in this manner, no prompts are displayed. The diagnostic obtains all prompt information from the parameter file 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 dev_v3480–2 shows the alternate test invocation sequence:

Figure dev_v3480–2, Alternate Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
:test dev_v3480x [-c [class number(s)]] [-s [subtest number(s)]] [-dV] [-f FILE] [+>filename] RETURN
```

Test Parameter Menu

Once the test is invoked, a test menu prompt is presented allowing selection of default switches. Figure dev_v3480–3 shows the TEST PARAMETER MENU with all prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )):

Figure dev_v3480–3, 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  0xee00  3   0.2 MTD-207
        4) viop  3     1    MTC-202  0xee00  3   0.3 MTD-207

        *** Enter 0 for manual configuration ***

2: Ioconfig File Units to Test [0-4,0,?]                (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 *all* 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 dev_v3480–4 list the help characters:

Table dev_v3480–4, 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 are answered, the screen displays a TEST PARAMETER SUMMARY, that displays prompts that were answered and their responses. Figure dev_v3480–4 illustrates an example of an initial TEST PARAMETER SUMMARY screen, but the actual values and responses vary according to the input. Figure dev_v3480–4 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 dev_v3480-1 for the test invocation sequence to produce this TEST PARAMETER SUMMARY.

Figure dev_v3480–4, 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

write_options: Pfile = dev_v3480.tmp
Initialing MBS processor queues ... Done
Loading CCU(s) ... Done
CCU 3/MTC-202 configure/probed completed
Tape unit (0) attach and connect completed
Main memory data buffer address starts at 0x02002000
```

If *dev_v3480* 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 dev_v3480–5 shows a sample TEST PARAMETER SUMMARY with the CCU drivers already loaded:

> **NOTE**
>
> Refer to Figure dev_v3480-2 for the test invocation sequence to produce this TEST PARAMETER SUMMARY.

Figure dev_v3480–5, 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

write_options: Pfile = dev_v3480.tmp
*** CCU driver already loaded on VIOP 3
Main memory data buffer address starts at 0x02002000
```

Initialization Sequence for *dev_v3480*

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

- The diagnostic determines if the test was invoked for quick startup (i.e., *dev_v3480x*). If so, the diagnostic reads the test parameters from the specified parameter file (default parameter file is */tmp/dev_v3480.tmp*). If not, the input to the parameters is written to the parameter file (default or user-specified).
- The largest contiguous main memory space after the first 2 Mbytes is located and reserved for use by *dev_v3480*.
- The diagnostic checks to see if the CCU is already loaded with the *dev_v3480* CCU driver. If the driver is not loaded, the CCU is reloaded. After the load completes, the driver is configured for EGOS and then 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.

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  0xee00  3   0.2 MTD-207
4) viop  3     1    MTC-202  0xee00  3   0.3 MTD-207

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.

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

Above, 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 desire to use most (but not all) 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 ^ command), and then specify **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 permits use of 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 dev_v3480–6 shows all the options for the TEST PARAMETER SUMMARY:

Figure dev_v3480–6, 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  0xee00  3   0.2 MTD-207
         4) viop  3     1    MTC-202  0xee00  3   0.3 MTD-207

         *** Enter 0 for manual configuration ***

 2: Ioconfig File Units to Test [0-4,0,?]                (1) -> RETURN
 3: Use Defaults for Remaining Parameters [y,n,?]        (y) -> n
 4: Enable Debug Monitor [y,n,?]                         (n) -> y
 5: Variable record subtest pattern [<hexadecimal pattern>,?]
                                                    (0x6db6) -> RETURN
 6: Variable record miscompare Line Dump Count  [1-100, ?](8) -> RETURN

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

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

Figure dev_v3480–7 shows the TEST SUMMARY MENU with online help for each option:

Figure dev_v3480–7, 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 _____________________________
    This option allows you to 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  0xee00  3   0.2 MTD-207
          4) viop  3     1    MTC-202  0xee00  3   0.3 MTD-207

          *** Enter 0 for manual configuration ***

 2: Ioconfig File Units to Test [0-4,0,?]                  (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 PROM.
------------------------------ 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 with in a subset.  The
    debugger may also be invoked by specifying the "-d" option at test
    invocation time (i.e., dev_v3480[x].t -d).
------------------------------ END OF HELP ------------------------------
```

Figure dev_v3480–7, Test Parameter Menu (with Help Fields) (continued)

```
 5: Variable record subtest pattern [<hexadecimal pattern>,?]
                                                    (0x6db6) -> ?
_______________________________ HELP FOLLOWS ________________________________
    Here you are requested to enter the hexadecimal data pattern to use for
    variable record write/read 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_v3480.pat".  The pattern can contain whitespace and 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 -----------------------------
 6: Variable record miscompare Line Dump Count  [1-100, ?](8) -> ?
_______________________________ HELP FOLLOWS ________________________________
    This option specifies the MAXIMUM number of display lines to print when
    a miscompare occurs.
------------------------------ END OF HELP -----------------------------

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

 7: 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 and its data have been
    displayed.  Although any string up to 64 characters may be sent, the
    intended use is to allow the ability to selectively turn a printer on
    before an error is displayed and turn a printer off after an 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 enabled.  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 -----------------------------
```

Figure dev_v3480–7, Test Parameter Menu (with Help Fields) (continued)

```
 8: Printer On Character Sequence (before error)
    [<printer on string>,?]                          (\033[?5i) -> ?
_____________________________ HELP FOLLOWS ____________________________
    Enter the character string to be displayed before an error and its data
    are 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 -------------------------------
 9: Printer Off Character Sequence (after error)
    [<printer off string>,?]                         (\033[?4i) -> ?
_____________________________ HELP FOLLOWS ____________________________
    Enter the character string to be displayed after an error and its data
    have 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 -------------------------------
10: Enter OK, or  :NN to return to question NN [OK]       (OK) -> RETURN
```

Class Descriptions

The *dev_v3480* test contains the three classes of subtests as listed in Table dev_v3480–5:

Table dev_v3480–5, *dev_v3480* 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

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 VMEbus Input/Output Processor (VIOP) and the VMEbus Control Unit (VBCU)
- The ability of the host adapter to interrupt and to mask interrupts
- Accessibility to main memory

Table dev_v3480–6 lists all Class 1 subtests, their descriptions, and the approximate times required to execute each subtest:

Table dev_v3480–6, Class 1 Subtests

Subtest	Description	Time (min:sec)[1]
100	Host Adapter Identify Test	00:01
101	Host Adapter RAM Test	00:06
102	Host Adapter PROM Test	00:10

[1] The times presented are approximated.

Subtest 100, Host Adapter Identify Test

Subtest 100 issues an *IDENTIFY* command to the host adapter. The *IDENTIFY* command is issued 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
```

Subtest 101, Host Adapter RAM Test

Subtest 101 issues a *BOARD TEST* command to the host adapter. The *BOARD TEST* command is issued from a diagnostic parameter block that contains the 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

The *BOARD TEST* command will not execute until all preceding commands have finished to avoid writing test patterns over data. 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.

Subtest 102, Host Adapter PROM Test

Subtest 102 issues a *BOARD TEST* command to the host adapter. The *BOARD TEST* command is issued 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 500 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.
```

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

Class 2 subtests invoke the self-test capabilities of the tape controller (formatter) and the 3480-compatible cartridge tape 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 are required for this class of tests.

NOTE

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

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

Table dev_v3480–7 lists all Class 2 subtests, their descriptions, and the approximate times required to execute each subtest:

Table dev_v3480–7, Class 2 Subtests

Subtest	Description	Time (min:sec)[1]
200	Tape Controller Self-Test	00:56[2]
201	Tape Logical Unit Self-Test	15:00

[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.

Subtest 200, Tape Controller Self-Test

Subtest 200 issues a *SEND DIAGNOSTIC* command to request the tape controller (formatter) 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, respectively, to direct the targeted tape controller (formatter) 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 in the controller (formatter) self-test:

- 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.
- Formatter digital logic test
  The digital portion of the formatter logic is diagnosed.

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_v3480.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.
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
Decode of relevant request sense bytes:
Error: (sense key) Nonrecoverable hardware failure detected.
Error: (fru code) CF board (core function) contains fmt mpu, rom/ram, data buf.
     : above is primary fru.
Error: (fru code) FM board (formatter) contains digital r/w, mtu interface.
     : above is secondary fru.
Error: (erpa code) 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_v3480-16, Sense Key Codes," on page 35 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_v3480-17, Additional Sense Codes and Descriptions," on pages 36 through 39 of this chapter.
- FRU code definitions are listed in "Table dev_v3480-18, FRU Codes for Byte 14 Nibbles," on page 40 of this chapter.
- ERPA code definitions are listed on pages 40 through 43 of this chapter.

Subtest 201, Tape Logical Unit Self-Test Test

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 each of 100 bytes, 1 Kbyte, 32 Kbytes, and 64 Kbytes are written to tape.

- Routine 53 (Read Data)

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

- Routine 54 (Combination test 1)

  Various combinations of WRITE, READ, REVERSE, WRITE FILEMARK, ERASE, and SPACE 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 executing occurs. 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 dev_v3480–8 lists the format of the diagnostic data returned:

Table dev_v3480–8, 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 5	(MSB) First Symptom Code (LSB)							
6 7	(MSB) Second Symptom Code (LSB)							
8 9	(MSB) Third Symptom Code (LSB)							

The following are field descriptions for the *Receive Diagnostic Results* command:

- **Routine in Error**

  This field contains the Routine ID of the failing 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 DIAGNOST* 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

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_v3480.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_v3480.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.
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
Decode of relevant request sense bytes:
Error: (sense key) Logical unit address cannot be accessed.
Error: (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_v3480-16, Sense Key Codes," on page 35 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_v3480-17, Additional Sense Codes and Descriptions," on pages 36 through 39 of this chapter.
- ERPA code definitions are listed on pages 40 through 43 of this chapter.

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.

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 dev_v3480–9 lists all Class 3 subtests, their descriptions, and the approximate time required to execute each one:

Table dev_v3480–9, Class 3 Subtests

Subtest	Description	Time (min:sec)[1]
300	Tape Mark Test	01:17
301	Space Record Test	02:06
302	Erase Test	00:48
303	Long Block Read Test	00:15
304	Fixed Record Size Read/Write Test	04:40
305	Write File Unix Style Test	01:12
306	Variable Size Records Test	00:42

[1] The times presented are approximated.

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
```

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
```

Subtest 302, Erase Test

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
```

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 with 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 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-compatible tape drive (0x20000). A check is made to ensure that the long-block status bit is checked for a reset state 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
```

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 dev_v3480–10 lists the data pattern used:

Table dev_v3480–10, 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
```

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 each of 32 Kbytes 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 backspace filemarks. The tape is rewound. All records are read and verified and, all backspace 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
```

Subtest 306, Variable Size Records Test

Subtest 306 verifies that the variable length records on a tape can be written, read, and verified. First, records of 1 byte to 258 bytes (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 is either defaulted or that is user specified from the keyboard or from a file. The default pattern is 0x6db6. The default number of error printouts is 8 or can be user specified up to 100 error printouts. 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 65538 bytes in this subtest.

Table dev_v3480–11, 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_v3480.t  1.1  Class: 3     Subtest: 306 1.1  Count: 1    Error: 0
Failed:  Variable size records read/write test
```

Class 4 Subtests, Tape driver exception tests

Class 4 subtests test various conditions that do not usually occur during normal tape drive operations. The purpose of these exception subtests are to 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.

Table dev_v3480–12 lists all Class 4 subtests, their descriptions, and the approximate time required to execute each subtest:

Table dev_v3480–12, Class 4 Subtests

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

[1] The times presented are approximated.

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 interrogated for the EOM bit to be 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 interrogated for the EOM bit to be set. A *FORWARD SPACE RECORD* command is then issued. The BOT status bit should be reset.

Subtest 401, Zero Length Operations Exception Test

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

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

The tape is rewound and two records of known patterns of 20,000 hex bytes are written. 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.

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

The tape is rewound and multiple 20,000 hex byte records are written to tape till the EOT is sensed in the status byte. A backspace records is issued. Then a 20,000 hex byte record with a known pattern is written. An EOT status condition is expected as a result of the *WRITE* command. The tape is backspaced 1 record and the record just written is verified. A *REWIND* command is issued that completes the test.

Interactive Debugger

The diagnostic provides an interactive debugger that supports the ability to execute commands from a script file. This allows more debug flexibility. Invocation of the debugger is achieved by the following methods:

- Use the **-d** option when invoking the diagnostic (enter **dev_v3480**[x] **-d**). No subtests are executed and the debugger is invoked.
- Respond with a **y** to `Enable Debug Monitor` question in the TEST PARAMETER SUMMARY. The debugger will 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 dev_v3480–8, 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)
    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
    dfdisplload string                       - diagcmd load display on tape unit
```

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

help *command*

In the previous sequence, replace *command* with 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

Interactive Debugger Command Descriptions

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

cd

Usage: **cd** [path]

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

quit

Usage: **quit**

Exits the interactive debugger.

echo

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

Sends echo statement to the display.

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

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 (optional)
- **step** is the address increment (optional (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, and 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
<CCU: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.

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 (optional)
- **step** is the address increment (optional (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, and 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
<Main-Mem: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.

fb, fw, fl

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

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

ffb, ffw, ffl

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

Fills main memory with specified pattern in byte-at-a-time mode (ffb), word-at-a-time mode (ffw), or longword-at-a-time move (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

weof count

Usage: **weof**

Writes to the end of the file *count* number of times.

fsr, bsr, fsf, bsf

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

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

erase

Usage: **erase**

Erases a default length of tape on the selected tape drive.

wphys

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

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

rphys

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

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

rewind

Usage: **rewind**

Rewinds tape unit.

changeunit

Usage: **changeunit unit subunit**

Selects the next tape unit to be tested in the debug mode. The **unit** is the SCSI ID of the tape formatter. The **subunit** is the logical unit address of the tape drive that is connected to the

formatter.

notimeout

Usage: **notimeout 0 or 1**

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

tracemsgs

Usage: **tm 0 or tm 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.

status

Usage: **status**

Reads and returns current tape unit status.

unload

Usage: **unload**

Unloads tape from tape unit.

unitclr

Usage: **unitclr**

Clears tape error status.

boot

Usage: **boot**

A diagnostic command that forces a reboot of the CCU driver.

reset

Usage: **reset**

A diagnostic command to reset 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.

dapselftest

Usage: **dapselftest**

A diagnostic command that tells the host adapter to run its PROM self-test.

darselftest

Usage: **darselftest**

A diagnostic command that tells the host adapter to run its RAM self-test.

daidentify

Usage: **daidentify**

A diagnostic commands that tells the host adapter to report its revision number.

dfcselftest

Usage: **dfcselftest**

A diagnostic command that tells target controller (formatter) to run its self-test.

dfuselftest

Usage: **dfuselftest**

A diagnostic command that tells the target tape unit to run its self-test.

dfrecvdiag

Usage: **dfrecvdiag**

A diagnostic command that tells a target device to report its diagnostics results.

dfdisplоad

Usage: **dfdisplоad string**

A diagnostic command that tells the target device to load a *string* on its display.

Using a Test Script

Use the following procedure for running a test script and an example test script:

1. Create two files. For this example, the script files are named *scr3* and *scr3a*. Figures dev_v3480–9 and dev_v3480-10 show examples of *scr3* and *scr3a* script files:

Figure dev_v3480–9, 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 dev_v3480–10, 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_v3480* diagnostic using the debug option (**dev_v3480.t -d**).

3. Respond to the questions asked in the diagnostic normally.
4. At the **Debug Mode->** prompt, type **<scriptname**. Figure dev_v3480–11 shows the output of the example test script:

Figure dev_v3480–11, 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**.

SCSI Host Adapter Status and Error Information

The following sections include information on status and error reporting for the SCSI host adapter. Table dev_v3480–13 lists the contents of the SCSI host adapter status block:

Table dev_v3480–13, 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

Flags Byte

This byte contains flags indicating the status of the host adapter. Table dev_v3480–14 lists each flag bit in the host adapter status block flags byte:

Table dev_v3480–14, 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 brief explanation for each flag bit:

CC Command complete

ERR Error status (this command had an error)

RTY Command required one or retries

DTT Data transfer truncated (SCSI command completed, but fewer)

TAR Target mode enabled in SCSI host adapter

CSB Continued status block (for additional sense data)

SE Soft error

Error Byte

The following list contains brief explanation for 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)

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 dev_v3480–15 lists the bits in a host adapter status byte:

Table dev_v3480–15, 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 dev_v3480–16 lists the status byte codes and their descriptions:

Table dev_v3480–16, 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

Tape System Status and Error Information

The following sections include information on status and error reporting for the formatter and tape drive(s) in a cartridge tape system.

Sense Data

The sense bytes, contained in the controller, indicate error, status, and statistical information about the controller to the drive. 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 dev_v3480–17 lists the sense byte format on the current command for error code 70:

Table dev_v3480–17, 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) Information Bytes (LSB)							
7	Additional Sense Length = 0x24							
8 to 11	(MSB) 0x00 (SCSI-2 Command Specific Information Bytes (LSB)							
12	Additional Sense Code							
13	Additional Sense Code Qualifier							
14	FRU Code							
15 to 17	SKSV	(MSB) (LSB)						
18	Format of additional sense							
19	Host ERPA							
20 to 43	(MSB) Additional Sense Bytes as Defined by the format indicated in byte 18. (LSB)							

Sense Bytes Description

This section describes of all the 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> Ignore

Bits <3..0> Sense key

Table dev_v3480–18, 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

Bytes 3 – 6 (information bytes)

Byte 7 (additional sense length)

Bytes 8 – 11 (command specific information)

Bytes 12 – 13 (Additional sense code and sense code qualifiers)

Table dev_v3480–19, Additional Sense Codes and Descriptions

Byte 12	Byte 13	Description
00	00	No additional sense information
00	01	Filemark detected
00	02	End-of-Tape (EOT) detected
00	03	Beginning-of-Data (BOD) detected
00	04	Beginning-of-Tape (BOT) detected
00	05	End-of-Data (EOD) detected
03	00	Peripheral device write fault
03	01	No write current
03	02	Excessive write errors
04	00	Logical unit not ready, cause not reportable
04	01	Logical unit not ready, manual intervention required
04	02	Logical unit not ready, initializing command required
04	03	Logical unit is in process of becoming ready
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
11	08	Incomplete block read (postamble not found)
11	09	No gap found
11	0a	Miscorrected error
14	00	Record 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 dev_v3480–19, 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
17	03	Recovered read data with negative head offset
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 dev_v3480–19, 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 dev_v3480–19, 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 appended 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
5a	00	Operator request or state change input unspecified
5a	01	Operator medium removal request
5a	02	Operator selected write protect
5a	03	Operator selected write permit
5b	00	Log execution
5b	01	Threshold condition met
5b	02	Log counters at maximum
5b	03	Log list codes exhausted

Byte 14 Nonzero values in the Field Replaceable Unit (FRU) field are used to define a specific FRU or FRU-pair that has failed. The FRU byte contains two nibbles of information. The high-order nibble indicates the highest probability FRU. The low-order nibble indicates a secondary FRU that may also be responsible for the reported failure. Table dev_v3480–20 lists the FRU codes for the two nibbles:

Table dev_v3480–20, 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/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.

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.

Error Examples

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

Figure dev_v3480–12 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 dev_v3480–12, Host Adapter Missing or Powerless VMEbus Error Example

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

Error: Egos write to adapter reset port failed - adapter installed correctly?
     :  from boot_ccu1
dev_v3480: ccu_init failed!
```

Figure dev_v3480–13 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 dev_v3480–13, 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 dev_v3480–14 show an example of an error message received when the wrong driver file is loaded:

Figure dev_v3480–14, 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_v3480: ccu_init failed!
```

THIS PAGE INTENTIONALLY LEFT BLANK

dev5210

VMEbus STC Tape Unit Controller Test

Overview

The *dev5210* test is a functional test for the CONVEX VMEbus Tape Controller (VBTC) and its connected tape unit(s). The VBTC provides a connection to a magnetic tape drive unit that is based on the STC interface. In general, this test attempts to force most software-forcible formatter and controller errors and verify that either the formatter or controller can detect them.

This functional test is modeled to test the STC 2921, STC 2922, and STC 1968 drives. In addition, the test supports testing of the Fujitsu 243XL. Read and write type testing is conducted at recording densities of:

- **800 bpi**—Nonreturn to Zero (NRZI)
- **1,600 bpi**—Phase Encoded (PE)
- **6,250 bpi**—Group Coded Recording (GCR)

Specifically, *dev5210* is designed to accomplish the following:

- Verifies the functional ability of the VBTC to operate in the CONVEX VMEbus I/O environment. This includes main memory access and interrupt generation. In addition, the FIFO is pattern tested along with verification of its ability to detect parity errors.
- Verifies 14 of the 15 supported STC interface commands. The controller's ability to use data chaining on transfers and pipeline commands is tested under normal and error conditions.
- Verifies the ability of the VBTC to detect most software-forcible formatter and controller errors. It insures that the VBTC can recover from bus errors.
- Verifies that the cable interface between the controller and the drive is operational.
- Provides an interactive debugger that can execute commands from a script file. This allows more flexibility in debugging.

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

Most subtests implement an error recovery scheme to minimize the impact of media errors. Although media errors are minimized at most points the user may select a pattern that will stress data recovery. As a result, the test requires that tape media be in adequate condition. In most write subtests, identification data is written in each block. This data is used to verify positional coordinates on the tape.

The following table lists all of the STC tape commands and indicates which ones are tested:

Table dev5210–1, STC Interface Commands Tested

COMMAND DESCRIPTIONS					
Number	**Mnemonic**	**STC code**	**Function**	**Type**	**Tested**
1	*NOP*	0x00	No Operation	Diagnostic	Yes
2	*LWR*	0x07	Write to Read Loopback	Diagnostic	Yes
3	*DMS*	0x02	Diagnostic Mode Set	Diag/Norm	Yes
4	*CLR*	0x01	Drive Clear	Normal	Yes
5	*REW*	0x0e	Rewind to BOT	Normal	Yes
6	*RUN*	0x0f	Rewind to BOT and Unload	Normal	No
7	*ERG*	0x0d	Erase Gap	Normal	Yes
8	*WRT*	0x06	Write Block	Normal	Yes
9	*RDF*	0x04	Read Forward Block	Normal	Yes
10	*RDB*	0x05	Read Backward Block	Normal	Yes
11	*FSB*	0x0b	Forward Space Block	Normal	Yes
12	*BSB*	0x09	Backward Space Block	Normal	Yes
13	*WTM*	0x0c	Write Tape/File Mark	Normal	Yes
14	*FSF*	0x0a	Forward Space File	Normal	Yes
15	*BSF*	0x08	Backward Space File	Normal	Yes

Related Documents

This test description is intended as a reference for users who have a good understanding of the VBTC 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 Storage Technology Corporation's *StorageTek Standard Interface* (STC interface). The following table lists several documents that may provide additional information not contained within this test description:

Table dev5210–2, Related Documents

Document Title	Document Origin
STC Compatible Multibus Controller Specification	CONVEX
VME Tape Controller Difference Document	CONVEX
VME IOP Difference Document	CONVEX
The STC 2920 Maintenance Manual	STC
The STC 1960 Maintenance Manual	STC
The Fujitsu 2xSXL Maintenance Manual	Fujitsu

Prerequisites and Required Equipment

The following table lists the required hardware depending on the type of machine under test:

Table dev5210–3, Hardware Requirements

C1, C120	C200 Series
MCU	Memory System[1]
MAU	CPX
SPU	SP2
VIOP	VIOP
VBCU	PIA
VBTC[2]	VBCU
Tape Drive	VBTC[2]
	Tape Drive

[1] Memory System consists of a minimum of one pair of memory boards (one odd and one even).

[2] VBTC represents VMEbus Tape Controller

Test Invocation

The *dev5210* 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 *dev5210* test, use the procedure shown in Figure dev5210–1, "Initial Test Invocation Sequence." All responses in **boldface** are entered by the user. The prompts and responses appear sequentially on the screen, one line at a time. All prompts and responses are shown in one figure for convenience.

> **NOTE**
>
> Use the following test invocation sequence for the initial invocation of *dev5210* 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 dev5210–1, Initial Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> sysreset RETURN
(spu)> mminit -s RETURN
(spu)> dshell RETURN
: test dev5210 [-c [class number(s)]] [-s [subtest number(s)]] [-dnqV] [-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.

The format for the *test* command is as follows:

test dev5210 [*option* ...] [[+>*filename*]]

where *option* is one of the following:

-c *class-number*	Execute one or more specific classes of subtest(s)
-d	Enter Debugger (no subtests are executed)
-n	Do not actually execute any subtest (used to create a parameter file without actually executing any subtest)
-q	Quick startup (use previous parameters from last test invocation—same as entering **dev5210x**; refer to the "Alternate Test Invocation Sequence")
-s *subtest-number*	Execute one or more individual subtests
-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/dev5210.tmp* is used as the parameter file.

Entering only **test dev5210** executes all *dev5210* subtests sequentially. The [+>*filename*] option allows the test results to be appended to *filename*.

> **NOTE**
>
> The following alternate test invocation procedure is optimal when invoking *dev5210* multiple times. Using this invocation sequence insures that the test is invoked and executed with all of the setup parameters supplied when the test was last executed with the initial invocation sequence.

Figure dev5210–2, Alternate Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> dshell RETURN
:test dev5210x [-c [class number(s)]] [-s [subtest number(s)]] [-dnqV] [-f FILE] [+>filename] RETURN
```

The only difference in this alternate invocation sequence is the **x** after **dev5210**. When invoking *dev5210* in this manner, no prompts are displayed. The diagnostic obtains all prompt information from the parameter file created when the initial invocation sequence was performed.

Test Parameter Menu

Once the test is invoked, a test menu prompt is presented allowing selection of default switches. Figure dev5210–3, "Test Parameter Menu," shows all prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )). The prompts and responses in the figure appear sequentially on the screen, one line at a time. The figure illustrates *all* 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 following characters followed by a RETURN:

Table dev5210–4, Getting Help During Test Parameter Entry

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

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

Figure dev5210–3, 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 specific help where available

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

                  PERIPHERAL CONFIGURATION DATA
            CCU    Chassis   Type    CSR   Int Unit   Type
          -------- ------- -------- ------ --- ---- --------
         1) viop  3     0    MTC-201  0x0600  1   1   MTD-204
         2) viop  3     0    MTC-201  0x3fc0  4   0   MTD-201

        *** Enter 0 for manual configuration ***

 2: Select Configuration File Entry to Test [0,1-2,?]     (1) -> 0
 3: Display VBTC Configuration Help [y,n,?]               (y) -> n
 4: VME IOP Number [3-7,?]                                (3) -> 3
 5: VME Chassis Number [0-1,?]                            (0) -> 0
 6: VMEbus Base Address of the Controller [0x0-0xffff,?]
                                                  (0x600) -> 0x600
 7: VMEbus Interrupt Level of the Controller [1-7,?]      (1) -> 1
 8: Tape Unit Number [0-7,?]                              (1) -> 1

          ** Tape Drive models **
    1:    Storage Technology Corp. 2921 ..... MTD-201
    2:    Storage Technology Corp. 2922 ..... MTD-204
    3:    Storage Technology Corp. 1963/1968 ..... MTD-202
    4:    Fujitsu 2436-L1 ..... MTD-203

 9: Tape Drive Type [1-4,?]                               (2) -> 2
10: Use Standard Defaults for Other Options [y,n,?]       (y) -> n

          ** Tape Spool Size in Feet **
    1:     600 ft.
    2:    1200 ft.
    3:    2400 ft.
    4:    3600 ft.

11: Select Mounted Tape Spool Size [1-4,?]                (1) -> 1
12: Run EOT Sensor Subtest #701 [y,n,?]                   (y) -> y
```

Figure dev5210–3, Test Parameter Menu (continued)

```
            ** Available Data Transfer VMEbus Address Modifiers **
    1:      Standard Supervisory Access (16 bit) (0x3d)
    2:      Standard Non-Supervisory Access (16 bit) (0x39)

13: Data Transfer VMEbus Address Modifier [1-2,?]          (1) -> 1

            ** Available Tape Velocities (IPS) **
    1:       50
    2:      100

14: Standard Tape Velocity [1-2,?]                          (2) -> 2
15: Standard Tape Block Size [16-16M,?]                    (8K) -> 8K
16: Large (Gapless) Tape Block Size [256K-16M,?]           (1M) -> 16M
*** WARNING:
    Due to insufficient tape length, the selected Large Block of 16777216
    will be reduced to 5760000 when using PE   recording format.
17: Run Only High Density Gapless Subtests (520,521) [y,n,?]
                                                           (y) -> y

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

18: Select Printer On/Off Mode [0x0-0x3,?]                (0x0) -> 0x3
19: Printer On Character Sequence (before error)
    [<printer on string>,?]                          (\033[?5i) -> \033[?5i
20: Printer Off Character Sequence (after error)
    [<printer off string>,?]                         (\033[?4i) -> \033[?4i
21: Use Standard Debug Setup [y,n,?]                       (y) -> n

  ** USER Debug Options (bit mapped) **
    0x0001: Disable Asynchronous Command Mode
    0x0002: Disable Multi-Record Buffering
    0x0004: Disable Reset of Controller After ^C
    0x0008: Disable Hard Error Trapping (logging)
    0x0010: Enter Debugger After an Error Occurs
    0x0020: Disable copying of errors to error file
    0x0100: Pause AFTER each CCU driver command is returned
    0x0200: Show each CCU driver message AFTER it is returned
    0x0400: Pause BEFORE each CCU driver message is sent
    0x0800: Show each CCU driver message BEFORE it is sent
    0x1000: Automatically enter debugger during a pause

22: User Debug Option Mask [0x0-0x1fff,?]                 (0x0) -> 0x1fff

            ** VBTC register dump modes **
    1:      After Error
    2:      After Each Command Completion
    3:      Never

23: Select Controller Register Dump Mode [1-3,?]           (3) -> 3
```

Figure dev5210–3, Test Parameter Menu (continued)

```
          ** Tape Media Error Logging Methods (bit mapped) **
    0x0001: After Test Completion
    0x0002: After Each Subtest
    0x0004: Upon Occurrence (Real Time) With Limited Register Dump
    0x0008: Upon Occurrence (Real Time) With Full Register Dump
    0x0010: Activate Printer During Media Errors (See Printer Options)

24: Select Media Error Logging Method [0x0-0x1f,?]      (0x3) -> 0x1f
25: Maximum Media Error Retry Count [0-999,?]            (10) -> 10
26: Maximum # of Non-Fatal Media Errors Allowed [0-9999,?]
                                                        (40) -> 40
27: Retry Formatter Correctable Media Errors On Writes [y,n,?]
                                                         (y) -> y
28: Pattern for Write/Read Subtests [<hexadecimal pattern>,?]
                                                  (0x6db6) -> 0x6db6

          ** Available Data Miscompare Dump Modes **
    1:      Separate Expected and Actual Values into Two Blocks.
    2:      Mix Expected and Actual Values (xx/xx).
    3:      Only Show Bytes that Miscompare.

29: Select Data Miscompare Display Mode [1-3,?]           (2) -> 2
30: Enter Data Miscompare Line Dump Count [1-1000,?]      (4) -> 4
31: Enter OK, or :NN to return to question NN [OK]       (OK) -> RETURN
```

NOTE

In the previous figure, a warning statement (see prompt #16) is printed because the selected tape length for the displayed density is not large enough to hold the large tape block selected. The test automatically compensates for this by changing the large tape block size to match the tape length. In all cases, the maximum large tape block size is limited by the amount of contiguous physical memory present in the target system and the maximum tape length. It is important to note that *dev5210* only compensates for insufficient tape length in the data chain mode read/write tests.

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

When all prompts are answered, the screen displays a test parameter summary which displays the prompts that were answered and their responses. Figure dev5210–4, "Sample Test Parameter Summary," shows a sample test parameter summary. The actual values and responses vary according to the input.

Figure dev5210–4, Sample Test Parameter Summary

```
                        TEST PARAMETER SUMMARY

Select Configuration File                                    : /ioconfig
Select Configuration File Entry to Test                      : 0
Display VBTC Configuration Help                              : y
VME IOP Number                                               : 3
VME Chassis Number                                           : 0
VMEbus Base Address of the Controller                        : 0x600
VMEbus Interrupt Level of the Controller                     : 1
Tape Unit Number                                             : 1
Tape Drive Type                                              : 2
          -> Storage Technology Corp. 2922 <-
Use Standard Defaults for Other Options                      : n
Select Mounted Tape Spool Size                               : 1
          ->  600 ft. <-
Run EOT Sensor Subtest #701                                  : y
Data Transfer VMEbus Address Modifier                        : 1
          -> Standard Supervisory Access (16 bit) (0x3d) <-
Standard Tape Velocity                                       : 2 (100 IPS)
Standard Tape Block Size                                     : 8K
Large (Gapless) Tape Block Size                              : 16M
Run Only High Density Gapless Subtests (520,521)             : y
Select Printer On/Off Mode                                   : 0x0003
          -> Enable printer ON string before error <-
          -> Enable printer OFF string after error <-
Printer On Character Sequence (before error)                 : \033[?5i
Printer Off Character Sequence (after error)                 : \033[?4i
Use Standard Debug Setup                                     : n
User Debug Option Mask                                       : 0x1fff
          -> Disable Asynchronous Command Mode <-
          -> Disable Multi-Record Buffering <-
          -> Disable Reset of Controller After ^C <-
          -> Disable Hard Error Trapping (logging) <-
          -> Enter Debugger After an Error Occurs <-
          -> Disable copying of errors to error file <-
          -> Pause AFTER each CCU driver command is returned <-
          -> Show each CCU driver message AFTER it is returned <-
          -> Pause BEFORE each CCU driver message is sent <-
          -> Show each CCU driver message BEFORE it is sent <-
          -> Automatically enter debugger during a pause <-
Select Controller Register Dump Mode                         : 3
          -> Never <-
Select Media Error Logging Method                            : 0x001f
          -> After Test Completion <-
          -> After Each Subtest <-
          -> Upon Occurrence (Real Time) With Limited Register Dump <-
          -> Upon Occurrence (Real Time) With Full Register Dump <-
          -> Activate Printer During Media Errors (See Printer Options) <-
Maximum Media Error Retry Count                              : 10
Maximum # of Non-Fatal Media Errors Allowed                  : 40
Retry Formatter Correctable Media Errors On Writes           : y
Pattern for Write/Read Subtests                              : 0x6db6
Select Data Miscompare Display Mode                          : 2
          -> Mix Expected and Actual Values (xx/xx). <-
Enter Data Miscompare Line Dump Count                        : 4
Enter OK, or :NN to return to question NN                    : OK
```

Initialization Sequence for *dev5210*

After the last prompt is entered, and before subtest code execution, the diagnostic performs the following:

- Determines if the test was invoked for quick startup (i.e., dev5210x or dev5210 -q). If so, the diagnostic reads the test parameters from the specified parameter file (default parameter file is */tmp/dev5210.tmp*). If not, the user is presented with the parameter menu. After parameter input, the responses are written to the parameter file (default or user-specified).
- Locates the largest contiguous main memory space after the first 2 Mbytes and reserves it for use by *dev5210*.
- Determines if the CCU is already loaded with the *dev5210* CCU driver. If the driver is not loaded, the CCU is reloaded. After the load completes, the driver is configured for EGOS and then the EGOS probe message starts the driver.
- Passes the current test parameters (i.e., retry information) to the CCU driver.

> **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.

Prompt Explanations

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

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

Enter the name of an alternate */ioconfig* file if an alternate is desired. 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      0    MTC-201  0x0600  1    1   MTD-204
2) viop  3      0    MTC-201  0x3fc0  4    0   MTD-201
     *** Enter 0 for manual configuration ***
2: Select Configuration File Entry to Test [0,1-2,?]      (1) ->
```

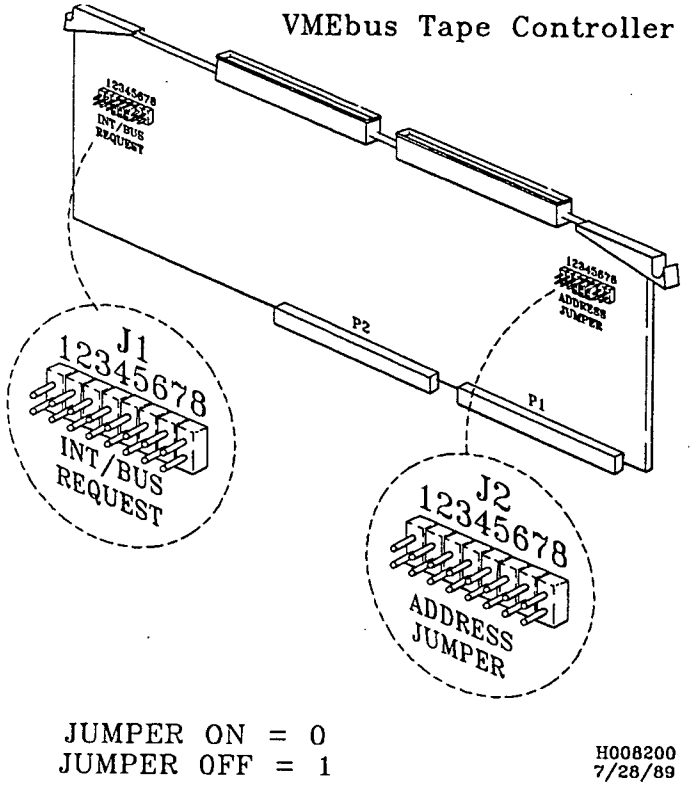
Enter the */ioconfig* file entry you want to test (if any). To select a predefined controller configuration entry from the */ioconfig* file, enter the number that is found to the left of the desired entry. Entering a **0** allows each item within the controller configuration entry to be independently specified. If most (but not all) of the items associated with a given entry in

/ioconfig are desired, select the number for that entry, backup to this prompt via the caret command (SHIFT 6), and then specify **0** the second time. This action causes the default values for the independent items to be the same as the originally-selected entry.

```
3: Display VBTC Configuration Help [y,n,?]                    (y) ->
```

This option displays the VME tape controller information. This information is useful when installing the jumpers that determine the controller's base address and interrupt level. The VBTC has two jumper blocks which are used to set the board's base address, interrupt level, and VMEbus request level. All logic is inverted, i.e., a removed jumper sets that bit. The jumper blocks are shown in the following figure:

Figure dev5210–5, VMEbus Tape Controller Jumper Blocks



The jumper blocks are defined in the following tables:

> **NOTE**
>
> From left to right, the jumper block pin numbers on the VBTC are numbered 1 through 8. To make it easier to construct binary patterns, they are listed from 8 through 1 in the following tables, with 8 being the most significant bit and 1 being the least significant bit. Also, the information listed in the following three tables is displayed on the screen by entering **y** or **?** as response to the prompt.

Table dev5210–5, VBTC Jumper Block Pin Settings

Jumper Block 1—J1 (Upper left side of board)								
Jumper	8	7	6	5	4	3	2	1
Bit	INT2	INT1	INT0	XX	XX	XX	BR1	BR0

Table dev5210–6, VBTC Jumper Block Pin Settings

Jumper Block 2—J2 (Upper right side of board)								
Jumper	8	7	6	5	4	3	2	1
Address	A13	A12	A11	A10	A9	A8	A7	A6

Table dev5210–7, Standard Address and Interrupt Levels

VBTC #	Address	Interrupt Level	Bus Request
1	0x1000	7	3
2	0x1040	site dependent	3
3	0x1080	site dependent	3
4	0x10c0	site dependent	3

```
4: VME IOP Number [3-7,?]                                    (3) ->
```

Enter the CCU slot number for the VMEbus IOP containing the VMEbus subsystem connected to the tape controller/drive. The CONVEX VMEbus tape controller part number is 410-001152-600.

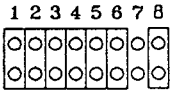
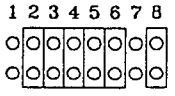
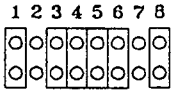
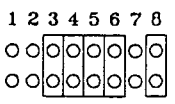
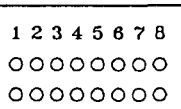
```
5: VME Chassis Number [0-1,?]                                    (0) ->
```

Enter the VMEbus chassis number for the chassis where the tape controller is installed. If there is only one chassis, the number is (in most cases) 0. The CONVEX VMEbus tape controller part number is 410-001152-600.

```
6: VMEbus Base Address of the Controller [0x0-0xffff,?]
                                                          (0x600) ->
```

Enter the low-order four hexadecimal digits of the VMEbus tape controller board's base address within the VMEbus standard address space. This address is controlled by jumper block 2 (**J2**) on the controller. The typical address is 0x1000. When all the straps are removed, the controller defaults to address 0x3fc0.

Table dev5210–8, VMEbus Address Level Jumper Settings (J2)

Address	Jumper Block 2 Settings
0x1000	
0x1040	
0x1080	
0x10c0	
0x3fc0	


```
7: VMEbus Interrupt Level of the Controller [1-7,?]          (1) ->
```

Enter the CONVEX VMEbus interrupt number assigned to the VMEbus tape controller board under test. This value is between 1 and 7, inclusive. The interrupt level is selected by jumper block 1 (**J1**) on the controller. The typical level is 7. When all the straps are removed, the controller defaults to level 7.

Table dev5210–9, VMEbus Interrupt and Bus Request Jumper Settings (J1)

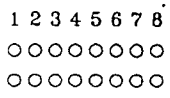
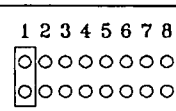
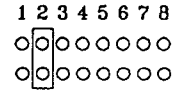
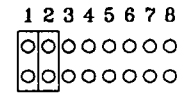
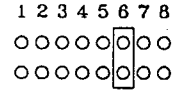
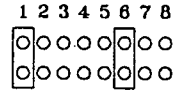
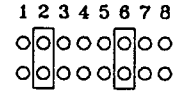
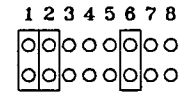
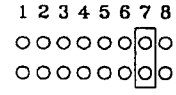
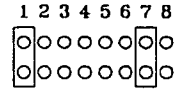
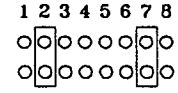
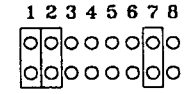
Interrupt Level	Bus Request	Jumper Block 1 Settings
7	3	
7	2	
7	1	
7	0	
6	3	
6	2	
6	1	
6	0	
5	3	
5	2	
5	1	
5	0	

Table dev5210–9, VMEbus Interrupt and Bus Request Jumper Settings (J1) (continued)

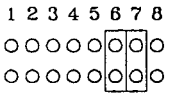
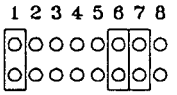
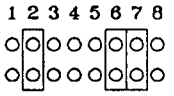
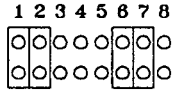
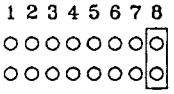
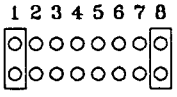
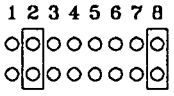
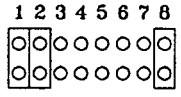
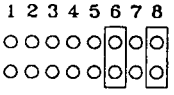
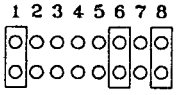
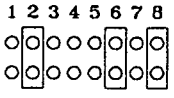
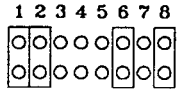
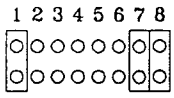
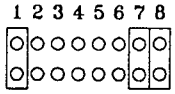
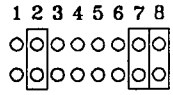
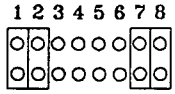
Interrupt Level	Bus Request	Jumper Block 1 Settings
4	3	
4	2	
4	1	
4	0	
3	3	
3	2	
3	1	
3	0	
2	3	
2	2	
2	1	
2	0	

Table dev5210–9, VMEbus Interrupt and Bus Request Jumper Settings (J1) (continued)

Interrupt Level	Bus Request	Jumper Block 1 Settings
1	3	
1	2	
1	1	
1	0	

```
8: Tape Unit Number [0-7,?]                                    (1) ->
```

The STC tape drive interface is a selector type interface (i.e., daisy chained). This entry specifies the tape unit select of the drive to test.

```
    ** Tape Drive models **
1:  Storage Technology Corp. 2921 ..................... MTD-201
2:  Storage Technology Corp. 2922 ..................... MTD-204
3:  Storage Technology Corp. 1963/1968 ................ MTD-202
4:  Fujitsu 2436-L1 ................................... MTD-203

9: Tape Drive Type [1-4,?]                                     (2) ->
```

Enter the menu number for the drive type to be tested. Menu entries are listed in the order they appear in the */mnt/bin/lib/DBtapefmt* file.

```
10: Use Standard Defaults for Other Options [y,n,?]            (y) ->
```

Enter **yes** to use the default values for all remaining prompts. This action eliminates the need to explicitly enter a RETURN in response to every remaining prompt.

```
   ** Tape Spool Size in Feet **
1:  600 ft.
2: 1200 ft.
3: 2400 ft.
4: 3600 ft.
11: Select Mounted Tape Spool Size [1-4,?]                (1) ->
```

Enter the menu number for the tape spool size loaded on the tape drive under test. This option allows limiting the amount of data written by any subtest, if applicable, to the amount that will fit on the tape. Four tape spool sizes are supported (600 ft., 1200 ft., 2400 ft., and 3600 ft.). Currently, this option only applies to the chain mode write/read subtests in the following list:

- 320—Chain mode write (NRZI)
- 321—Chain mode read (NRZI)
- 420—Chain mode write (PE)
- 421—Chain mode read (PE)
- 520—Chain mode write (GCR)
- 521—Chain mode read (GCR)

```
12: Run EOT Sensor Subtest #701 [y,n,?]                   (y) ->
```

Enter **y** if subtest 701, the EOT Sensor test, is to be executed. If a large tape spool is mounted, the EOT subtest can take more than five minutes to execute on some of the larger tape spool sizes.

```
   ** Available Data Transfer VMEbus Address Modifiers **
1:  Standard Supervisory Access (16 bit) (0x3d)
2:  Standard Non-Supervisory Access (16 bit) (0x39)
13: Data Transfer VMEbus Address Modifier [1-2,?]         (1) ->
```

Enter the menu number for the desired VMEbus address modifier to use with all VMEbus data transfers from or to the tape drive. The CONVEX VMEbus subsystem answers to all VMEbus address modifiers. Therefore, this option is likely to have no affect on transfers.

```
 ** Available Tape Velocities (IPS) **
1:   50
2:  100
14: Standard Tape Velocity [1-2,?]                        (2) ->
```

Enter the menu number for the default tape velocity (tape speed in inches per second) to use for all subtests that are not dependent on tape velocity.

```
15: Standard Tape Block Size [16-16M,?]                 (8K) ->
```

Enter the tape block size to use for all subtests that do not use predefined block sizes (90% of the tests in Classes 3, 4, and 5). A size greater than 128 Kbytes results in all standard transfers being performed in data chain mode with a chain interval size of 64 Kbytes. Increasing the size can greatly increase the amount of time required for test execution. An even binary-size multiple may be specified by appending the letter "K" for kilobytes or "M" for megabytes.

```
16: Large (Gapless) Tape Block Size [256K-16M,?]          (1M) ->
```

Enter the tape block size to use for the chain mode write and read subtests. The minimum size is 256 Kbytes. ConvexOS can write tape blocks as long as 16 Mbytes. The maximum size is limited by the amount of contiguous physical memory present in the target system. An even binary-size multiple may be specified by appending the letter "K" for kilobytes or "M" for megabytes.

NOTE

With the default retry limit of 10, the test will erase up to 17.5 feet of tape during an attempt to recover from a media error (1.75 feet per retry). The maximum STC legal gap size is 20 feet. Gaps longer than 20 feet may cause the formatter to timeout searching for a tape block. Therefore, if the specified block size results in a tape block longer that 17.5 feet, a simple media error beyond the first 17.5 feet results in a failure. The following table illustrates how tape block sizes in bytes relate to sizes in feet.

Table dev5210–10, Tape Block Size Relationships

Block Size		Block Length (feet)		
Kbytes	**Mbytes**	**GCR**	**PE**	**NRZI**
16384	16.0	223.6	873.8	1747.6
1024	1.0	13.9	54.6	109.2
512	0.5	7.0	27.3	54.6

```
17: Run Only High Density Gapless Subtests (520,521) [y,n,?]
                                                          (y) ->
```

Enter **y** to inhibit the execution of the NRZI and PE density Data Chain Mode subtests (i.e., NRZI: 320,321; PE: 420,421). This response is recommended. A "n" response requires that the tape drive and media be in superb condition because Data Chain Mode is used to write very large tape blocks, and larger blocks are more prone to media errors.

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

18: Select Printer On/Off Mode [0x0-0x3,?]                (0x0) ->
```

This prompt allows a specified character string to be sent to the printer before and/or after an error (and its data) is displayed. Although any string up to 64 characters in length may be specified, the intended use is to selectively turn a printer on before an error is displayed and to turn a printer off after an error has been displayed. This option is a paper-saving feature when executing *dev5210* multiple times or for long periods of time. This assumes a printer is connected to the auxiliary port on the display terminal where *dev5210* is executing and that the auxiliary port can be turned on and off via an escape character sequence.

To select both options, enter the hexadecimal mask obtained by ORing the two options together (0x03).

```
19: Printer On Character Sequence (before error)
    [<printer on string>,?]                          (\033[?5i) ->
```

This option allows the specification of the enable printer port escape character sequence. The default value is the "Enter Auto Print Mode" sequence for terminals that support VT100 terminal protocol. Control characters may be specified by using standard escape sequences used in C programming, for example:

\033	Octal value of the ASCII escape character (0x1b)
\x1b	Same as above value but specified in hex
\r	Carriage return
\n	Line feed
\t	Tab character
\b	Backspace character
\f	Form feed character

```
20: Printer Off Character Sequence (after error)
    [<printer off string>,?]                         (\033[?4i) ->
```

This option allows the specification of the disable printer port escape character sequence. The default value is the "Exit Auto Print Mode" sequence for terminals that support VT100 terminal protocol. 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 value but specified in hex
\r	Carriage return
\n	Line feed
\t	Tab character
\b	Backspace character
\f	Form feed character

```
21: Use Standard Debug Setup [y,n,?]                          (y) ->
```

Enter **y** to use the default values for all prompts related to debug mode. This action eliminates the need to explicitly enter a RETURN in response to every remaining prompt.

```
** USER Debug Options (bit mapped) **
  0x0001: Disable Asynchronous Command Mode
  0x0002: Disable Multi-Record Buffering
  0x0004: Disable Reset of Controller After ^C
  0x0008: Disable Hard Error Trapping (logging)
  0x0010: Enter Debugger After an Error Occurs
  0x0020: Disable copying of errors to error file
  0x0100: Pause AFTER each CCU driver command is returned
  0x0200: Show each CCU driver message AFTER it is returned
  0x0400: Pause BEFORE each CCU driver message is sent
  0x0800: Show each CCU driver message BEFORE it is sent
  0x1000: Automatically enter debugger during a pause

22: User Debug Option Mask [0x0-0xfff,?]                    (0x0) ->
```

Enter the number that indicates which debug option you want to enable. These options allow the standard test behavior to be altered in ways that are often useful for troubleshooting. In addition, the test can be paused before or after each command (i.e., typically one controller operation) is sent to the CCU driver. If you want multiple options, enter the hexadecimal mask obtained by ORing the desired options together. For example, if the first two options are desired, enter **0x3**. If all options are desired, enter **0x1fff**. Each option is described below in more detail.

`0x0001: Disable Asynchronous Command Mode`

This option inhibits the asynchronous command mode used in most of the write/read subtests. Asynchronous command mode occurs when multiple CCU driver commands are transmitted without waiting for a response to the commands. This mode allows the test to meet the tape drive reinstruct time and keep the tape spinning at maximum velocity by eliminating the 2 ms to 4 ms overhead required to transmit a command and receive its response via MBS. Unfortunately, the asynchronous command mode can be confusing when using the CCU driver single-step mode; therefore, selecting this option disables the asynchronous command mode.

`0x0002: Disable Multi-Record Buffering`

This option inhibits the multirecord buffering that takes place in 75% of the write/read subtests. Multirecord buffering causes the test to write or read as much data as the main memory buffer can hold at once. This buffering allows the test to keep the tape spinning at maximum velocity as long as possible (i.e., the number of tape start/stop cycles is reduced). Unfortunately, this buffering can be confusing when using the register dump on error mode. For example, if 4 blocks are read into the buffer and a data compare error occurs on the first and if the register dump on error mode was set, the register dump would be for the fourth block read, not the first. Therefore, this option forces one read at a time each followed by a data compare.

> **NOTE**
>
> The previous option does not work for all subtests that do multirecord buffering.

`0x0004: Disable Reset of Controller After ^C`

Normally, the controller is reset whenever the test is aborted by entering CTRL C. This action may destroy valuable register states in some cases. This option prevents the reset from occurring.

`0x0008: Disable Hard Error Trapping (logging)`

By default, this subtest monitors for hard system errors and aborts if a hard error occurs. Because hard errors are processed as an interrupt to the normal test logic, the state of the test

cannot be reported when a hard error occurs. Since hard errors always result in the immediate halt of the main memory system, a subtest may eventually fail without hard error logging enabled. This is often useful since the subtest state is reported (i.e., more fail data is reported to allow more insight into the possible source of the problem).

```
0x0010: Enter Debugger After an Error Occurs
```

This option specifies automatic entry into the interactive debugger any time an error is detected within a subtest. The debugger may also be invoked by specifying the *-d* option at test invocation time (i.e., **dev5210**[x] **-d**).

```
0x0020: Disable copying of errors to error file
```

This option disables the automatic copying of error reports to the file */tmp/dev5210.err*. This option might be used if 1000 lines of miscompared data were to be displayed; in that case, the file system might fill up or it could require a large amount of time to write the data to the file.

```
0x0100: Pause AFTER each CCU driver command is returned
0x0200: Show each CCU driver message AFTER it is returned
0x0400: Pause BEFORE each CCU driver message is sent
0x0800: Show each CCU driver message BEFORE it is sent
0x1000: Automatically enter debugger during a pause
```

These bit options allow single stepping before and/or after each CCU driver command. In addition, the internal command can be displayed by selecting one of the "show command" option bits. This is a valuable debugging tool when used with the register dump after each command mode. Also, the main memory read/write buffer and CCU memory can be examined by entering the debugger. See the "Interactive Debugger" section at the end of this test description for more details on the debugger.

For most cases, the "after" modes are sufficient. If the bit for the option "Automatically enter debugger during a pause" is not set, the test will pause, at which time the user has the following options:

Space bar	Continue
Return	Continue
:	Enter debugger

```
 ** VBTC register dump modes **
1: After Error
2: After Each Command Completion
3: Never

23: Select Controller Register Dump Mode [1-3,?]          (3) ->
```

Enter the menu number that indicates when to display the last known register state of the VBTC. The "After Each Command Completion" mode only works if the "CCU driver single step mode" option (show each CCU driver message after it is returned) is selected in the next option.

```
  ** Tape Media Error Logging Methods (bit mapped) **
0x0001: After Test Completion
0x0002: After Each Subtest
0x0004: Upon Occurrence (Real time) With Limited Register Dump
0x0008: Upon Occurrence (Real Time) With Full Register Dump
0x0010: Activate Printer During Media Errors (See Printer Options)

24: Select Media Error Logging Method [0x0-0x1f,?]        (0x3) ->
```

Enter the number that indicates when tape media errors are logged to the display. If you want multiple options, enter the hexadecimal mask obtained by ORing the desired options together. For example, if the first two options are desired, enter **0x3**. If all options are desired, enter **0x1f**.

Media errors are isolated into two classes:

- **1**—Formatter-correctable errors
- **2**—Uncorrectable errors

Formatter-correctable errors are errors that are mathematically correctable by the tape formatter. These errors can occur on both reads or writes. In the case of a write, a formatter-correctable error is detected by the read after write logic in the formatter. Uncorrectable errors cannot be corrected by the formatter and must be retried. Both of the error types can be logged as they occur, after each subtest, after the entire test, or after both. The individual options are selected by ORing the individual option bits together to build a mask of the desired options. Each option is described below in more detail:

```
0x0001: After Test Completion
```

This option prints a summary of all media errors that occurred during the entire test. This summary is printed after the last subtest is executed or at test termination when a fatal error has occurred. The format of the report is as follows:

```
Media Error Class ---- Total ---- NOT Recovered ---- Recovered ---- Ignored
Formatter-Corrected       13                  0              0           13
Uncorrectable              0                  0              0            0
```

Formatter-corrected errors during a read operation are ignored.

```
0x0002: After Each Subtest
```

This option specifies that a report such as the one in the previous example is to be printed after each subtest completes. The report is only printed if one or more media errors occur. In addition, only the applicable classes of media errors will be printed (i.e., if one formatter-correctable error occurred, the uncorrectable line would be omitted).

```
0x0004: Upon Occurrence (Real Time) With Limited Register Dump
```

This option prints a detailed media error report at the time a media error is detected. The report includes:

- **1**—Media error class
- **2**—Action taken (Recovered/Not recovered/Ignored)
- **3**—The number of retries required to recover the error (if any)
- **4**—The tape position (block and or file number if known)
- **5**—The pertinent controller registers that reflect the error

```
0x0008: Upon Occurrence (Real Time) With Full Register Dump
```

This option is identical to the previous "Real Time" option except that, with this option, all registers are printed after a media error occurs.

```
0x0010: Activate Printer During Media Errors (See Printer Options)
```

This option causes the Printer On/Off character sequences to be sent to the display before and after any media error reports displayed. This option allows the printer to be turned on and off automatically to include only the media error reports.

> **NOTE**
>
> In order for this option to work, the printer on/off character sequences must be enabled via the "Select Printer On/Off Mode" option (see option 18).

```
25: Maximum Media Error Retry Count [0-999,?]          (9) ->
```

Enter the maximum number of media-type error retries to be made after the original attempt when attempting to recover from a media error. A retry count of 9 results in 10 total maximum tries of a tape operation. A 0 value causes all media errors to be considered fatal.

```
26: Maximum # of Non-Fatal Media Errors Allowed [0-9999,?]
                                                       (40) ->
```

Enter the maximum number of non-fatal (i.e., recovered plus ignored) media errors allowed during one invocation of the test. An ignored media error could be a formatter-correctable error during a read operation. Recovered media errors are errors that are corrected by a retry.

The default is 20 errors per available recording density (e.g., NRZI, PE, GCR). For strenuous testing, set this value to 5 or less, which requires tape media that is in superb condition.

```
27: Retry Formatter Correctable Media Errors On Writes [y,n,?]
                                                       (y) ->
```

Enter **y** to retry tape media errors that the tape formatter can correct. This option only applies to tape writes; a formatter-correctable error during a read is not retried. An example of a formatter-correctable error during a write would be a single dead track while in GCR mode. The formatter can mathematically reconstruct a single dead track in GCR.

```
28: Pattern for Write/Read Subtests [<hexadecimal pattern>,?]
                                                  (0x6db6) ->
```

Enter the hexadecimal data pattern to use for all tape write/read subtests. The pattern may be any length from 1 nibble to 256 bytes. The pattern is replicated as necessary when formatting buffers for tape writes.

The pattern can be specified via a file on the SPU disk. For this method enter a "+" followed by the optional file name of the pattern file. If the file name is omitted, the test defaults to the file *dev5210.pat*. The pattern may contain white space and may be spread across multiple lines.

NOTE

The buffer fill pattern preceding a read is "0x55aa55aaaa55aa55." Therefore, to ensure accurate test results, this pattern should be avoided as the write/read data pattern.

```
  ** Available Data Miscompare Dump Modes **
1: Separate Expected and Actual Values into Two Blocks.
2: Mix Expected and Actual Values (xx/xx).
3: Only Show Bytes that Miscompare.

29: Select Data Miscompare Display Mode [1-3,?]          (2) ->
```

Enter the menu number for the desired format of data miscompare dumps. The "separate block" mode requires the most display space while "mixed" mode uses the least amount of space. The "show miscompares only" gives greater depth into the data (assuming miscomparisons occur at sparse intervals). A sample of each format's output follows:

```
------------ Separate Expected and Actual Values into Two Blocks ------------
Expected:
    00100000:  6d* b6  6d  b6* 6d  b6  6d  b6* 6d  b6  6d  b6  6d  b6  6d* b6
    00100010:  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6
Actual:
    00100000:  00* b6  6d  77* 6d  b6  6d  88* 6d  b6  6d  b6  6d  b6  99* b6
    00100010:  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6  6d  b6


------------------- Mix Expected and Actual Values (xx/xx) ------------------
Expected/Actual:
    00100000:  6d/00* b6/b6  6d/6d  b6/77* 6d/6d  b6/b6  6d/6d  b6/88*
    00100008:  6d/6d  b6/b6  6d/6d  b6/b6  6d/6d  b6/b6  6d/99* b6/b6


--------------------- Only Show Bytes that Miscompare ----------------------
Expected/Actual (Miscompares only):
    00100000: 6d/00*  00100003: b6/77*  00100007: b6/88*  0010001e: b6/88*
```

```
30: Enter Data Miscompare Line Dump Count [1-1000,?]      (4) ->
```

Enter the maximum number of display lines to use when dumping data after a miscompare.

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

Enter **OK** or (RETURN) to terminate the test parameter menu; inputs are no longer changeable. Otherwise, enter **:NN** to return to the question indicated by *NN*.

Class Descriptions

The *dev5210* test contains the seven classes of subtests listed in the following table:

Table dev5210–11, *dev5210* Test Classes

Class	Description
1	VBTC loopback tests
2	Formatter/drive tests
3	Write/read tests—800 bpi (NRZI)
4	Write/read tests—1,600 bpi (PE)
5	Write/read tests—6,250 bpi (GCR)
6	Exception tests
7	Miscellaneous tests

NOTE

In the previous table, Class 3, 4, and 5 all perform the same subtests but at different recording densities and parameters.

All *dev5210* subtests are loopable under the *dshell*.

Class 1 Subtests

Class 1 subtests only test the VBTC via the VBTC Diagnostic Loopback Mode; no tape drive is required. Class 1 subtests verify the following functionalities:

- Ability to communicate with the VMEbus Input/Output Processor (VIOP), VMEbus Control Unit (VBCU), and the VMEbus Tape Controller (VBTC)
- Ability of the VBTC to interrupt and to mask interrupts
- Accessibility to main memory
- Detection of FIFO parity errors

Most Class 1 subtests use the VBTC Diagnostic Loopback Mode. Diagnostic Loopback Mode is enabled by setting bit <6> in the control register at address 0x*n*01. When the Diagnostic Loopback Mode is enabled, it is verified (upon command termination) by reading the status register at address 0x*n*10. The VBTC is reset at the start of each subtest in this class (except Subtest 101) by writing to the VBTC register at address 0x*n*0F.

The following table lists all Class 1 subtests, their descriptions, and the approximate times required to execute each subtest:

Table dev5210–12, Class 1 Subtests

Subtest	Description	Time (min:sec)
101	VBTC Pending Register RAM Pattern Test	00:07
102	VBTC Reset Test	00:00
103	VBTC Interrupt Loopback Test	00:02
104	VBTC FIFO Pattern Test	00:14
105	VBTC FIFO Variable Size Write/Read Test	01:46

The times listed are approximate. The time required to execute each subtest depends on the amount of memory, size of the tape, density, and the velocity of the tape drive.

Subtest 101, VBTC Pending Register RAM Pattern Test

Subtest 101 verifies the integrity of the VBTC pending level registers via pattern tests. Registers from 0x01 through 0x0E are pattern-tested using the patterns in the following table:

Table dev5210–13, Subtest 101 Test Patterns

Description	Hexadecimal Pattern	Rotations (iterations)
Eight zeros and eight ones	0x00ff	2
Walking ones	0x0102040810204080	8
Walking zeros	0xfefdfbf7efdfbf7f	8
Alternating zeros and ones	0xaa55	2
Two ones and a zero	0xdb6db6	3
Two zeros and a one	0x249249	3
Two zeros and two ones	0xcc993366	4

Each byte of each pattern is written to and read from each RAM location. For example, the eight zeros and eight ones pattern (0x00ff) is written to and read from each RAM location once. Then it is rotated (0xff00), rewritten, and read. Each rotation is written, read, and compared twice.

The subtest does not write the start flag or the reset flag. It requires that the VBTC decode addresses on the bus.

Subtest 102, VBTC Reset Test

Subtest 102 verifies that multiple VBTC resets result in the same controller state. This subtest is the first to actually execute a command. This subtest first resets the VBTC, then executes a *NOP* command in the Diagnostic Loopback Mode. After the *NOP* command completes (i.e., the

controller interrupts), the register state of the controller is saved. The *NOP* command is supervised by a 1-second timeout. The above sequence is then repeated, and the ending register state is partially compared with the original register state. Differences are reported if they occur.

Subtest 103, VBTC Interrupt Loopback Test

Subtest 103 verifies that the VBTC command completion interrupts work. This subtest first resets the VBTC, then verifies that the VBTC can generate a 68020 interrupt on the VIOP. The interrupt capability is tested at the selected VBCU interrupt level and at 68020 interrupt level 3. The subtest also verifies that the interrupt enable bit works by de-asserting it and expecting the driver to receive a timeout error while executing a command. A *NOP* command, in Diagnostic Loopback Mode, is used for all commands.

Subtest 104, VBTC FIFO Pattern Test

Subtest 104 verifies the VBTC FIFO via pattern tests in the Diagnostic Loopback Mode. Initially, the subtest resets the VBTC. Since the VBTC FIFO cannot be filled (i.e., capacity is 1,024 bytes) in the Diagnostic Loopback Mode, the subtest writes and reads half of the FIFO (512 bytes) with each pattern rotation twice. Due to the way the FIFO works, this tests all FIFO RAM locations. The patterns in the following table are used:

Table dev5210–14, Subtest 104 Test Patterns

Description	Hexadecimal Pattern	Rotations (iterations)
Eight zeros and eight ones	0x00ff	2
Walking ones	0x0102040810204080	8
Walking zeros	0xfefdfbf7efdfbf7f	8
Alternating zeros and ones	0xaa55	2
Two ones and a zero	0xdb6db6	3
Two zeros and a one	0x249249	3
Two zeros and two ones	0xcc993366	4

Each byte of each pattern is written to and read from each RAM location. For example, the eight zeros and eight ones pattern (0x00ff) is written to and read from each RAM location once. Then it is rotated (0xff00), rewritten, and read. Each rotation is written, read, and compared twice.

Subtest 105, VBTC FIFO Variable Size Write/Read Test

Subtest 105 verifies the VBTC FIFO via variable-length reads and writes. The subtest first resets the VBTC, then writes and reads varying block sizes in the Diagnostic Loopback Mode from both even and odd addresses. During reads, the controller transfer count is set to 512 bytes more than

the current size of the written data. At the end of the read, the VBTC is expected to have a residual transfer count of 512. In addition, data compares occur on the read data and the following 512 bytes. The 512-byte overflow buffer beyond the read data is expected to contain the initial buffer fill pattern. The subtest tries all transfer sizes from one to 957 bytes (even and odd addresses).

Class 2 Subtests

Class 2 contains subtests not specific to tape recording density. No data is actually written to the tape (i.e., the digital loopback write command *LWR* is used).

The following table lists all Class 2 subtests, their descriptions, and the times required to execute each subtest:

Table dev5210–15, Class 2 Subtests

Subtest	Description	Time (min:sec)
200	Formatter Accessibility Test	00:00
201	Formatter/VBTC Interface Test	00:02
203	Formatter Looped Write to Read Test	00:03
204	VBTC Piped Write Test	00:04

The times listed are approximate. The time required to execute each subtest depends on the amount of memory, size of the tape, density, and the velocity of the tape drive.

Subtest 200, Formatter Accessibility Test

Subtest 200 verifies that the formatter is accessible and can execute a command. This subtest uses a *NOP* command and attempts to execute the simplest formatter command available to verify interconnect to the tape drive/formatter. According to the *STC 2920 Series Maintenance Manual* (StorageTek Interface Specification), a *NOP* results only in the formatter asserted and then deasserting BUSY. No status lines should be changed.

Subtest 201, Formatter/VBTC Interface Test

Subtest 201 verifies the general interface to the formatter by using the *DMS, NOP* command sequence available on the STC 2920 Series drives. It verifies that many of the discrete status lines from the drive are physically connected and readable by the VBTC. Refer to the *STC 2920 Series Maintenance Manual* (Chapter 4) for more information.

> **NOTE**
>
> Subtest 201 is only supported on the STC 2920 Series drives.

The following table lists the status lines that are tested:

Table dev5210–16, Tested STC Status Lines

TESTED STC INTERFACE SIGNALS				
NOP #	**Description**	**Mnemonic**	**Connector Pin Number**	**Software Detectable**
1	Online status	ONLS	J6-49	Yes
2	Identification burst status	ID BRST	J6-25	No
3	File protect status	FPTS	J7-27	Yes
4	Rewinding status	REWS	J7-59	Yes
5	Expecting data status	RECV	J6-23	No
6	Operation incomplete status	OP INC	J6-27	Yes
7	Online status	ONLS	J6-49	Yes
8	Tape mark status	TMS	J6-31	Yes
9	Command reject	REJ	J6-33	Yes
10	Overrun status	OVRNS	J6-35	Yes
11	Data check status	DATA CHK	J6-37	Yes
12	EPROM error status	ROMPS	J6-39	Yes
13	Corrected error status	CRERR	J6-41	Yes
14	Online status	ONLS	J6-49	Yes
15	High density status	HDNS	J6-45	No
16	Ready status	RDYS	J6-53	Yes
17	Write status	WRTS	J6-55	Yes
18–53	MUX bytes (0–3) all 9 bits	ERRMX <0..7,P>	J6-<1,3,5,..17>	Yes

Subtest 203, Formatter Looped Write to Read Test

Subtest 203 verifies the data bus interface to the formatter. This subtest first rewinds the tape to BOT, then writes ten 16-Kbyte blocks via the *LWR* command. This subtest verifies the digital data path between the drive analog circuitry and VBTC.

Subtest 204, VBTC Piped Write Test

Subtest 204 verifies that the VBTC can operate in a pipelined mode. This subtest rewinds the tape to BOT, then uses the *LWR* command to verify that the VBTC can pipeline two commands. Twenty *LWR* commands are pipelined with a transfer size of 16 Kbytes.

Class 3, 4, and 5 Subtests

Class 3, 4, and 5 subtests verify that the drive can write and read data to its tape. All tape motion commands are verified. Where applicable, Class 3, 4, and 5 subtests verify the correct state of the drive prior to the actual test logic being executed. A *NOP* command is issued at the start of each test to select the correct unit. In the event the drive is in an incorrect state, the subtest pauses and the operator is requested to intervene and place the drive online.

Each subtest pauses for up to 30 minutes while waiting for the drive to come online and ready. Once the drive is online and ready, the *CLR* command is issued, followed by a check of the formatter status registers for errors. Most subtests rewind the tape at this point.

> **NOTE**
>
> This diagnostic allows a maximum of 5 minutes for the rewind to complete.

Read and write subtests take advantage of both current tape position and controller state. This implies the position of the tape and the controller status is known at most times. Initially, if the tape position is not known, the tape is rewound, and the VBTC is reset, if necessary. Written tape blocks contain address information for verification of positional coordinates.

The following table illustrates the layout of the tape header written at the start of each block:

Table dev5210–17, Tape Header Layout

Field Description	Number of Bytes
Current block number	2
Number of blocks in this file	2
Current file number	2
Total number of files present	2
Size of this block	4
Subtest that wrote this data	2
Checksum of header	2
Total	16

Class 3, 4, and 5 tests perform the same subtests but at different densities as follows:

- **3**—Write/read tests (800 bpi (NRZI))
- **4**—Write/read tests (1,600 bpi (PE))
- **5**—Write/read tests (6,250 bpi (GCR))

Table dev5210–18, "Class 3, 4, and 5 Subtests," lists all Class 3, 4, and 5 subtests, their descriptions, and the time required for each to execute. The times listed are approximate. The time required to execute each subtest depends on the amount of memory, size of the tape, density, and the velocity of the tape drive.

These subtests are numbered from 300–599 (n00–n99). Subtest numbers in these three classes contain a prefix of *n* (as the hundreds digit) that is used to select tape density. The 300 series subtests are for 800 bpi (NRZI) tapes, the 400 series for 1,600 bpi (PE) tapes, and the 500 series for 6,250 bpi (GCR) tapes.

Table dev5210–18, Class 3, 4, and 5 Subtests

Subtest	Description	Time (min:sec)
*n*00	ID Burst Write Test	00:02
*n*01	ID Burst Read Test	00:03
*n*02	Write Block Test	00:17
*n*03	Read Block Fwd/Bwd Test	01:06
*n*04	Skip Block Fwd/Bwd Test	01:20
*n*05	Erase Gap Test	00:17
*n*06	Write Tape Mark Test	00:12
*n*07	Skip Tape Mark Test	00:14
*n*08	Write Block/Tape Mark Test	00:25
*n*09	Read Block/Tape Mark Fwd/Bwd Test	01:22
*n*10	Skip Block/Tape Mark Fwd/Bwd Test	01:07
*n*11	Write File UNIX Style Test	00:38
*n*12	Read UNIX Style Written File Test	00:48
*n*13	Write Variable Size Blocks Test	00:22
*n*14	Read Variable Size Blocks Test	00:29
*n*20	Chain Mode Write Test	00:23
*n*21	Chain Mode Read Test	00:44
*n*50	Write/Read All Velocities Comb. Test	01:28

[1] *n* varies from 3 to 5: 3 is 800 bpi; 4 is 1,600 bpi; and 5 is 6,250 bpi. The diagnostic defaults to all densities specified in the */mnt/bin/lib/DBtapefmt* file.

Subtest *n*00, ID Burst Write Test

Subtest *n*00 verifies that the drive under test can write an ID burst. This subtest first rewinds the tape to BOT. It then writes a tape mark and verifies that no errors occur and that the correct recording density is selected. This subtest verifies that the ID burst was successfully written because any write from BOT causes the ID BURST, for the current recording density, to be written and verified (read after write).

Subtest *n*01, ID Burst Read Test

Subtest *n*01 verifies that the drive under test can read an ID burst. This subtest rewinds the tape to BOT and then issues a Read Forward (*RDF*) command, expecting to transfer zero bytes (i.e., a tape mark is detected). In addition, the test verifies that the density reflects the assumed ID burst that subtest *n*00 wrote.

Subtest *n*02, Write Block Test

Subtest *n*02 rewinds the tape, then writes 200 fixed-length tape blocks from BOT. The block size is the size specified in the Test Parameter Menu. No tape marks are written during the subtest.

Subtest *n*03, Read Block Fwd/Bwd Test

Subtest *n*03 verifies simple reads by reading the 200 blocks that Subtest *n*02 wrote. It rewinds the tape to BOT, issues 200 *RDF* commands, and issues 200 Read Backward (*RDB*) commands. Data compares occur at the end of each read pass.

Subtest *n*04, Skip Block Fwd/Bwd Test

Subtest *n*04 verifies that the Forward Skip Block (*FSB*) and the Backward Skip Block (*BSB*) commands can skip over tape blocks. This subtest skips over the data that Subtest *n*02 wrote. The subtest first rewinds the tape. After each seek from block to block (using the necessary count of either the *FSB* or *BSB* command), a Read Forward (*RDF*) command is issued, and the tape header is checked to verify that a seek error (tape positioning error) has not occurred. The following list contains the order in which the subtest skips blocks and the number of the blocks that it skips:

- Skip forward (further each time) to the following blocks:
  - 2, 4, 8, 14, 22, 32, 64, 128, 200
- Skip backward (further each time) to the following blocks:
  - 198, 196, 192, 186, 178, 168, 136, 72, 1
- Skip forward then backward (sawtooth-like pattern) to the following blocks:
  - 90, 102, 92, 104, 94, 106, 96, 108, 98, 110, 100
- Skip backward then forward (sawtooth-like pattern) to the following blocks:
  - 110, 98, 108, 96, 106, 94, 104, 92, 102, 90, 100
- Skip backward and forward (accordion-like pattern) to the following blocks:
  - 130, 70, 120, 80, 110, 90, 106, 94, 104, 96, 102, 98

Subtest *n*05, Erase Gap Test

Subtest *n*05 verifies that the Erase Gap (*ERG*) command can erase the tape. This subtest begins by rewinding the tape. It then writes 10 blocks and verifies them by reading backwards all 10 blocks. It then positions the tape just beyond the first block with a Backward Space File (*BSF*) command. Next, it erases the second tape block and then rewinds the tape. Since the *ERG* command is supposed to erase approximately 3.5 inches of tape, the length of the blocks written is slightly more than this. The size is, therefore, based on the current recording density. Table dev5210–19, "Erase Gap Sizes," lists the different erase gap sizes.

Table dev5210–19, Erase Gap Sizes

Density	Block Sizes (inches)
800 bpi	5.12 inches (4 Kbytes)
1,600 bpi	5.12 inches (8 Kbytes)
6,250 bpi	5.24 inches (32 Kbytes)

Block 1 should not be affected by the erase. Block 2 should be unreadable after the erase. This subtest first reads and verifies Block 1. Next, it tries to read the erased block 2. It is possible to obtain two different outcomes. First, a tape read error may result (most likely). Second, the read may complete successfully if the third block is actually read instead of the second (rare possibility).

Subtest *n*06, Write Tape Mark Test

Subtest *n*06 verifies that multiple tape marks can be written. This subtest first rewinds the tape, then writes one block followed by 200 tape marks and a second block. Both blocks are the size of a tape block header since they are only used to verify tape position.

Subtest *n*07, Skip Tape Mark Test

Subtest *n*07 verifies that multiple tape marks (200 with no data between them) can be skipped over correctly and that the tape marks are written correctly. This subtest uses the data that *n*06 wrote. It first rewinds the tape to BOT, reads the first block, and executes 200 Forward Skip File (*FSF*) commands. Next, the subtest attempts to read the second block. Tape position is verified by checking the second block's header.

Subtest *n*08, Write Block/Tape Mark Test

Subtest *n*08 verifies that a combination of blocks and file marks can be written. This subtest writes 10 files of 25 blocks each, with a tape mark separating each file. An end-of-tape (EOT) mark is written after the last tape mark.

Subtest *n*09, Read Block/Tape Mark Fwd/Bwd Test

Subtest *n*09 verifies that the combination of blocks and file marks that subtest *n*08 wrote is readable. It uses the *RDF* and *RDB* commands to read all blocks and tape marks and perform a data compare. The data is first read forward and a data compare performed. Next, the data is read backward and another data compare is performed.

Subtest *n*10, Skip Block/Tape Mark Fwd/Bwd Test

Subtest *n*10 verifies that the Forward Skip Block (*FSB*), Backward Skip Block (*BSB*), Forward Skip File (*FSF*), and the Backward Skip File (*BSF*) commands execute properly. This subtest skips over the data that subtest *n*08 wrote. It first rewinds the tape. After each seek from point to point (using the appropriate combination of the Skip commands), a Read Forward (*RDF*) command is issued and the tape header checked to verify that a seek error (tape positioning error) has not occurred. The following list contains the file and block coordinates reached after each skip operation:

- Skip forward (further each time) to {file, block}
  - {2, 1}
  - {5, 1}
  - {10, 1}
- Skip backward (further each time) to {file, block}
  - {8, 24}
  - {4, 24}
  - {1, 24}
- Skip backward and forward (accordion-like pattern) to {file, block}
  - {2, 10}
  - {1, 2}
  - {9, 23}
  - {7, 24}
  - {4, 1}
  - {10, 25}

Subtest *n*11, Write File UNIX Style Test

Subtest *n*11 verifies that a file can be written as the ConvexOS driver writes files. This subtest writes several files using the same command sequence as the ConvexOS driver. The sequence for each file is:

1. *N* write-blocks
2. Write-tape mark
3. Write-tape mark
4. Backspace-file

Asynchronous I/O is used for the entire command stream to the CCU driver. The number of blocks per file and the file count varies with the recording density. Table dev5210–20 lists the block count and file count:

Table dev5210–20, Subtest *n*11 Write File/Block Count

Density	Block Count (*N*)	File Count
800 BPI	8	8
1,600 BPI	16	16
6,250 BPI	50	20

Subtest *n*12, Read UNIX Style Written File Test

Subtest *n*12 verifies that the file written by subtest *n*11 can be read. The subtest initially rewinds the tape, then reads all tape blocks and marks that subtest *n*11 wrote.

Subtest *n*13, Write Variable Size Blocks Test

Subtest *n*13 verifies that variable-length tape blocks can be written. This subtest writes varying block sizes from one byte to 258 bytes, in increments of one, and then -2 through +2 around every base 2 boundary up to 2^{16} (64 Kbytes). Tape headers are only written if the current block size is large enough to contain the entire header.

Subtest *n*14, Read Variable Size Blocks Test

Subtest *n*14 verifies that the variable length tape blocks can be read. This subtest reads varying block sizes of data that subtest *n*13 wrote from one byte to 258 bytes, in increments of one, and then -2 through +2 around every base 2 boundary up to 2^{16} (64 Kbytes). Tape headers are only read and verified if the current block size is large enough to contain the entire header.

Subtest *n*20, Chain Mode Write Test

Subtest *n*20 verifies that chain mode write operations execute properly. It initially rewinds the tape, then writes one long gapless tape block using the VBTC data chaining mode. The chain interval size is 32 Kbytes. Normally the CCU driver would default to 64-Kbyte intervals (128-Kbyte windows). The actual size of the block written by the test is controlled by the size of the "Large Tape Block Size" specified in the "Test Parameter Menu". The size is reduced, if necessary, to the maximum size that can be written in the current density on the currently mounted tape spool.

Subtest *n*21, Chain Mode Read Test

Subtest *n*21 verifies that chain mode reads execute properly. This subtest reads the data that subtest *n*20 wrote and performs a data compare. Refer to subtest *n*20 for a description of the subtest algorithm.

Subtest *n*50, Write/Read All Velocities Comb. Test

Subtest *n*50 writes 50 blocks delimited by two tape marks in each velocity and reads back each written group in every other available velocity.

Class 6 Subtests

Class 6 subtests verify that the VBTC and tape formatter can detect and recover from a variety of error conditions. The initial setup at the start of each test is the same as that described for the Class 3, 4, and 5 subtests (i.e., check for online, *CLR*, and rewind). The following table lists all Class 6 Subtests, their descriptions, and the times required to execute each subtest.

Table dev5210–21, Class 6 Subtests

Subtest	Description	Time (min:sec)
600	VBTC Read Underrun Test	00:08
601	VBTC/Drive Forced Parity Error Test	00:03
602	VBTC Chain Mode Missed Interrupt Test	00:10
603	VBTC Chain Mode Read Underrun Test	00:05
604	VBTC Pipe Mode Error Test	00:06
606	VBTC Bus Error Recovery Test	00:23
608	Chain Mode Interval Boundary Write Test	02:10
609	Chain Mode Interval Boundary Read Test	02:20
610	Chain Mode Error Test	00:10

The times listed are approximate. The time required to execute each subtest depends on the amount of memory, size of the tape, density, and the velocity of the tape drive.

Subtest 600, VBTC Read Underrun Test

Subtest 600 verifies that the VBTC detects a read underrun condition and correctly handshakes for and drops the remaining data. This subtest begins by writing three 16-Kbyte tape blocks from BOT. It then writes two delimiting tape marks and rewinds the tape. Each of the three blocks is read back with transfer counts less than their size, which should cause read underrun. At the end of each read, the subtest verifies that only a read underrun is reported by the VBTC (i.e., no formatter-reported errors). Although no data compare occurs, the subtest verifies the read tape block headers. After the read of the third block, the subtest attempts to read across the two trailing tape marks.

Subtest 601, VBTC/Drive Forced Parity Error Test

Subtest 601 verifies that parity errors are detectable. This subtest initially rewinds the tape to BOT, then writes four 4-Kbyte blocks using the *LWR* command with good parity. The subtest then repeats the write command sequence with bad parity forced by the VBTC. After each *LWR* command, the test checks for a drive-reported error (parity or data check). The expected error type is controlled by one of the personality bits in the drive database file (*/mnt/bin/lib/DBtapefmt*). Next, an optional drive reset may occur (based on another one of the personality bits in the drive database file) since some drives may not clear a parity error until after a reset. Finally, the subtest uses the *CLR* command to insure that the previous error state can be cleared.

Subtest 602, VBTC Chain Mode Missed Interrupt Test

Subtest 602 verifies that the VBTC halts when an interrupt receives late service in the data chain mode. This subtest first rewinds the tape to BOT. It then uses the *LWR* command to verify that the VBTC halts when command chaining is active and that a chain interval interrupt is not serviced before the next interval interrupt arrives. The test begins by starting a 128-Kbyte write in

data chain mode with an interval size of 16 Kbytes. It then pauses on the third interval for two seconds and expects the VBTC to halt with the appropriate error status. The process is repeated on the fourth, fifth, and sixth chain intervals.

Subtest 603, VBTC Chain Mode Read Underrun Test

Subtest 603 verifies that a read underrun is detected in the data chain mode under a variety of boundary conditions. This subtest rewinds the tape to BOT and then writes two 40,960-byte tape blocks (8,192 * 5) followed by two tape marks. It then rewinds the tape and issues a read with a size of 28,672, followed by a size of 36,864 and a chain interval size of 8192 or 8 Kbytes. Both reads should cause a read underrun to occur, and the VBTC should continue to read the data until the real end-of-block is detected. As a result, the only error that should be indicated in the terminating status is a read underrun. The above read sizes cause the read underrun to occur once on Buffer A and once on Buffer B. Although the data chain mode bit is already reset when read underrun occurs, this subtest still tries to assure it can be detected at the end of a data chain mode transfer (on both Buffer A and Buffer B).

Subtest 604, VBTC Pipe Mode Error Test

Subtest 604 verifies that the VBTC halts command pipelining when an error occurs during piped commands. This subtest rewinds the tape, then writes a 32-Kbyte block, backspaces the tape, and writes a 16-Kbyte block. This action guarantees that a read after the 16-Kbyte block is written generates an error since there is no evident division before the second block begins. Next, a *RDF, REW* command sequence is pipelined. The rewind should not execute because the *RDF* fails, and the VBTC should halt pipelining. The fact that the following *REW* command did not execute should be indicated by the Command Abort status bit being asserted by the VBTC. In addition, BOT should not be present after the failure (i.e., make sure the *REW* command was not executed).

Subtest 606, VBTC Bus Error Recover Test

Subtest 606 verifies that VMEbus errors are recoverable. This subtest first rewinds the tape to BOT. It then writes four 4,096-byte tape blocks with a VMEbus standard address of 0xc00000 (bits <22> and <23> set). This address is not mapped by the VBCU, and the read should result in a VMEbus bus error. This subtest verifies that the VBTC reports a bus error after each write.

Subtest 608, Chain Mode Interval Boundary Write Test

Subtest 608 tests VBTC data chain mode under various boundary conditions. This subtest initially rewinds the tape to BOT, then writes varying block sizes starting with 96,296 bytes (96 Kbytes - 8) up to 98312 bytes (96 Kbytes + 8). Two blocks of each size are written (one from an even address alignment and one from an odd address alignment). Tape headers are written in each block. All tape blocks are written in data chain mode with a chain interval size of 32,768 bytes (32 Kbytes). Next, the test verifies the previously written records by rewinding the tape to BOT, reading the blocks in normal mode (i.e., no data chain mode), and comparing the data after the read.

Subtest 609, Chain Mode Interval Boundary Read Test

Subtest 609 uses the tape block data that subtest 608 wrote. This subtest initially rewinds the tape to BOT, then attempts to to read each of the 34 records written by subtest 608. Two blocks of each size are read (one to an initial even address alignment and one to an odd address alignment). All tape blocks are read in data chain mode with a chain interval size of 32,768 bytes (32 Kbytes). An overflow area of 512 bytes is skipped after each read to insure the VBTC does not transfer more than the correct amount of data. After each read, the data, tape header, and the overflow buffers are compared against expected results. Next, the subtest rewinds the tape to BOT and reads all the record sizes again with the transfer count set to 192 Kbytes. This read terminates the controller while in chain mode. It expects the appropriate residual transfer count at the end of each. As with the previous group of reads, all data is again compared.

Subtest 610, Chain Mode Error Test

Subtest 610 verifies that latter versions of the VBTC do not require an STC *CLR* command when the previous command terminated with an error. Older versions of the VBTC require the drive status to be error-free before initiating a command that utilizes data chain mode. This subtest initially rewinds the tape, then writes a 32-Kbyte block, backspaces the tape, and writes a 16-Kbyte block. This action guarantees a read (after the 16-Kbyte block is written) generates an error since there is no evident division before the second block begins. Next, the subtest starts a 256-Kbyte data chain mode write, thus guaranteeing that a chain mode command can be executed when an error was present at the end of the previous command.

> **NOTE**
>
> This subtest will not execute on VBTCs that are earlier than Fab revision B, assembly revision K.

Class 7 Subtests

Class 7 consists of the tape velocity measurement subtest and the end of tape sensor subtest. Both of these subtests have initialization sequences similar to Classes 3-6. The following table lists all Class 7 Subtests, their descriptions, and the times required to execute each subtest.

Table dev5210–22, Class 7 Subtests

Subtest	Description	Time (min:sec)
700	Drive Tape Velocity Verification	00:39
701	Drive EOT Sensor Verification Test	01:20

The times listed are approximate. The time required to execute each subtest depends on the amount of memory, size of the tape, density, and the velocity of the tape drive.

Subtest 700, Drive Tape Velocity Verification

Subtest 700 verifies all available tape velocities on the current drive in all supported densities. This subtest verifies that the achieved tape velocity is within the specified tolerance (in the database file */mnt/bin/lib/DBtapefmt*). Current suggested tolerance is ± 5%. This subtest also verifies that each velocity can be selected via software control (whatever the method). This subtest starts with the lowest tape density and ends with the most dense recording format while verifying every available velocity. For each density and velocity, this subtest rewinds the tape, then writes enough data to keep the tape moving for a minimum of 4 seconds. This subtest determines the number of blocks needed to meet the 4-second time requirement. The following table lists the block sizes used for each density:

Table dev5210–23, Block Sizes Written

Density	Block Sizes
800 bpi	64 Kbytes
1,600 bpi	128 Kbytes
6,250 bpi	512 Kbytes

The time required to write the first block is ignored to get the tape up to speed and to write the ID burst. The time required to write the remaining blocks is measured, and the resulting tape velocity in Inches Per Second (IPS) is calculated and displayed in the following format:

```
Density ---- Velocity Nom/Meas ---- Tolerance ---- Delta
PE              50.0 /  49.8         +/-5.0%       -0.4%
PE             100.0 /  99.4         +/-5.0%       -0.6%
GCR             50.0 /  50.0         +/-5.0%        0.0%
GCR            100.0 / 100.1         +/-5.0%       +0.1%
```

Subtest 701, Drive EOT Sensor Verification Test

Subtest 701 verifies that EOT can be sensed by writing until EOT is detected. This subtest rewinds the tape to BOT and continues to write 8-Kbyte blocks until EOT is sensed. It is probable that the tape will go off the spool if EOT is not sensed.

Interactive Debugger

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

- Use the **-d** option when invoking the diagnostic (enter **dev5210**[x] **-d**). No subtests are executed.
- The prompt "`User Debug Option Mask [0x0-0xfff,?]`" provides a bit option to force the diagnostic to enter the debugger after an error is reported. To ensure that this option is set, **0x10** must be ORed into the hexadecimal bit-pattern response for the prompt.
- Enter a "**:**" when in single-step mode.

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

Figure dev5210–6, 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.

    boot                                      - reboot the CCU driver
    buffer                                    - display main memory buffer data
    cd      [DIRECTORY]                       - change to DIRECTORY
    continue [-p]                             - continue from single step
    echo    [-n] [word1 [word2 ...]]          - write message to display
    exit                                      - exit the diagnostic
    fb      begin [end] value [incr [step]]   - fill bytes (VIOP)
    fw      begin [end] value [incr [step]]   - fill words (VIOP)
    fl      begin [end] value [incr [step]]   - fill longs (VIOP)
    help    [COMMAND ...]                     - display general or specific help
    mb      begin [end]                       - modify/[dump] bytes (VIOP)
    mw      begin [end]                       - modify/[dump] words (VIOP)
    ml      begin [end]                       - modify/[dump] longs (VIOP)
    mfb     begin [end] value [incr [step]]   - fill bytes (Main mem)
    mfw     begin [end] value [incr [step]]   - fill words (Main mem)
    mfl     begin [end] value [incr [step]]   - fill longs (Main mem)
    mmb     begin [end]                       - modify/[dump] bytes (Main mem)
    mmw     begin [end] [count]               - modify/[dump] words (Main mem)
    mml     begin [end] [count]               - modify/[dump] longs (Main mem)
    pause   [-n] [seconds]                    - pause for <C/R> or seconds
    quit                                      - exit debug mode
    reg                                       - display current VBTC registers
    reset                                     - reset VBTC/tape drive
    skip    [-d dbg_mask] [-emq] [count]      - set single step skip mode/count
```

In addition to the help screen in the previous figure, you can display help for a specific command by entering:

help *command*

where *command* is the desired debugger command. Abbreviations of desired commands may be used as long as they are unique. For example, to display help for all commands starting with the letter "r," enter **help r**.

Interactive Debugger Command Descriptions

boot

Usage: **boot**

Reboots the CCU driver.

> **WARNING**
>
> This command can cause a subtest to fail if used from within the middle of the subtest.

buffer

Usage: **buffer**

Displays main memory buffer address and length.

cd

Usage: **cd** [DIRECTORY]

Changes to another directory, where DIRECTORY is any valid directory path. If DIRECTORY is omitted, the default path is $HOME or / if $HOME not set.

continue

Usage: **continue** [-p]

Continues from single step after restoring previous skip mode and counter, where *-p* prints the current skip option setting and does not continue.

This command restores the previous skip count and exits the debugger. The **-p** option prints the current skip option setting and does not continue. This command is useful for continuing from a single step pause point with the same skip options that last caused the test to pause in single step mode.

echo

Usage: **echo** [-n] [word1 [word2 ...]]

Writes words separated by blanks and terminated by a newline to the display, where *-n* means do not echo the terminating newline character.

exit

Usage: **exit**

Exits the diagnostic. This command exits the entire test. If the command is executed interactively, you will be asked to confirm that you really want to exit the test.

fb, fl, fw

Usage: **fb begin value**
fb begin end value [incr [step]]
fb begin,count value [incr [step]]

fl begin value
fl begin end value [incr [step]]
fl begin,count value [incr [step]]

fw begin value
fw begin end value [incr [step]]
fw begin,count value [incr [step]]

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

- **begin** is the starting address
- **value** is the initial fill value
- **end** is the ending address
- **incr** is the fill value increment
- **step** is the address increment
- **count** is the count of elements to fill

The first format (e.g., **fb begin value**) stores *value* at address *begin*.

The second format (e.g., **fb begin end value** [incr [step]]) fills from the address *begin* up to and including address *end* with the value *value*. If the optional *incr* parameter is specified, *value* is incremented by *incr* after each fill. If *incr* is followed by *step*, the fill address is incremented by *step* elements instead of the normal step of one.

The third format (e.g., **fb begin,count value** [incr [step]]) is identical to the second with one exception: *end* is not specified. Instead, the *end* parameter is calculated from the *count* parameter.

The following examples illustrate use of these commands.

1. *ffl 2000000 12345678*

   The above command stores one longword (32 bits) of value 0x12345678 at main memory address 0x200000.

2. *ffl 200000,0d1000 0*

   The above command zeroes 1000 longword (32 bit) locations starting a main memory address 0x200000.

3. *fl ffa048,0d20 78000200 1*

   The above command maps 20 CCU main memory windows starting at window 18 to contiguous physical main memory addresses starting at address 0x200000. The map registers are set up for modes ACCEL, PBUS, 68K, and ALL-VME-SLOTS.

4. *fb 30000,0d40 10 4 2*

   The above command fills 40 even bytes starting at CCU address 0x30000 with a value that begins at ten and increments by four each time.

help

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

Displays general or specific help information, where COMMAND is the desired debugger command. Abbreviations of desired commands may be used as long as they are unique. For example, the following command displays help for all commands starting with the letter "r":

help r

mb, mw, ml

Usage: **mb begin,count**
mb begin end
mb begin

mw begin,count
mw begin end
mw begin

ml begin,count
ml begin end
ml begin

Displays and/or modifies 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 starting CCU microprocessor address
- **end** is the ending CCU microprocessor address
- **count** is the number of elements to display

The first format (e.g., **mb begin,count**) displays *count* number of elements from the starting address. The second format (e.g., **mb begin end**) displays all elements from the address *begin* up to and including address *end*.

The third format (e.g., **mb begin**) 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, and 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
<CCU: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.

mfb, mfw, mfl

Usage: **mfb begin value**
mfb begin end value [incr [step]]
mfb begin,count value [incr [step]]

mfl begin value
mfl begin end value [incr [step]]
mfl begin,count value [incr [step]]

mfw begin value
mfw begin end value [incr [step]]
mfw begin,count value [incr [step]]

Fills main memory with specified pattern in byte-at-a-time mode (mfb), word-at-a-time mode (mfw), or longword-at-a-time move (mfl), where:

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

The first format (e.g., **mfb begin value**) stores *value* at address *begin*. The second format (e.g., **mfb begin end value** [incr [step]]) fills from the address *begin* up to and including address *end* with the value *value*. If the optional *incr* parameter is specified, *value* is incremented by *incr* after each fill. If *incr* is followed by *step*, the fill address is incremented by *step* elements instead of the normal step of one.

The third format (e.g., **mfb begin,count value** [incr [step]]) is identical to the second with one exception: *end* is not specified. Instead, the *end* parameter is calculated from the *count* parameter.

mmb, mmw, mml

Usage: **mmb begin,count**
mmb begin end
mmb begin

mmw begin,count
mmw begin end
mmw begin

mml begin,count
mml begin end
mml begin

Displays and/or modifies 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 starting main memory address
- **end** is the ending main memory address
- **count** is the count of elements to fill

The first format (e.g., **mmb begin,count**) displays *count* number of elements from the starting address. The second format (e.g., **mmb begin end**) displays all elements from the address *begin* up to and including address *end*.

The third format (e.g., **mmb begin**) 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, and 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
<Main-Mem: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 c03fc1 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.

pause

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

Wait for specified amount of time or for a RETURN if the time is omitted, where *-n* means do not echo the pause message, and *seconds* specifies the number of seconds to pause.

quit

Usage: **quit**

Exits the interactive debugger and continues to the next single step point, if applicable.

reg

Usage: **reg**

Displays current VBTC register state. This command reads the VBTC registers and displays the entire register image in an easily-readable format.

reset

Usage: **reset**

Resets the VBTC controller and its attached drive(s) by writing (ORing) a one into the Reset Register at address offset 0xf. It then resets the one by ANDing with 0xfe.

skip

Usage: **skip** [-emqrx] [-d [MASK]] [COUNT]

Displays or sets the single step modifier and skip counter, where:

-e	Skips until CCU driver returns an error (non-zero in the status field)
-m	Same as -e, except non-fatal media errors are not stopped on
-q	Disables printing while in skip mode. This will cause the test to "skip pause points" at normal execution speed since nothing will be printed (i.e., no pause messages)
-r	Resets current skip options (e.g., skip count = 0)
-x	Exits debugger after skip command
COUNT	Skips the next COUNT (or 1 if COUNT omitted) single step pause points. If a COUNT is specified with the *-e* or *-m* option, the test will skip the next COUNT errors before pausing.
-d [MASK]	Sets debug mask to MASK. If MASK is omitted, the current mask is printed. The debug bit definitions follow:

```
** USER Debug Options (bit mapped) **
0x0001: Disable Asynchronous Command Mode
0x0002: Disable Multi-Record Buffering
0x0004: Disable Reset of Controller After ^C
0x0008: Disable Hard Error Trapping (logging)
0x0010: Enter Debugger After an Error Occurs
0x0020: Disable copying of errors to error file
0x0100: Pause AFTER each CCU driver command is returned
0x0200: Show each CCU driver message AFTER it is returned
0x0400: Pause BEFORE each CCU driver message is sent
0x0800: Show each CCU driver message BEFORE it is sent
0x1000: Automatically enter debugger during a pause
```

If no options are specified, only the current skip modifier and counter are printed.

The **skip** command is useful when you want to avoid stepping through several hundred single step points (pressing RETURN each time). By stopping only on errors (*-e* or *-m*), the test does not stop at every single step point. With the *-q* option, the test does not print single step displays until it stops at one.

Error Messages

The diagnostic reports four categories of errors:

- Tape positioning errors
- Register mismatch errors
- Data compare errors
- Media errors (not fatal in most cases)

> **NOTE**
>
> There are two type of media errors. Formatter-corrected errors are media errors that were mathematically corrected by the tape formatter. These can be ignored or retried based on selected options from the test parameters. Uncorrectable media errors are errors that cannot be corrected by the formatter and must be retried or ignored.

In all cases, the diagnostic returns as much information as possible. The format of the first three types of error messages (tape positioning, register mismatch, and data compare) are basically consistent.

In the following examples, `exp` refers to the expected value, `act` refers to the actual value, `dc` refers to the insignificant bits (don't care bit mask). Also, `diff` is the difference between the actual value and the expected value excluding the insignificant bits. The `diff` value is obtained by ANDing the actual value with the complement of the insignificant bits (don't care bit mask) and exclusive ORing the result with the expected value. All bits that are in discrepancy are set. For example:

```
exp/act/dc = 0x0000/0x0100/0x0000, diff = 0x0100
```

The following figures show examples of all four types of messages:

Figure dev5210–7, Tape Positioning Error Example

```
*****  Thu May 31 14:42:21 1990  *****
Test:    dev5210.t  1.1    Class: 5    Subtest: 504 1.1   Count: 1    Error: 7
Failed:  GCR: Skip Block Fwd/Bwd Test

        ------------------- trace point: 504.115.20 -------------------
Tape Position:     File 1 of 1, Block 14 of 200
Current Action:    Verifying the positional coordinates of the tape are as
                   expected.
Other Info:        Error occurred while skipping in the forward direction from
                   block 9 to block 14.  There were 3 successful skips
                   completed prior to the error.
Error Description: The read tape block header indicates a tape positioning
                   error has occurred (i.e., a seek error).
Tape Block Header Mismatch:
          Block-num Block-cnt File-num  File-cnt  Block-size    Wrt-Subtest
exp/act = 000e/000f 00c8/00c8 0001/0001 0001/0001 002000/002000 502
```

NOTE

In the following example, the bytes that miscompare are marked by an asterisk (*).

Figure dev5210–8, Data Compare Error Example

```
*****  Thu May 31 14:52:43 1990  *****
Test:    dev5210.t  1.1    Class: 1    Subtest: 105 1.1   Count: 1    Error: 7
Failed:  VBTC FIFO Variable Size Write/Read Test

        -------------------- trace point: 105.125 --------------------
Iteration:         Loop 1 of 1914
Current Action:    Comparing the buffer fill pattern with the data beyond the
                   expected read area.
Other Info:        The failure occurred with a transfer size of 1 and began on
                   an EVEN address.  Transfer sizes from 1 through 957 were
                   being tested on both EVEN and ODD addresses.
Error Description: Data compare error occurred.
                   1 data compare error(s) occurred.  The first was at byte
                   offset 1 (0x1) in the data block under compare.  The data
                   block begins at main memory address: 0x02002000 and is 1
                   bytes in length.  It is in NORMAL order.  The error was in
                   the 512 byte overflow space (at byte offset 0 (0x0) beyond
                   the end of the read data).
Expected/Actual:
    02002001: aa/33* 55/55  aa/aa  aa/aa  55/55  aa/aa  55/55  55/55
    02002009: aa/aa  55/55  aa/aa  aa/aa  55/55  aa/aa  55/55  55/55
    02002011: aa/aa  55/55  aa/aa  aa/aa  55/55  aa/aa  55/55  55/55
    02002019: aa/aa  55/55  aa/aa  aa/aa  55/55  aa/aa  55/55  55/55
```

Figure dev5210–9, Register Mismatch Error Example

```
*****  Thu May 31 14:56:43 1990  *****
Test:    dev5210.t  1.1    Class: 2    Subtest: 201 1.1   Count: 1    Error: 7
Failed:  Formatter/VBTC Interface Test

       -------------------- trace point: 201.125 --------------------
Current Action:     Attempting to execute a tape formatter NOP command.
Other Info:         The STC Mnemonic of the interface signal that was under
                    test was "FPTS".  This signal is located on connector J7,
                    pin 57.
                    This failure occurred at NOP number 3 of the DMS/NOP
                    command sequence.  See the STC 2920 series maintenance
                    manual (Ch 4) for more info.
Error Description: The VBTC register state at completion of the last command
                    did not match the expected register state.
VBTC Register mismatches:
        3fda(rtsb):  exp/act = 0x02/0x81, dc = 0x9c
Extra bits(rtsb):   0x1<Rew>
Missing bits(rtsb): 0x2<Wrt_Prt>
```

Figure dev5210–10, Media Error Example

```
                    ---- TEST MEDIA ERROR TOTALS ----

Media Error Class ---- Total ---- NOT Recovered ---- Recovered ---- Ignored
Formatter-Corrected       13                  0              0           13
Uncorrectable              0                  0              0            0
```