CONVEX VMEbus SMD/ESDI Disk and Formatter (*dev5130*) Diagnostics Manual

Document No. 760-003030-000

First Edition
May 1991

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX VMEbus SMD/ESDI Disk and Formatter
(dev5130) Diagnostics Manual
Order No. DHW-242
First Edition

Printed in the United States of America

Revision Sheet

CONVEX VMEbus SMD/ESDI Disk and Formatter (dev5130) Diagnostics Manual

Edition	Document No.	Date	Description
First	760-003030-000	May 1991	First release. Contains the *dev5130* diagnostic test information from the *CONVEX PBUS I/O Systems Diagnostics Manual.*

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

Appendixes

A Reporting Problems

List of Tables

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Purpose and Intended Audience

This manual explains how to run the *dev5130* diagnostic, which checks the Interphase 4200 VMEbus/Storage Module Device (V/SMD) and Interphase 4201 VMEbus/Enhanced Small Device Interface (V/ESDI) disk controllers and attached drives, formats disk drives, verifies drives, and performs disk maintenance. This document is not a tutorial, but rather a reference for the users of the *dev5130* diagnostics, including field service and manufacturing test personnel, as well as the diagnostics sustaining staff. In addition, CONVEX customers can use this manual to execute the *dev5130* diagnostic.

Scope

This manual applies to all CONVEX computers.

Organization

This document consists of the following:

- **Chapter 1. Diagnostics Environment**—Introduces the theories and concepts that underlie I/O diagnostics on CONVEX machines as well as the basic overview, philosophy, and structure of I/O diagnostics.
- **Chapter 2. EGOS Overview**—Provides a brief overview of the Event Governed Operating System (EGOS) and how it relates to device and peripheral diagnostics testing.
- **Chapter 3. Dshell Overview**—Provides a brief overview of and a general introduction to the *dshell* utility.
- **Chapter 4. VMEbus SMD/ESDI Disk Test and Formatter Test (*dev5130*)**—Describes how to operate the diagnostic, including prerequisites, test invocation, hardware initialization sequence, and class descriptions. It also describes the manufacturer's defect list, track relocation area, disk parameters file, the diagnostic cylinder description, and error codes and messages.
- **Appendix A. Reporting Problems**—Provides an example of the CONVEX ***contact*** utility for reporting minor software and hardware problems.

Notational Conventions

The notational conventions used in this text are listed below:

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

Warnings

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

WARNING

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

CAUTION

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

NOTE

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

Associated Documents

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

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

Ordering Documentation

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

The product number for this manual is DHW-242.
The document number for this manual is 760-003030-000.

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

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

Technical Assistance

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

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

Reader's Forum

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

Chapter 1
Diagnostics Environment

1.1 Overview

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

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

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

1.2 Test Program Naming Conventions

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

- *cat* is the subsystem being tested
- *type* is the type of test being performed, e.g., standalone, self-test, or offline functional test
- *dev* is the device being tested, e.g., disk, tape, or printer. This segment of the test program name is used *only* if the category is a device.
- *nn* is a CONVEX code used for distinguishing between test programs
- *suffix* is one of three program identifiers:
  - *.t* are programs that execute on SP2
  - *.x00* and *.rnn* are object files for different target processors other than the SP2. The target processor depends on the subject of the test. The test program name must have the test program category (*cat*) at the beginning of the name to determine the target processor.

1.2.1 Test Program Categories

Test program categories include those tests for the CPU, peripheral devices, I/O system, memory system, SP2, and entire system. For example, *cpu4041* is a CPU vector instruction test while *mem4000* is a memory system functional test. The following table lists test program categories:

Table 1–1, Test Program Categories

TEST PROGRAM CATEGORIES	
Test Category (*cat*)	**Description**
cpu	CPU subsystem related test
dev	Peripheral device test
io, idc, tli	I/O subsystem related test
mem	Memory subsystem related test
spu	SP2 subsystem related test

1.2.2 Test Program Types

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

Table 1–2, Test Program Types

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

1.2.3 Test Program Device Types

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

Table 1–3, Test Program Device Types

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

1.2.4 Examples of Test Program Names

The following table presents some examples using the naming conventions outlined above:

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

Table 1–4, Example Test Program Names

EXAMPLE TEST PROGRAM NAMES	
Test Program Name	**Description**
cpu4041.t	SP2 object code in *b.out* format for *cpu4041*
cpu4041.rnn	C210 or C220 machine object code in SOFF format (relocatable)
cpu4041.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
mem4000.t	SP2 object code in *b.out* format for *mem4000*
mem4000.x00	C210 or C220 machine object code in SOFF format (linked to run in segment 0)
dev4100.t	SP2 object code in *b.out* format for *dev4100*
dev4100.x00	IOP object code in *b.out* format

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2
EGOS Overview

2.1 Overview

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

2.2 Purpose of EGOS for Diagnostic Testing

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

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

2.3 EGOS for the Multibus Interface

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

2.4 EGOS for HSP Interface, HSP EGOS

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

2.5 EGOS for VME Interface, VIOP EGOS

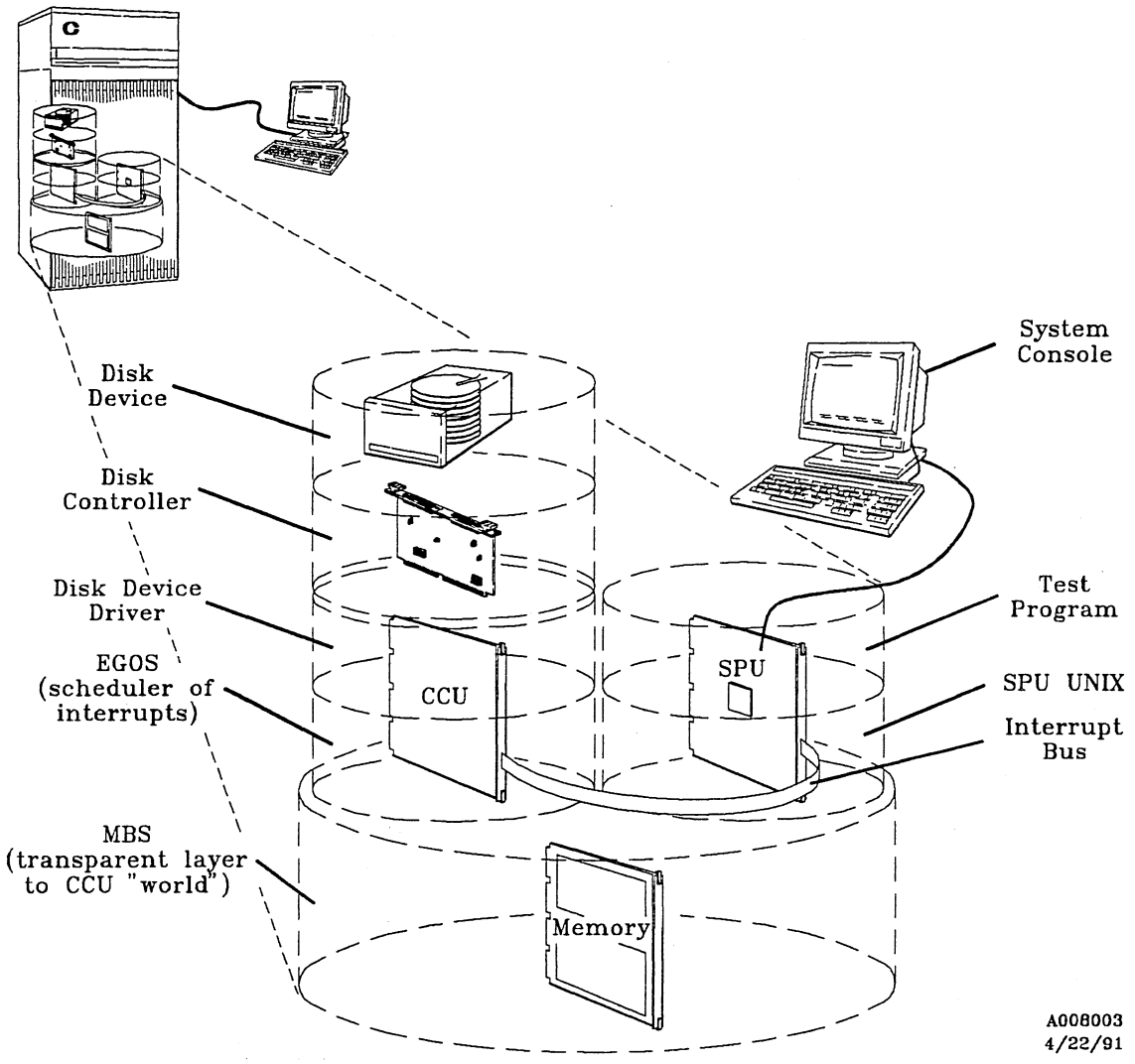
The VME interface version of EGOS is designed with a scheduler for the VIOP and is called VIOP EGOS. VIOP EGOS supports event driven device drivers as well as process type device drivers. VIOP EGOS utilizes a *sleep/wakeup* type of process control that improves efficiency of the device driver and makes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

2.6 EGOS Position in the Environment

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

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

Figure 2–1, EGOS' Position in the Environment



THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3
Dshell Overview

3.1 Overview

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

3.2 Diagnostic Shell (*dshell*) Overview

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

The *dshell* has two basic functions:

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

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

Table 3–1, *dshell* Commands

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

3.3 Syntax Help for *dshell* Commands

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

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

```
: loop
Proper syntax is:

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

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4
VMEbus SMD/ESDI Disk Test and Formatter (*dev5130*)

4.1 Overview

Test *dev5130* is a multi-purpose disk test which allows the user to do the following:

- Verify Interphase 4200 VMEbus/Storage Module Device (V/SMD) and Interphase 4201 VMEbus/Enhanced Small Device Interface (V/ESDI) disk controllers and attached drives operate properly.
- Format or reformat up to 12 drives at one time.
- Verify previously formatted drives.
- Perform disk maintenance such as slipping of bad sectors and mapping bad tracks to alternate tracks. All maintenance operations can be performed nondestructively so user data can be retained.

> **CAUTION**
>
> Some subtests in *dev5130* are data destructive.

> **WARNING**
>
> CONVEX UNIX cannot be running concurrently with the *dev5130* test.

This test description is intended as a reference for users who have a good understanding of disk architecture and basic controller operations and want a tool that will allow them to track down and fix often difficult to locate problems with disks. For the casual user, in other words, the person who wants to format a disk now and then or slip a defective sector or verify a disk by reading all the usable parts, the user should refer to the section "Examples of Typical Usage of This Test" within this test description. It is suggested that the Table of Contents be used as an outline for finding information within this test.

The *dev5130* test is divided into two code bodies designated *dev5130.t* and *dev5130.x00*. The code bodies *dev5130.t* and *dev5130.x00* refer to Service Processor Unit (SPU) and VIOP resident code, respectively. The first body of code *dev5130.t* executes under the *dshell* from SPU UNIX. The primary function of this code body is to interface the user to the test and to control the testing sequence. This interface software consists of user prompts and error message printouts for all subtests except Subtest 300, Format Setup and Subtest 400, Interactive Test. The user prompts depend on the *dshell* extension routine *read_parm* for user inputs. The use of this routine allows

the user interface of *dev5130.t* to be directly compatible with the other peripheral test programs. Subtests 300 and 400 are interactive with their own prompts and set of commands including help commands.

The second body of code is *dev5130.x00* and executes under the Event Governed Operating System (EGOS). This body of code contains the command processing software used to test the controller(s) and SMD devices. The code communicates to the Service Processor via the Message-Based Subsystem (MBS) and through main memory.

4.2 Prerequisites and Required Equipment

The SMD 4200 or ESDI 4201 controller must be configured for use. This is accomplished by changing jumpers on the board. Only changes to the VMEbus address of the controller are supported by this test. All other jumpers must be in a predefined configuration. All jumpers except the bus request block and address jumpers should be set properly by Interphase.

The configuration options and the CONVEX configuration are defined in the following table:

Table 4–1, 4200 Controller Jumper Configurations

CONFIGURATION OPTION	POSSIBLE CONFIGURATIONS	CONVEX CONFIGURATIONS
Address Modifier	0x29; 0x2D	0x2D
Base Address	0x0000,0x0200,...,0xFE00	Any are possible
Bus Request Priority	0-3	3

All other jumpers are for factory use only and should not be changed.

The test requires the system configuration in one of the next two table depending on the type of machine under test.

Table 4–2, Hardware Requirements (C1, C120)

ITEM	MINIMUM	MAXIMUM
Interphase SMD 4200/ESDI 4201 controller	1	12
SMD device	1	12
Input/output processor (VIOP)	1	5
Multibus card cage	1	10 (2/IOP)
Service processor unit (SPU)	1	1
VMEbus control unit (VBCU)	1	12
Memory control unit (MCU)	1	1
Memory array unit (MAU)	4MB	System limit

Table 4–3, Hardware Requirements (C200 Series)

ITEM	MINIMUM	MAXIMUM
Interphase SMD 4200/ ESDI 4201 controller	1	12
SMD device	1	12
Input/output processor (VIOP)	1	12
Multibus card cage	1	12
Service processor unit (SP2)	1	1
VMEbus control unit (VBCU)	1	12
Memory system (one odd, one even)	1	System Limit
Peripheral interface adapter (PIA)	1	4
CPU utilities (CPX)	1	1

4.3 Test Invocation

Test *dev5130* executes under the Diagnostic Shell (Dshell), and supports most of the features of the Dshell. Control of failures is provided by the test since the Dshell does not support multiple drive operations where one drive could fail and yet the subtest should continue on the remaining drives. The Dshell permits the user to initiate subtests in any order. Therefore, each subtest is designed to be independent of all other subtests except in Class 3 which performs system formatting. The default order of subtest execution is from the least complex to the most complex operation.

To invoke the *dev5130* test, use the procedure shown in the following figure. All responses in **boldface** are entered by the user. The prompts and responses in the following figure would appear sequentially on the screen, one line at a time. All the prompts and responses are shown in one figure for convenience.

CAUTION

Entering only **test dev5130** should never be done on unformatted SMD/ESDI drives since Class 1 and Class 2 tests can overwrite the original Manufacturer's Defect map.

Figure 4–1, Test Invocation Sequence

```
(spu)>  cd /mnt/test (RETURN)
(spu)>  sysreset (RETURN)
(spu)>  mminit -s (RETURN)
(spu)>  dshell (RETURN)

CONVEX DIAGNOSTIC SHELL

: test dev5130 [-c [class number(s)]] [-s [subtest number(s)]] [+> filename] (RETURN)
```

NOTE

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

Execute a specific class(es) of subtest(s) or one or more individual subtests by using the **-c** or **-s** options, respectively. Detailed information for using these options can be found in the "Dshell Overview" chapter within this manual.

Entering only **test dev5130** executes all *dev5130* subtests sequentially except Class 4 which must be invoked with the **-c** or **-s** options. The [+> *filename*] option appends all test results to file *filename*.

The following alternate test invocation procedure may be required in some cases.

CAUTION

The user response, **initall**, is typically required if the *initall* utility has not been run since the last power up. However, if any problems have occurred subsequent to the last time *initall* was run, (i.e., system crash, hard error, or failure of previous diagnostic), it should be run again. In this case, failure to run *initall* could result in invalid test results.

NOTE

The *initall* utility requires a significant amount of time (2 to 3 minutes depending on whether the control stores have been previously loaded) to execute. If no system abnormalities have occurred subsequent to the last time the system was booted or *initall* was executed, it is not necessary to run *initall*.

Figure 4–2, Alternate Test Invocation Sequence

```
(spu)> cd /mnt/test RETURN
(spu)> initall RETURN
(spu)> dshell RETURN
: test dev5130 [-c [class number(s)]] [-s [subtest number(s)]] [+> filename] RETURN
```

4.3.1 Test Parameter Menu

Once the test is invoked, the first prompt allows selection of defaults. If the test is run with defaults (by answering **y**, several prompts are skipped and the next prompt displayed is the `Device selection` prompt. If **n** is entered to the first prompt (no defaults are assumed), then a series of prompts are presented. The following figure shows all possible prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )). The prompts and responses in the following figure appear sequentially on the screen, one line at a time. To abort (terminate) the test, enter **^c** (hold down the CTRL key and press **c**) during test execution. All possible prompts and responses are shown in one figure for convenience.

For help or information during test parameter entry, enter one of the following characters followed by a RETURN:

Table 4–4, Getting Help During Test Parameter Entry

Character	Description
?	Displays this help menu
d	Displays *DB_diskfmt* file containing the drive parameters
e	Displays entered drive entries
h	Provides help for a specific prompt
i	Displays the */ioconfig* file

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

The **Test Parameter Menu** 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. However, the numbers displayed on the screen during testing may not correspond to those shown in the example **Test Parameter Menu**, as the questions illustrated are examples only.

Figure 4–3, *dev5130* 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
  ?         Prints an additional help menu
 1:  Default type (r[read only], w[rite allowed], n[o_defaults])
    [w,r,n]                                              (r) ->
 2:  Are writes to other than the diagnostics cylinder allowed
    [y,n]                                                (n) ->
 3:  Are writes to the diagnostics cylinder allowed
    [y,n]                                                (n) ->
 4:  Number of errors allowed per device each subtest
    [0-65535]                                            (0) ->
 5:  Number of devices failed after which test aborts
    [0-65535]                                           (12) ->
 6:  Threshold for new flaws found during pattern test
    [0-65535]                                            (0) ->
 7:  Verbosity of test [0-31]                            (3) ->
 8:  If pattern testing, start after last completed pattern
    [y,n]                                                (y) ->

             PERIPHERAL CONFIGURATION DATA
      CCU   Chassis  Type     CSR    Int Unit Type
    ------- ------- -------- ------ --- ---- --------
 1) viop 7    0     DKC-204  0x200  2   0    DKD-208
 2) viop 7    0     DKC-204  0xc00  6   0    DKD-214
Enter device 99 to begin user-defined configurations or -1 to end selection
 9:  Device selection [-1,1-2,99]                        (-1) ->
10:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                               (n) ->
11:  Serial number (5 to 31 characters) []
                          (Default: read from drive) ->
12:  Device selection [-1,1-2,99]                       (-1) ->
Enter VIOP -1 to end user-defined configurations
13:  VIOP [-1,3-7]                                      (-1) ->
14:  VMEbus Chassis [0-1]                                (0) ->
15:  Controller Offset in VMEBus [0x0-0xffff]
                                                    (0xe00) ->
16:  Interrupt number [1-7]                              (2) ->
17:  Unit number [0-1]                                   (0) ->
18:  Drive name []                                 (DKD-208) ->
19:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                               (n) ->
20:  Serial number (5 to 31 characters) []
                          (Default: read from drive) ->
Enter VIOP -1 to end user-defined configurations
21:  VIOP [-1,3-7]                                      (-1) ->
22:  Enter OK, or :NN to return to question NN [OK]
                                                        (OK) ->
```

At any time during the test parameter sequence, several options are available as denoted at the top of the Test Parameter Menu. The following list summarizes the available options:

:nn — Returns to an earlier prompt (n is the prompt number)

: — Advances to the next unanswered prompt

:? — Displays (reviews) all responses up to the current prompt

? — Requests help for the current prompt (if available)

^ — Returns to the previous prompt

4.3.2 Prompt Explanations

A description of the meaning of each prompt follows:

```
Default type (r[read only], w[rite allowed], n[o_defaults])
[w,r,n]                                              (r) ->
```

> **CAUTION**
>
> If **w** is entered to this prompt, then destructive writes are allowed to the entire disk and loss of data and system information is possible.

If **w**, **r**, or RETURN is entered, several test parameter prompts for default selections are skipped and the default values are retained with the exception of the following two prompts:

```
 Are writes to other than the diagnostics cylinder allowed
[y,n]                                                (n) ->
 Are writes to the diagnostics cylinder allowed
[y,n]                                                (n) ->
```

The default values for these prompts are set to **y** if **w** is entered for the `Default type` prompt. They are set to **n** if **r** is entered. The configuration file is then displayed and drive selection begins. Enter **n** to change any of the default parameters.

The following prompts are displayed if the first prompt is answered with **n**:

```
 Are writes to other than the diagnostics cylinder allowed
[y,n]                                                (n) ->
```

If answered with **n** or RETURN, writes are not allowed to cylinders other than the diagnostic cylinder. However, if answered with **y**, writes are allowed to all cylinders with the exception of the diagnostic cylinder which is controlled by the next prompt.

```
 Are writes to the diagnostics cylinder allowed
[y,n]                                                (n) ->
```

If answered with **n** or RETURN, writes are not allowed to the diagnostic cylinder. However, if answered with **y**, writes are allowed to the diagnostic cylinder.

```
Number of errors allowed per device each subtest
[0-65535]                                        (0) ->
```

If answered with **0** or (RETURN), a device fails if an error occurs during execution of the subtest. However, if answered with a value in the range [**1-65535**], a device fails if the number of errors that occur during the execution of the subtest exceeds this value.

```
Number of devices failed after which test aborts
[0-65535]                                        (12) ->
```

If a value in the range [**12-65535**] or (RETURN) is entered, the test will not terminate unless all devices fail because only 12 devices can be exercised at one time. However, if a value in the range [**0-11**] is entered, the test fails when the number of failed devices exceeds this limit.

```
Threshold for new flaws found during pattern test
[0-65535]                                        (0) ->
```

If **0** or (RETURN) is entered, a device fails if one or more new flaws are detected during pattern test. However, if a value in the range [**1-65535**] is entered, a device fails if the number of new flaws detected during pattern test exceeds this limit.

```
Verbosity of test [0-31]                         (3) ->
```

If **1** or (RETURN) is entered, a summary of all flaws for each drive is printed during Subtest 302, Fix Flaws. In general, verbosity is a sum of values which determines how informative the test is. The values which may be chosen are shown in the following table:

Table 4–5, Verbosity Values and Printed Information

VALUE	PRINTED INFORMATION
1	Print summary of bad sectors during Subtest 302, Fix Flaws Subtest
2	Print each grown defect as it is found
4	Print Manufacturer's Defect and Grown Defect list reads
8	Print pattern during the interactive command *pattern_test*
16	Print all errors during the pattern test in Subtest 301, Format and Pattern Subtest

For example, a value of 11 means print the data associated with values of 1, 2, and 8 (1+2+8=11).

```
If pattern testing, start after last completed pattern
[y,n]                                            (y) ->
```

If **y** or (RETURN) is entered, all drives begin pattern test after the last completed pattern. If the drives were previously using different pattern sets or any of the drives had not completed at least pattern 1, then pattern testing is restarted from the beginning on all drives. However, if **n** is entered, then pattern testing is restarted from pattern 1.

The following prompts are displayed if the first prompt is answered with **y**, or after the `If pattern testing, start after last completed pattern` prompt is answered:

```
Device selection [-1,1-2,99]                    (-1) ->
```

The range [1-2] in the list of valid inputs corresponds to all the drives listed in the configuration file */ioconfig*. This range varies depending on how many drives are in the */ioconfig* file.

If answered with **-1** or (RETURN), the input is rejected since this is the first request for a device. If answered with one value in the range [**1-2**], then that drive is selected. If at least one drive has already been selected, the value entered must correspond to a different configuration file entry from the entry previously chosen. However, if answered with **99**, all configuration information in subsequent prompts must be manually entered.

The following prompts are displayed if a value in the range [**1-3**] is entered for the `Device Selection` prompt:

```
Previously formatted on CONVEX system with Interphase 4200/4201 controller
[y,n]                                           (n) ->
```

If **n** or (RETURN) is entered, then Class 1 and Class 2 diagnostics format the portions of the disk they need. In Class 3, if the drive has defect lists, then a message is displayed that the disk appears to be previously formatted and a prompt asks whether to continue this drive as a formatted drive. If **n** is entered, the drive is failed. However, if **y** is entered, then Class 1 and 2 diagnostics only perform formatting when the format command is being tested. In Class 3, if the drive does not appear to have defect maps, then a message is displayed that the disk appears to be unformatted and a prompt asks whether to continue this drive as an unformatted drive. If **n** is entered, the drive is failed. The Class 4 subtest ignores the input to this prompt.

```
Serial number (5 to 31 characters) []
                    (Default: read from drive) ->
```

If (RETURN) is entered, then **y** should have been entered to the previous prompt since the serial number must be read from a formatted drive. If **n** was entered to the previous prompt, a serial number is entered at this prompt. If a serial number is entered, it can be from 5 to 31 characters in length and can consist of lower or uppercase characters, numbers, and dashes. Internally, all letters are converted to uppercase so lower and uppercase inputs have the same result.

```
Device selection [-1,1-2,99]                    (-1) ->
```

The range [1-2] in the list of valid inputs corresponds to all the drives listed from the configuration file */ioconfig*. This range varies depending on how many drives are in the */ioconfig* file. If **-1** or (RETURN) is entered, the test exits device selection and asks the user to confirm that all inputs are ok. However, if **99** is entered, then manual inputs of configuration data begins. If one value in the range [**1-2**] is entered, then that drive is selected. If at least one drive has already been selected, the value must correspond to a different configuration file entry from the one previously chosen.

The following prompts are displayed if the `Device Selection` prompt is answered with **99**:

```
VIOP [-1,3-7]                                   (-1) ->
```

If **-1** or (RETURN) is entered and at least one drive has been chosen, then the user is asked to confirm that all inputs are okay. If no drives have been chosen, then this input (-1) is rejected. If the test is executing on a C1 or C120 machine and one value in the range [**3-7**] is entered, then this value selects the position of the VIOP in the Central Processing Unit (CPU) card cage in the C1 or

C120. If the test is executing on a C200 Series machine and one value in the range [**0-3**] is entered, then this value selects the position of the VIOP in the CPU card cage in the C200 Series machine.

```
VMEBus Chassis [0-1]                                    (0) ->
```

If **0** or RETURN is entered, then VMEbus chassis 0 is selected. There are two chassis numbered 0 and 1. If **1** is entered, the second chassis is selected.

```
Controller Offset in VMEbus [0x0-0xffff]
                                                    (0xe00) ->
```

If **0xe00** or RETURN is entered, then the controller which has been set to address 0xe00 is selected. This must be the controller to which the desired drive is attached. Offsets range from 0x0000 to 0xfe00 on 512 byte boundaries (for example, 0x0000, 0x0200, 0x0400, ... etc).

```
Interrupt number [1-7]                                  (2) ->
```

If **2** or RETURN is entered, then interrupt two is used by the controller when an I/O operation completes to interrupt the VIOP. However, if a value in the range [**1, 3-7**] is entered, then this interrupt is used. There is only one requirement when choosing interrupt numbers. Each controller in a VME chassis must have a unique interrupt number. Two controllers in different chassis can use the same interrupt number.

```
Unit number [0-1]                                       (0) ->
```

If **0** or RETURN is entered, then the drive must be set to unit address 0 (usually a switch setting internal to the drive). The cable with the small connector (B cable) from the drive must also be connected to the unit 0 socket on the controller. However, if **1** is entered, then the drive must be set to unit address 1 and the B cable from the drive must be connected to the unit 1 socket on the controller.

```
Drive name []                                       (DKD-208) ->
```

If **DKD-206** or RETURN is entered, then the drive must be an NEC D2352A 520-Mbyte drive. The current alternatives to this drive are the **DKD-208** which is the NEC D2363 1.1-GByte drive and the **DKD-214** which is the Hitachi DK514-38 380-Mbyte Removable Disk Storage (RDS) drive. The RDS drive has an ESDI interface so it must be connected to the Interphase 4201 ESDI controller while the other drives connect to the Interphase 4200 SMD controller.

Any SMD or ESDI drive that works with the Interphase controllers can be used. The drive must be added to the */mnt/bin/lib/DB_diskfmt* file.

```
Previously formatted on CONVEX system with Interphase 4200/4201 controller
[y,n]                                                   (n) ->
```

If the drive is unformatted, has been formatted on any other system than a CONVEX system, or has been formatted with a controller other than an Interphase 4200/4201 controller, then **n** is entered.

> **NOTE**
>
> Previously formatted means that the drive has completed Subtest 300, Format Setup Subtest, not that all of Class 3 subtests have been completed.

If **n** or RETURN is entered, then Class 1 and Class 2 diagnostics format the portions of the disk they need. In Class 3, if the drive has defect lists, then a message is displayed that the disk appears to be previously formatted and the user is asked whether to continue this drive as a formatted drive. If **n** is selected, the drive is failed. However, if **y** is entered, then Class 1 and Class 2 diagnostics only do formatting when the format command is being tested. In Class 3, if the drive does not appear to have defect maps, then a message is displayed that the disk appears to be unformatted and the user is asked whether to continue this drive as an unformatted drive. If **n** is entered, the drive is failed. The Class 4 subtest ignores the input to this prompt.

```
Serial number (5 to 31 characters) []
                    (Default: read from drive) ->
```

If RETURN is entered, then **y** should have been entered to the last prompt since the serial number must be either provided at this prompt or be read from the drive. If **n** is entered to the previous prompt, the serial number must be entered at this prompt. If a serial number is entered, it can be from 5 to 31 characters in length and can consist of lower or uppercase characters, numbers, and dashes. Internally, all letters are converted to uppercase so lower and uppercase inputs have the same result.

```
VIOP [-1,3-7]                                    (-1) ->
```

If **-1** or RETURN is entered and at least one drive has been chosen, then the user is asked to confirm that all inputs are okay. If no drives have been chosen, then this input (-1) is rejected. If the test is executing on a C1 or C120 machine and one value in the range [**3-7**] is entered, then this value selects the position of the VIOP in the Central Processing Unit (CPU) card cage in the C1 or C120. If the test is executing on a C200 Series machine and one value in the range [**0-3**] is entered, then this value selects the position of the VIOP in the CPU card cage in the C200 Series machine.

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

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

```
**********************************************
* DATA DESTRUCTIVE OPERATION IS ABOUT TO START.*
* VERIFY WRITE-PROTECT IS SET ON ALL DRIVES    *
* EXCEPT THOSE YOU ARE TESTING/FORMATTING.     *
**********************************************
 Then reconfirm that you want to continue [yn] ->
```

If **y** is entered then the actual testing commences with the loading of a driver into the VIOPs and then the execution of subtests begins. However, if **n** is entered, the test terminates and returns to the SPU prompt.

When all prompts have been answered, the screen displays a test parameter summary which echos the prompts that have been answered. The following figure illustrates an example of a "Test Parameter Summary" screen. The actual values and responses vary according to the input.

Figure 4–4, Sample Test Parameter Summary

```
              TEST PARAMETER SUMMARY

Default type (r[read only], w[rite allowed], n[o_defaults]) : n
Are writes to other than the diagnostics cylinder allowed
                                                          : n
Are writes to the diagnostics cylinder allowed            : n
Number of errors allowed per device each subtest          : 0
Number of devices failed after which test aborts          : 12
Threshold for new flaws found during pattern test         : 0
Verbosity of test                                         : 3
If pattern testing, start after last completed pattern
                                                          : y
Enter OK, or :NN to return to question NN                 : OK

                        DRIVE CONFIGURATION DATA

       CCU  VME  Cntlr  Int Unit #    # Phys Log Prev
Drive   #    #   CSR  Level #  Cyl Hds Sec Sec Fmtd Name           Serial #
-----  ---  ---  -----  --- --- --- --- --- --- --- ------- ----------------
   1    7    0   0x200   2   0 1024  27  68  67  no DKD-208 00000
1000    7    0   0xe00   3   0 1024  27  68  67  no DKD-208 99999

Performing sysreset of ccus and memory...
Initializing I/O subsystem and Loading VIOP(s)...
```

4.4 Hardware Initialization Sequence

After the last prompt is entered, and before test code execution, the following message is displayed during hardware initialization:

Figure 4–5, Test Initialization Message

```
Performing sysreset...
Initializing I/O subsystem and Loading VIOP(s)...
```

If writes to any of the disks are allowed, the following message is displayed immediately following the summary of inputs:

Figure 4–6, Data Destructive Message

```
*********************************************
* DATA DESTRUCTIVE OPERATION IS ABOUT TO START.*
* VERIFY WRITE-PROTECT IS SET ON ALL DRIVES    *
* EXCEPT THOSE YOU ARE TESTING/FORMATTING.     *
*********************************************

Then reconfirm that you want to continue [yn] ->
```

If **y** is entered then the actual testing commences with the loading of a driver into the VIOPs and then the execution of subtests begins. However, if **n** is entered, the test terminates and returns to the SPU prompt.

Figure 4–7, Class 3 Test Initialization Message

```
Subtest 300   0:00:00       Prepare for format
```

Before *dev5130* test code execution, the following events occur:

- A sysreset is performed
- Main memory is allocated for the test
- SPU windows to main memory are initialized
- SPU local test variables are initialized
- The VIOP is booted and loaded
- A driver on the VIOP is started
- VIOP local test variables are initialized

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

4.5 Class Descriptions

The *dev5130* test contains four classes of subtests as shown in the following table:

Table 4–6, *dev5130* Test Classes

CLASS	DESCRIPTION
1	Controller functionality tests
2	Drive functionality tests
3	Drive system format tests
4	Interactive test

The following sections describe the different classes and each of their subtests. Each section contains a table listing each subtest in that class and a description of the subtest. The Class 1 and Class 2 tests verify the controller functionality of the following:

- Reset through VMEbus backplane reset and through Command Status Register
- Used portion of short I/O space
- Invalid command detection and reporting
- Command set coverage includes the following:

Tested commands	Untested commands
Diagnostics	Verify Sectors
Read Long	Reformat*
Write Long	Read Sectors Sequential*
Read Header	Write Sectors Sequential*
Read Raw Data	Verify Sectors Sequential*
Read CDC Flaw Map	Read Sequential, Disable Address Bump*
Report Configuration	Write Sequential, Disable Address Bump*
Write Sector Buffer	Verify Track
Read Sector Buffer	Verify Track Sequentially*
Report Sector ID	Extended Diagnostics*
Format with Sector ID	Dual Port Priority Select*
Initialize Long	Read and Scatter*
Report Configuration Long	Gather and Write*
Read Sectors	Read and Scatter 32*
Write Sectors	Gather and Write 32*
Verify Sectors	Read and Scatter 32 (list 2)*
Format Track	Read and Scatter 32 (list 2)*
Map Track	Buffer Allocation Check*
Handshake	Write After Cache*
Initialize	Clear Drive Fault
Restore	Map Sector
Seek	
Format Track with Data	
Read Noncached	
Track ID	
Fetch and Execute IOPB	

* The commands marked with an asterisk were not available when this diagnostic was designed. They have been recently added. None of these commands are used by the *dev5130* test or by CONVEX UNIX. When a drive faults, the *Restore* command is used to clear the drive fault and rezero the unit instead of using the *Clear Drive Fault* command.

4.5.1 Class 1 Subtests

Class 1 subtests consist of tests which exercise and verify most features of a controller. All of the tests are not data destructive since only the diagnostic cylinder is used for writing. Two options are given at the beginning of the test to specify whether writes to the diagnostic cylinder are allowed and whether writes to cylinders other than the diagnostics cylinder are allowed. These options control which subtests are allowed to run. The following table lists each Class 1 subtest and its description.

Table 4–7, Class 1 Subtests

SUBTEST	DESCRIPTION	TIME (hr:min:sec)	DATA DESTRUCTIVE
100	Controller Reset	0:00:05	NO
101	Verify Diagnostics Commands	0:00:02	NO
102	Verify Product Identification	0:00:02	NO
103	Verify *Write Sector Buffer* and *Read Sector Buffer* Commands (max throttle)	0:00:02	NO
104	Verify Command Chaining	0:00:03	NO
105	Verify *Initialize* and *Initialize Long* Commands	0:00:03	NO
106	Verify *Format Track* Command and all combinations of interleaves 1 and 2 and skews 0 and 1	0:00:07[1]	YES[2]
107	Verify *Format Sector ID* and *Report Sector ID* Commands	0:00:03[1]	YES[2]
108	Verify *Write Sectors* and *Read Noncached* Commands (all throttle settings)	0:00:15[1]	YES[2]
109	Verify *Format Track with Data* Command	0:00:02[1]	YES[2]
110	Verify *Write Long* and *Read Long* Commands	0:00:04[1]	YES[2]
111	Verify *Read Sectors* Command	0:00:05[1]	YES[2]
112	Verify *Map Track* and *Map Sector* Commands	0:00:03[1]	YES[2]
113	Verify Controller Detects Bad IOPBs	0:00:03	NO

[1] This time will double if two drives are connected to one controller.
[2] Set write protection for drive(s) to prevent loss of data.

4.5.1.1 Subtest 100, Controller Reset

Subtest 100 performs a VMEbus chassis reset and verifies that the controller's Command Status Register (CSR) contains a `0x4000` pattern. Then the LED bit (0x8000) is set and read back to ensure it was set. The BDCLR bit in the CSR (0x2000) is set which resets the controller. Then the CSR is verified to indicate that it contains a `0x4000` pattern. The one bit that is on in this pattern is the board OK bit which is the controller's way of indicating it has passed its internal diagnostics.

4.5.1.2 Subtest 101, Verify Diagnostics Commands

Subtest 101 performs each controller's diagnostics test using interrupts to respond to each controller's completion of a diagnostics command. The command's completion is verified by receiving an interrupt and checking that the board OK bit in the controller's CSR is set.

> **NOTE**
>
> The returned status and error codes for the remaining commands are checked in the Input/Output Parameter Block (IOPB) for correctness.

4.5.1.3 Subtest 102, Verify Product Identification

Subtest 102 executes a controller *Handshake* command which returns the controller's product code, product variation, revision level, month, day, and year. This information is printed each time the subtest is run. If the information fails gross range-checking, the Input/Output Parameter Block (IOPB) indicates that an error has occurred, or if the release date of the firmware is older than the oldest acceptable firmware date, then the controller and all attached drives fail.

4.5.1.4 Subtest 103, Verify *Write Sector Buffer* and *Read Sector Buffer*

Subtest 103 performs writes to and reads from one sector buffer in the controller's memory with an all '1's pattern and then an all '0's pattern. After the buffer is read, the data is compared. The throttle setting is set to 512 bytes per DMA burst which is the maximum amount of data that can be transferred in one controller *Write Sector Buffer* or controller *Read Sector Buffer* operation.

4.5.1.5 Subtest 104, Verify Command Chaining

Subtest 104 verifies command chaining by performing chained controller *Write Buffer* and controller *Read Buffer* commands.

> **NOTE**
>
> The throttle setting is 64 bytes per DMA burst which is the size of the cache on the VIOP. This is the size of all DMA bursts in Subtests 104 though Subtest 400 except Subtest 108.

4.5.1.6 Subtest 105, Verify *Initialize* and *Initialize Long*

Subtest 105 sets up a unit initialization block in VIOP memory and performs a controller *Initialize* command to load various device parameters into the controller. The subtest then uses the

controller *Report Configuration* command to verify the initialization. It then performs a controller *Initialize Long* command and verifies it using the controller *Report Configuration Long* command. All subsequent subtests use the *Initialize Long* command to initialize the controller. CONVEX UNIX also uses the *Initialize Long* command to perform controller initialization.

4.5.1.7 Subtest 106, Verify *Format Track* and all Combinations of Interleaves

Subtest 106 executes the controller *Format Track* command to format all tracks on the diagnostic cylinder. The tracks are formatted with interleaves of 1 through 16 while using a skew of 0. Then skews of 0-15 are performed with interleave first set to 1 and then set to 2. Each of these 48 tests is verified by reading the track headers with the controller *Track ID Read* command and comparing the track headers to the expected headers.

4.5.1.8 Subtest 107, Verify *Format Sector ID* and *Report Sector ID*

Subtest 107 verifies the controller *Format Sector Id* and controller *Report Sector Id* commands. First, track 0 of the diagnostic cylinder is formatted. Then a controller *Report Sector Id* command is performed on one track of the diagnostic cylinder and the headers are written out in reverse order using the controller *Format Sector Id* command. Then the format of the headers is verified by performing another controller *Report Sector Id* command and verifying the headers are reversed. Finally, the track is restored by another *Format Track* command.

4.5.1.9 Subtest 108, Verify *Write Sectors* and *Read Noncached*

Subtest 108 verifies the controller *Write Sectors* and *Read Noncached* commands. First, a *Format Track* command is performed on one track on the diagnostic cylinder. Then the *Write Sectors* and *Read Noncached* commands are verified by performing writes to the minimum sector on the track. Then using the multiple sector technique, writes are performed to the minimum + 1 sector and minimum + 2 sector. Then reads are performed on the minimum sector using single sector I/O and then reads are performed on the next two sectors using multiple sector I/O. Next, a verify is performed to ensure that the data written was read back successfully. These operations (except for the format track) are repeated for each throttle setting (4, 8, 16, 32, 64, ... 512 bytes per transfer to and from main memory). If other controllers are available, the above steps are repeated on the other controllers.

4.5.1.10 Subtest 109, Verify *Format Track with Data*

Subtest 109 verifies the *Format Track with Data* command. The *Format Track* command is used to format track 0 of the diagnostics cylinder by writing '0's to sector 0. Then the *Format Track with Data* command reformats the track with a `0x9abcdef0` repeating pattern. Then sector 0 is read and compared. The first four bytes of the sector are altered when formatted to reflect the cylinder, head and sector address of the sector. Then the first four bytes are verified.

4.5.1.11 Subtest 110, Verify *Write Long* and *Read Long*

Subtest 110 verifies *Write Long* and *Read Long* commands by reading data and Error Correction Code (ECC) information from each drive and then writing the data and information back to the drive. Before beginning, the subtest performs a *Format Track* command if needed on track 0 of

the diagnostic cylinder. The minimum sector and the next sector are written to with test patterns `0x00000000` and `0xffffffff` respectively and then the sectors are read using two *Read Long* commands. The headers and data fields are verified and the data and ECC information are swapped between the two sectors. The *Write Long* command is used to write the sectors back out and then the *Read Sectors* command is used to read the data back from the two sectors. The data should be in the opposite sectors corresponding to the sectors where the data was originally written.

4.5.1.12 Subtest 111, Verify *Read Sectors*

Subtest 111 verifies the *Read Sectors* command. The *Write Sectors* command is used to write to the first two cylinders. Then the *Read Sectors* command is used to read the two cylinders. The subtest is repeated for all default patterns (refer to Subtest 301, Format and Pattern Test for patterns).

4.5.1.13 Subtest 112, Verify *Map Track* and *Map Sector*

Subtest 112 verifies the *Map Track* and *Map Sector* commands. Two tracks on the diagnostic cylinder are formatted using the *Format Track* command and then sector 0 of the first track is mapped to sector 0 of the second track with the *Map Sector* command. Then the first sector is read and verified that the first word contains the address of the second sector in the first word (written by the *Format Track* command).

Then the two tracks are reformatted using the *Format Track with Data* command and the first track is mapped to the second track using the *Map Track* command. Then the sectors on the first track are read. The first word of each sector should contain the address from the second track (written by the *Format Track with Data* command).

Next, the first track is reformatted.

4.5.1.14 Subtest 113, Verify Controller Detects Bad IOPBs

Subtest 113 verifies the controller is capable of detecting bad Input/Output Parameter Blocks (IOPBs). A head address outside the controller's limit is sent to the controller for checking. The controller should reject the address. Then a cylinder address outside the drive's range is sent to the controller. The controller should also reject the cylinder address

4.5.2 Class 2 Subtests

Class 2 subtests consist of tests which exercise drives to verify that they are mechanically and electrically functional.

The Class 2 subtests verify the drive functionality of the following:

- Write and read every head
- Single and multi-track seeks
- Fault condition detection

The following table lists each Class 2 subtest and its description.

Table 4–8, Class 2 Subtests

SUBTEST	DESCRIPTION	TIME[3] (hr:min:sec)	DATA DESTRUCTIVE
200	Verify Head Switching	0:00:42[1]	YES[2]
201	Verify Sequential-Track Seeks	0:00:30[1]	YES[2]
202	Perform Min-to-Max Track Seeks	0:00:20[1]	NO
203	Perform Accordion Seeks	0:01:00[1]	NO
204	Perform Random Seeks	0:01:00[1]	NO
205	Verify Detect of Forced Faults (cylinder and head)	0:00:30[1]	NO

[1] This time will double if two drives are connected to one controller.
[2] Set write protection for drive(s) to prevent loss of data.
[3] Times shown are for one Hitachi DK514-38 drive.

4.5.2.1 Subtest 200, Verify Head Switching

Subtest 200 verifies head switching and also verifies that each head can write and read. Special formatting takes place for Class 2 subtests, if necessary, before the main portion of this subtest executes. The controller *Write Sectors* command is used to write to each sector of each track on the diagnostic cylinder. A controller *Read Noncached* command is used to read the data and then the write and read buffers for each sector are compared. First a `0x0000` pattern is used and followed by a `0xffff` pattern. The *Write Sectors* command and the *Read Noncached* command, and the compare are repeated ten times.

4.5.2.2 Subtest 201, Verify Sequential-Track Seeks

Subtest 201 verifies sequential-track seeking by first formatting head 0 on all cylinders (if writing is allowed). Then a controller *Report Sector ID* command is performed on head 0 of each cylinder starting with cylinder 0 to verify positioning.

4.5.2.3 Subtest 202, Perform Min-to-Max Track Seeks

Subtest 202 performs min-to-max track alternate seek. Special formatting takes place for Class 2 subtests, if necessary, before the main portion of this subtest executes. A controller *Track ID Read* command performs an implied seek to the minimum track. After the test verifies the cylinder number, a controller *Track ID Read* command performs an implied seek to the maximum track. When this cylinder is verified, the seek repeats to the minimum and maximum track 100

more times with verification at each track. Then the seek repeats, using the controller *Seek* command with no verification until the seek is complete. To complete the test, a controller *Track ID Read* command is used to read the last track and positioning is verified.

4.5.2.4 Subtest 203, Perform Accordion Seeks

Subtest 203 performs an accordion seek on a drive. Special formatting takes places for Class 2 subtests, if necessary, before the main portion of this subtest executes. A controller *Track ID Read* command performs an implied seek to the minimum track, maximum system track, min+1, max-1, min+2, max-2, etc., until a one-track seek is performed. The test verifies the seek by checking each of the headers just read. The accordion seek then repeats using the controller *Seek* command with no verification. After the last seek is performed, a controller *Read Header* command is performed to verify the final positioning. The test chains commands to two drives on a controller if both drives are connected to one controller.

4.5.2.5 Subtest 204, Perform Random Seeks

Subtest 204 performs 1000 random seeks within the specified minimum and maximum cylinders on a drive. Special formatting takes places for Class 2 subtests, if necessary, before the main portion of this subtest executes. Random numbers are created by initializing C's random number generator (*rand*) with a seed value of 113 and then calling *rand* for successive numbers. The controller *Track ID Read* command is used for implied seeks which verify positioning. The test then performs the random seek using the controller *Seek* command with no verification until the seeks are complete. Then a controller *Read Header* command is used to verify the last cylinder. The test chains commands to two drives on a controller if both drives are connected to one controller.

4.5.2.6 Subtest 205, Verify Detect of Forced Faults

Subtest 205 verifies the ability of the SMD drives to detect fault conditions. A controller *Initialize* command is used to set the head and cylinder limits to two greater than the maximum values specified in the file */mnt/bin/lib/DB_diskfmt*. Then a controller *Read Sectors* command is used to read a track from a cylinder that is two greater than the maximum. This generates a seek error. An error may not occur if the number of cylinders specified in */mnt/bin/lib/DB_diskfmt* is incorrect or if the drive has exactly 1023 or 1024 cylinders. The subtest expects certain errors or no errors since all known drives are characterized. For future drives, an `Expected error did not occur` error should be displayed if a drive fault or seek error does not occur.

Next, the test sets the head to two greater than the maximum value and then a controller *Write Sectors* command is used to write to this head on the diagnostic cylinder. This generates a write fault. After the write fault, the error does not clear until a drive reset is performed. The test performs a read of head 0 of the diagnostic cylinder and verifies that the write fault still exists. Then the test resets the drive and reads head 0 again; the write fault should no longer exist.

4.5.3 Class 3 Subtests

Class 3 subtests consist of tests which format new drives or reformat previously formatted drives for system use. The following table lists each Class 3 subtest and its description.

Table 4–9, Class 3 Subtests

SUBTEST	DESCRIPTION	TIME (hr:min:sec)	DATA DESTRUCTIVE
300	Format Setup	N/A	YES[3]
301	Format and Pattern Test	2:30:00[1]	YES[3]
302	Fix Flaws	0:01:00	YES[3]
303	Initialize Diagnostic Cylinder	0:04:00[2]	YES[3]
304	Verify System Format	0:08:00	NO
305	Verify Diagnostic Cylinder	0:00:30[2]	NO
306	Verify Pattern Test Error Threshold	0:00:02	NO

[1] Time varies depending on the types of drives and number of drives per controller. For one drive, the times are the following:

NEC 2352 (1/2 GByte) -> 1.25 hours
NEC 2363 (1 GByte) -> 2.30 hours
Hitachi 514-38 (RDS drive) -> 1.00 hours

Two drives on one controller will double these times.
If drive types are mixed, use the longer time.

[2] Multiply this time by the number of drives being formatted.
[3] Set write protection for drive(s) to prevent loss of data.

4.5.3.1 Subtest 300, Format Setup

Subtest 300 provides an interactive means of changing the way disks are formatted and also allows the user to input/edit flaw location information.

First, the subtest attempts to read the Manufacturer's Defect (MD) list and Grown Defect (GD) list on the innermost cylinder. If a MD list is not available, then the subtest sees if the file */mnt/usr/backup_maps/bkup.s* exists where *s* is the last five characters of the drive's serial number. If so, this copy is used.

First, the following initialization message for Subtest 300 is displayed:

Figure 4–8, Subtest 300 Initialization Message

```
Subtest 300   0:00:00       Prepare for format
```

Next, the following interactive prompt is displayed:

Figure 4–9, Subtest 300 Interactive Prompt

```
Type 'h' for help with commands
format setup (D1;Defect)->
```

If **h** is entered, a help facility is displayed which lists the available commands. The help display is shown in the following figure:

Figure 4–10, Subtest 300 Help Screen

```
Commands:
q[uit]                      - terminate the test and return to UNIX prompt
c[hange_mode]               - toggles logical sector/defect map entry mode
co[ntinue]                  - exit setup and continue with format
cyl hd sec                  - logical sector to be slipped
cyl hd bcai len             - defect map entry to be slipped
de[lete] cyl hd sec         - delete logical sector entry
de[lete] cyl hd bcai len    - delete defect map entry
d[rive] [n]                 - change/display current drive
f[ile] filename             - get inputs from specified file
fo[rmat_options]            - allows you to modify the format
h[elp]                      - display this information
l[ist] [{m,g,n,needs}]      - list all, manu[m], grown[g], new[n], needs
m[ap_track] cyl hd          - maps a track to an alternate
no_f[law_map]               - to inform formatter that no flaw map exists
no_t[rack_flaw] cyl hd      - used when no flaw data exists for a track
!unix-command               - execute unix command
'comment                    - comment; echoed to screen
```

4.5.3.2 Subtest 300, *change_mode* Command

c[hange_mode]

When manually entering defects, first select the entry mode. The default mode is Byte-Count-After-Index (BCAI) mode. To enter defects as logical sectors, modes must be switched with the *change_mode* command. Then enter the position of each flaw. In sector mode, this means typing the cylinder, head, and sector where the flaw exists. In BCAI mode, enter the cylinder, head, byte-count-after-index, and bit-length.

The input mode (logical sector or defect map mode) is revealed in the slip prompt as shown in the following figure:

Figure 4–11, Changing Input Modes

```
format setup (D1;Sector)->c
format setup (D1;Defect)->
```

4.5.3.3 Subtest 300, *continue* Command

co[ntinue]

The *continue* command is used to exit setup and continue with the format. When all inputs have been entered, type **continue**. Before Subtest 300, Format Setup exits, it will read the original Manufacturer's Defect (MD) maps from all drives for which no MD lists already existed and for which no backups existed.

Any drives which had no MD list on the innermost cylinder or backup file but did have original MD maps now have backup files containing the data from the original MD maps. The name of the files are */mnt/usr/backup_maps/bkup.s* where *s* is the last 6 digits of each drive's serial numbers.

For those drives that did not have a valid MD list or GD list, the defect cylinder (the cylinder where MD and GD lists are recorded) is formatted, pattern-tested and flaws remapped. Then the MD and GD lists for these drives are written to the drives. Subtest 300 then exits.

Refer to the *list*, *no_flaw_map*, and *no_track_flaw* commands for more information.

4.5.3.4 Subtest 300, *cyl hd bcai len* and *cyl hd sec* Commands

cyl hd bcai len

or:

cyl hd sec

The *cyl hd bcai len* command allows a defect map entry to be slipped. The *cyl hd sec* command allows a logical sector to be slipped. When a defect is entered, a list of logical sectors that correspond to the defect is generated as follows:

Figure 4–12, List of Logical Sectors For a Defect

```
format setup (D1;Defect)-> 293 1 572 9510
        Associated logical sectors: 59 0 1
```

The figure illustrates that three sectors are affected by the media flaw which starts at byte 572 from index and is 9510 bits long.

If two defects exist that affect the same sector, the logical sector will only be entered once. Subsequent entries of the same sector will result in the message:

```
This entry already exists (ignored)
```

For instance, given the following two defects:

cylinder	head	bcai	bit-length
10	0	70	1
10	0	80	1

Both defects fall in sector 0 so when the second defect is entered, it will not create a new entry in the input list.

4.5.3.5 Subtest 300, *delete* Commands

de[**lete**] *cyl hd sec*

or:

de[**lete**] *cyl hd bcai len*

The *delete* command is used to delete a logical sector entry or a defect map entry. An example of entering a bad location and deleting it is shown along with the *list* command in the following figure:

Figure 4–13, Entering and Deleting an Incorrect Location

```
format setup (D1;Sector)-> 17 7 23
format setup (D1;Sector)-> 17 7 46
format setup (D1;Sector)-> del 17 7 46
        Input 17 7 46 has been removed
format setup (D1;Sector)-> list

Summary of flaws on drive 1 serial number 003013
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR  SRC
  17  7  23  35032  4825      NONE USR |
```

As shown in the previous figure, use the *list* command to display inputs and defect list entries and with the *delete* command delete any incorrect entries except MD list entries (refer to the *list* command in Subtest 400, Interactive Test for more information about listing flaws). MD list entries are marked with MAP in the SRC column of a list.

4.5.3.6 Subtest 300, *drive* Command

d[rive] [*n*]

The *drive* command is used to change or display the current drive. If a drive is not specified, a list of all selected drives is displayed and a prompt is displayed to enter a drive number. An example of switching drives follows:

Figure 4–14, Switching Drives Using the *drive* Command

```
format setup (D1;Sector)-> d 2
        Drive 2 is now selected
```

4.5.3.7 Subtest 300, *file* Command

f[ile] *filename*

To enter the defect data from one or more files, select each file by typing:

file *filename*

where *filename* is the actual name of the file. The first line of the file defines the drive type and looks like this:

n *drivename*

where *drivename* is the drivename which can be found in */mnt/bin/lib/DB_diskfmt* on the Service Processor (for example, DKD-208 for the 1 GByte NEC 2363 drive). Following the drive name line, a file has remaining lines contain either BCAI inputs or logical sector inputs. A BCAI input is preceded with the letter **d** while a logical sector input is preceded with the letter **s**. The spare sector can be specified by using a logical sector number one greater than the last used sector number.

To specify a track that is to be mapped to an alternate track, use the letter **m** followed by the cylinder and head of the bad track. One or more spaces separate each item in the file and a new line can be started between any two items. To include comments in the file type a **#** at any place on a line. Everything from the **#** to the end of the line is ignored. Blank lines are also allowed. A file containing defect data is shown in the following figure:

Figure 4–15, Creating a Defect Data File

```
#------------------------------ start of file ----------------------------
# serial number: 003013
n  DKD-206
#    cyl  hd   bcai  len       cyl  hd   bcai  len  # NEC 2352 drive
d    157  15  24395   34    d  297  14  14102    4  # BCAI Inputs
d    466   0    206    1

#    cyl  hd  sec             cyl  hd  sec
s    457  12   42          s  821   5   11         # Logical sector inputs

#    cyl  hd
m    622  14                                      # map track input
#------------------------------ end of file ----------------------------
```

Defect input mode only applies to manual inputs. File inputs are controlled by the character that precedes the defect location.

4.5.3.8 Subtest 300, *format_options* Command

fo[rmat_options]

The *format_options* command is used to modify the format of the drive. To manipulate usage of the MD list or GD list, type **format_options**. Several prompts must be answered in response to this command. The prompts are displayed in the following figure:

Figure 4–16, Prompt Display for the *format_options* Command

```
For Drive 1 on controller (viop/vb/csr) 7/0/e00:
     Wipe out defect lists if they exist [yn] (n) -> RETURN
     Use manufacturer's defects during slip of sectors [yn] (y) -> RETURN
     Keep old grown defects  [yn] (y) -> RETURN
               - if 'n', next question will be asked
     About to delete grown defects. Continue  [yn] (y) -> RETURN
```

The defaults are in effect unless changed.

NOTE

If a drive is being reformatted, the use of the Manufacturer's Defects will be the same as the last time it was formatted unless this command is used to change them.

4.5.3.9 Subtest 300, *help* Command

h[elp]

This command lists the commands and their arguments.

4.5.3.10 Subtest 300, *list* Command

l[ist] [{**m,g,n,needs**}]

The *list* command is used to list all manufacturer defects, grown defects, new defects, or defect needs. If the currently-selected disk has been previously formatted, then the MD list and GD list will already be read when the first prompt appears. This data can be listed using the *list* command. If the drive is unformatted, the original manufacturer's defect map is read when exiting setup. If defect needs were listed after the *continue* command, type **list needs** to redisplay the current needs.

Figure 4–17, Using the *list* Command

```
format setup (D2;Sector)-> list

Summary of all flaws on drive 2 serial number 003418
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
 100  5  52  10536     1       NONE USR | 257  9  35  26401  1392       NONE USR
 257  9  58  26576   600       NONE USR |

format setup (D2;Sector)->
```

4.5.3.11 Subtest 300, *map_track* Command

m[ap_track] *cyl hd*

To map an entire bad track, use the *map_track* command. The test selects one of the alternate tracks for use.

4.5.3.12 Subtest 300, *no_flaw_map* and *no_track_flaw* Commands

no_f[law_map]

or:

no_t[rack_flaw] *cyl hd*

The *no_flaw_map* command is used to inform the formatter that no flaw map exists. The *no_track_flaw* command is used when no flaw data exists for a track. After using the *continue* command, if no defects were readable from anywhere or some of the original MD map was unreadable, then the user is prompted to enter the needed defect data. These inputs can come from a file of defects or from manual inputs. If inputs do not exist to satisfy the listed needs, then use one of the two commands to satisfy the test. To cancel a need for inputs due to an

unreadable map type **no_flaw_map**. To cancel a need for a specific track type **no_track_flaw** *cyl hd* where *cyl hd* is the track for which a need was listed.

4.5.3.13 Subtest 300, *quit* Command

q[uit]

The *quit* command is used to terminate the test and return to the UNIX prompt. The prompt in the following figure is displayed when the **quit** command is issued:

Figure 4–18, Executing the *quit* Command Within Subtest 300

```
format setup (D2;Sector)->quit
        Are you sure you want to terminate the test [yn] (y) ->
```

4.5.3.14 Subtest 300, Execute UNIX Command

! *UNIX Command*

The *!* (UNIX command) is used to execute a UNIX command while within Subtest 300. An example of its use is displayed in the following figure:

Figure 4–19, Executing a UNIX Command Within Subtest 300

```
format setup (D2;Sector)-> !cat def_003418     <-to illustrate unix cmd
# Serial Number: 003418
n DKD-206
d  100   5 10536      1
d  257   9 26401   2000
```

4.5.3.15 Subtest 300, Enter Comments

'comment

This command is used to enter a comment. To enter a comment, type the ' command and a comment. The comment is echoed so it is redirected output. This is useful to document redirected output.

4.5.3.16 Subtest 301, Format and Pattern Test

Subtest 301 formats a drive and pattern tests all sectors except the ones in the defect list cylinder. This includes sectors that will later be used as spares. The patterns used for pattern testing are listed in the following table:

Table 4–10, Patterns for Pattern Testing

SET	PATTERN	
1	MFM Data Encoding	
	0x00000000	0x6DB66DB6
	0xFFFFFFFF	
	0x55555555	
	0xAAAAAAAA	
2	2-7 RLL Data Encoding	
	0x00000000	0x9ABCDEF0
	0xCCCCCCCC	0xAF5BAF5B
	0x88888888	0x22222222
	0xBCDEF09A	
3	1-7 RLL Data Encoding	
	0x00000000	0x9ABCDEF0
	0xCCCCCCCC	0xAF5BAF5B
	0x88888888	0x22222222
	0xBCDEF09A	
4	Combination of Unlike Drives	
	0x6db66db6	0xffffffff
	0x9abcdef0	0xcccccccc
	0x00000000	0x33333333
5	User Specified Patterns	
	Patterns in *dev5130.patt* file	

If all drives that are to be formatted use the same encoding scheme, then the corresponding set of patterns will be used. However, if drives with different encoding schemes are formatted together, then pattern set 4 is used.

These default patterns can be replaced by up to 15 other patterns if the file *dev5130.patt* is created in the current directory or if it is created in */mnt/bin/lib*. The current directory is searched first. Each pattern is preceded with a 'p' and at least one space, tab or newline character. The pattern can be from 1 to 1024 hex characters per pattern (upper or lowercase letters are allowed). More than one line can be used to define the patterns. Blank lines are also allowed.

The following is an example of two patterns in a *dev5130.patt* file:

```
p A    p 23456789a
```

The first pattern is expanded so that every byte written contains a 0xAA pattern. The second pattern has 9 hex digits which are repeated when written. The following is the resulting data:

```
23456789 a2345678 9a234567 89a23456  789a2345 6789a234 56789a23 456789a2
                ---------- etc ----------
```

Any sector-related errors (header errors; hard Error Correction Code (ECC) error; or correctable ECC error) are retried immediately and, if an error occurs again in ten retries (does not have to be the same error), the sector location is saved for later slipping. Errors that do not repeat are discarded. A maximum of 2048 errors can be saved per drive. This includes the flaws from Subtest 300, Format Setup and any found during pattern test. If a verbose value of 16 is selected during test parameter inputs, each error is printed at the time it is detected. A verbose value of 2 results in only grown defects being printed. The last operation in Subtest 301, Format and Pattern Test writes the MD and GD lists.

4.5.3.17 Subtest 302, Fix Flaws

If the innermost cylinder does not yet contain the new Manufacturer's Defect (MD) list, then the drive fails.

If Subtest 301 Format and Pattern Test was not run, then the drive is formatted with one spare sector per track. Then all sectors which have been marked bad will be slipped. If more than one sector is bad on a track or a header-related error occurred on the track, the entire track is remapped to an alternate track. The number of cylinders worth of spare tracks for a particular drive is defined in the *DB_diskfmt* file.

If verbose value 1 is selected during the test parameter menu (this is the default along with printing each grown defect as it is found), a summary of bad sectors and tracks for each drive will be printed before slipping occurs.

The last operation writes the MD and GD lists.

4.5.3.18 Subtest 303, Initialize Diagnostic Cylinder

Subtest 303 writes special patterns to the diagnostic cylinder which is the cylinder just before the defect list cylinder. Sector zero of each track contains a table of contents describing what is written to each of the remaining sectors. If no sectors have been relocated on a track, sector 1 is marked bad. The next twelve sectors contain Error Correction Code (ECC) errors of length 1 to 12 respectively. The remaining sectors have the patterns listed in Subtest 301, Format and Pattern Test written to them. Sector 13 (or 14 if sector 1 is marked bad) will contain the first pattern, sector 14 (or 15) will contain the second pattern, and so on with the patterns repeating throughout the remaining sectors of each track.

4.5.3.19 Subtest 304, Verify System Format

Subtest 304 reads all five copies of the original Manufacturer's Defect (MD) list, Grown Defect (GD) list, and track map log and verifies that at least three of each are good. If not, the test fails.

Then the subtest reads every logical sector in the usable portion of the disk. Any errors are reported since all bad sectors should be relocated and be handled by the controller.

Any errors are reported and do count against the device errors per subtest limit. If the number of errors exceeds the limit, the drive is failed. The usable portion of each drive is all cylinders preceding the alternate track area and any used alternate tracks.

4.5.3.20 Subtest 305, Verify Diagnostic Cylinder

Subtest 305 reads each sector on the diagnostic cylinder and expects soft or hard Error Correction Code (ECC) errors on sectors 1 through 12 of each track (or sectors 2 through 13 if sector 1 was marked bad). The remaining sectors are read with data compare and must be error-free.

4.5.3.21 Subtest 306, Verify Pattern Test Error Threshold

A threshold can be set during user prompts which will determine how many new errors can be detected during Subtest 301, Format and Pattern Test pattern testing before this subtest fails. The intention is that all errors detected by pattern test should already be in the MD map. If the errors are actually new, this may indicate an unacceptable degradation of the disk since it was tested at the manufacturer or inaccuracy in manufacturer's media tester.

4.5.4 Class 4 Subtest

The Class 4 subtest consists of one test, the interactive test, which allows the user to read and display sectors, verify parts of disks, format single tracks, slip sectors, etc.

Table 4–11, Class 4 Subtest

SUBTEST	DESCRIPTION	TIME (hr:min:sec)	DATA DESTRUCTIVE
400	Interactive Test	N/A[1]	YES[2]

[1] The time for the interactive test depends on the operations performed.
[2] Set write protection for drive(s) to prevent loss of data.

4.5.4.1 Subtest 400, Interactive Test

Subtest 400 provides a special interactive test interface to allow the user to perform a variety of disk maintenance tasks such as pattern-testing portions of the disk, slipping bad sectors, formatting single tracks, and several other useful operations. Help information is available when needed.

> **CAUTION**
>
> Five of the commands are potentially data destructive: *pattern_test*, *slip_sectors*, *format*, *data_verify*, and *copy_from_spu*.

The first four commands do an automatic save of the data the command is going to overwrite before the operation begins with the following exceptions:

- An attempt to pattern test more than one track is performed and **y** is entered when asked to confirm multiple track pattern test (because only one track can be saved).
- An attempt to preserve a track's data before a destructive operation fails and a response is entered to force the operation anyway.
- The pattern test, slip sectors, or format operation has completed and then data errors occur during the restore of the track's data.
- The keys CTRL and **c** or CTRL and **b** are pressed in the middle of the operation.
- Automatic saving is disabled by using the command *toggle_save*.
- A power outage or some other unforeseen event causes the test to abort in the middle of the operation.

The fifth command, *copy_from_spu* restores data to the drive which has been saved using *copy_to_spu*. Since it writes to the disk, it can be destructive. However, the usual way this could corrupt data is if the temporary file created by *copy_to_spu* is edited before using *copy_from_spu* to restore the data.

4.5.4.2 Subtest 400, Interactive Test Invocation

To invoke the *dev5130* interactive test, use the procedure shown in the following figure. All responses in **boldface** are entered by the user. The [+> *filename*] option appends all test results to file *filename*. The prompts and responses in the following figure would appear sequentially on the screen, one line at a time. All the prompts and responses are shown in one figure for convenience.

To invoke the test, use the procedure shown in the following figure:

Figure 4–20, Subtest 400 Interactive Test Invocation Sequence

```
(spu)>sysreset; mminit -s; dshell RETURN
CONVEX DIAGNOSTIC SHELL
: test dev5130 -s 400 [+> filename] RETURN
```

4.5.4.3 Subtest 400, Interactive Test Menu

Once the test is invoked, prompts are displayed to specify various options. The following figure shows typical prompts, their possible answers (in brackets [ ]), and their default answers (in parentheses ( )). The prompts and responses in the following figure appear sequentially on the screen, one line at a time. All the prompts and responses are shown in one figure for convenience.

Figure 4–21, Subtest 400 Interactive 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
Default type (r[read only], w[rite allowed], n[o_defaults])
[w,r,n]                                          (r) ->

              PERIPHERAL CONFIGURATION DATA
     CCU     Chassis   Type      CSR     Int  Unit  Type
    -------  -------  --------  -----   ---  ----  -------
 1)  viop 3     0     DKC-204   0xc00    1     0   DKD-206
 2)  viop 3     0     DKC-204   0xa00    4     0   DKD-208
 3)  viop 6     1     DKC-203   0xc00    3     0   DKD-214

Enter device 99 to begin user-defined configurations or -1 to end selection
 2:  Device selection [-1,1-3,99]                    (-1) ->1
 3:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                           (n) ->y
 4:  Serial number (5 to 31 characters) []
                        (Default: read from drive) ->RETURN
 5:  Device selection [-1,1-3,99]                    (-1) ->2
 6:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                           (n) ->y
 7:  Serial number (5 to 31 characters) []
                        (Default: read from drive) ->RETURN
 8:  Device selection [-1,1-3,99]                    (-1) ->3
 9:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                           (n) ->y
10:  Serial number (5 to 31 characters) []
                        (Default: read from drive) ->RETURN
11:  Device selection [-1,1-3,99]                   (-1) ->RETURN
12:  Enter OK, or :NN to return to question NN [OK]
                                                   (OK) ->RETURN

*********************************************
* DATA DESTRUCTIVE OPERATION IS ABOUT TO START.*
* VERIFY WRITE-PROTECT IS SET ON ALL DRIVES    *
* EXCEPT THOSE YOU ARE TESTING/FORMATTING.     *
*********************************************

Then reconfirm that you want to continue [yn] ->y
```

As shown in the previous figure, enter **y** to the first prompt. Then select a drive to test at the `Device selection` prompt and always enter **y** to the `Previously formatted` prompt and always enter (RETURN) to the `Serial number` prompt. This drive selection process may be repeated for additional drives.

When all drives to be tested have been selected, press (RETURN) twice to select the defaults to the `Device selection` and `Enter OK` prompts. Then enter **y** to the `Then confirm` prompt to allow the test to continue. After a minute, the interactive test will start up.

4.5.4.4 Subtest 400, Interactive Test Mode

Once test initialization is complete (typically takes less than one minute), the following is displayed:

Figure 4–22, Subtest 400 Interactive Test Mode

```
                    INTERACTIVE TEST MODE
                    FOR SMD AND ESDI DRIVES

Type 'h' for help
(D1;Save)->
```

NOTE

`D1` means drive 1 is selected and `Save` means that save of track data is automatic during the destructive commands *format, pattern_test, data_verify* and *slip_sectors.*

Type **h** at any time to display the list of available commands.

To exit the interactive test, type **q** and press (RETURN). If a track is saved, a prompt asks if it is okay to lose this data. If **y** is entered or if a track is not saved, *dev5130* ends normally and exits back to the *dshell* prompt. If **n** is entered, test execution returns to the interactive prompt with nothing changed.

NOTE

When the interactive test is exited, the drives under test will ALWAYS show PASSED in the summary. This is because all failures are ignored during the interactive test.

4.5.4.5 Subtest 400, Interactive Test Commands

The following table lists all the available interactive test commands. Also a brief description of each command's purpose is listed. For the exact syntax of each command, refer to the command's description in the following section.

Table 4–12, Subtest 400 Interactive Test Commands

COMMAND	DESCRIPTION
a[lternate_seek]	Seek between two selected cylinders
copy_f[rom_spu]	Restore sectors from service processor disk to SMD
copy_t[o_spu]	Write sectors from SMD to service processor disk
da[ta_verify]	Write once, repetitively read and compare
dr[ive_select]	Select drive
du[mp_lists]	Dump defect lists to service processor disk
fo[rmat]	Reformat one track
hd[e_display]	Display header, data, and ECC
h[elp]	Display list of commands and arguments
l[ist]	Display list of flaws
pa[ttern_test]	Pattern test range of sectors
pr[otection]	Enable or disable write protection
q[uit]	Exit Interactive Test
res[tore_track_data]	Restore previously saved track data
sa[ve_track_data]	Save one track of data
sec[tor_display]	Display data from range of sectors
ser[ial_number]	Display or change disk serial number
sl[ip_sectors]	Slip one or more sectors
st[atus]	Display information about device
to[ggle_save]	Enable and disable automatic save
tr[ack_headers_display]	Display track headers
v[erify_format]	Verify a range of sectors
!*UNIX-command*	Execute UNIX command
'*comment*	Comment; ignored

Each command can be executed by entering the portion of the command that precedes the brackets or by entering all of the command. For example, the *quit* command can be executed by entering **q** or **quit**. The following sections describe the different interactive test commands.

4.5.4.6 Subtest 400, Seek Between Two Selected Cylinders

a[lternate_seek] [*nnn* **times**] *min-cyl* **to** *max-cyl*

This command provides the capability of performing alternate seeks between two cylinders so adjustments can be made to drives. Seeks are between the two specified cylinders to head. Data is not transferred during the seeks.

Arguments:

- [*nnn* **times**] — Repetition count. Possible values include:
  - **1** — Lower bound value, default value
  - **2,147,483,647** — Upper bound value
- *min-cyl* **to** *max-cyl* — The seeks occur between these two cylinders.

4.5.4.7 Subtest 400, Restore Sectors From Service Processor Disk to SMD

copy_f[rom_spu] [*filename*]

This command restores sectors previously stored to the service processor disk by the *copy_to_spu* command. If a filename is not specified, the default file created by *copy_to_spu* (filename */mnt/usr/backup_maps/save.s* where *s* is the serial number) is used. This is the file that *copy_from_spu* attempts to read if a file is not specified.

Since the service processor file is in an editable (ASCII) format, the data or the cylinder, head, and sector where it is to be restored can be edited before performing the *copy_from_spu* operation.

4.5.4.8 Subtest 400, Write Sectors From SMD to Service Processor Disk

copy_t[o_spu] [*filename*] *cyl hd sec* [**to** *cyl hd sec*]

This command allows the saving of one or more sectors to the service processor disk in an editable (ASCII) format.

The *copy_from_spu* command restores these sectors. If a filename is not specified, the default file (filename */mnt/usr/backup_maps/save.s* where *s* is the drive's serial number) is used. The save file can be edited to change the data in the sectors or to change where the sectors will be restored. If a file is chosen that already exists, this command overwrites the file.

Arguments:

- *cyl hd sec* — Starting sector
- [**to** *cyl hd sec*] — Ending sector, default is starting sector

4.5.4.9 Subtest 400, Write Once then Repetitively Read and Verify

da[**ta_verify**] [*nnn* **times**] *cyl hd sec* [**to** *cyl hd sec*] [**with** *pattern*]

This command writes to all the sectors specified with the first default pattern or the pattern specified on the command line. It then rereads the data *nnn* times (or one time by default) without data compare. Only errors detected by the drive or controller are reported. In the list of arguments, [**to** *cyl hd sec*] may be substituted with [**to end**]. Arguments:

- [*nnn* **times**]— Repetition count. Possible values include:
  - **1** — Lower bound value, default value
  - **2,147,483,647** — Upper bound value
- *cyl hd sec* — Starting sector
- [**to** *cyl hd sec*] — Ending sector, default is starting sector
- [**with** *pattern*] — The *pattern* is a 1 to 8 hexadecimal digit pattern

4.5.4.10 Subtest 400, Select Drive

dr[**ive_select**] [*drive-number*]

This command can change which drive is the active drive. Most of the other interactive test commands operate on the active drive only. The argument *drive-number* is the entry number from the *Drive Configuration Data* which is displayed during the test parameter entry. To redisplay the entry numbers enter **d** and press RETURN. The *Drive Configuration Data* is redisplayed along with which drive is currently active. Then a new drive selection prompt is displayed, and an entry number can be entered to change the active drive or pressing RETURN leaves it unchanged.

Argument:

- [*drive-number*] — The entry number from the *Drive Configuration Data* which is displayed during the test parameter entry, the active drive

The following is an example of displaying the drive configuration data and selecting a drive:

Figure 4–23, Displaying Drive Data and Selecting a Drive

```
(D1;save)-> d

                        DRIVE CONFIGURATION DATA
      CCU VME Cntlr Int Unit #   # Phys Log Prev
Drive  #   #   CSR Level #  Cyl Hds Sec Sec Fmtd Name          Serial #
----- --- --- ----- --- --- --- --- --- --- --- ------- ----------------------
   1   3   0  0xc00  1   0  760  19  60  59 yes DKD-206 03991
   2   3   0  0xa00  4   0  760  19  60  59 yes DKD-206 03268
   3   6   1  0xc00  3   0  842  20  46  45 yes DKD-208 04123

        Drive 1 is currently selected
        Enter new selection or 0 to leave it as is (0,1-3) [0] -> 2
        Drive 2 is now selected
(D2;save)-> d 3
        Drive 3 is now selected
(D3;save)->
```

The configuration data is not displayed if the drive number is entered with the command. Also, the previous figure shows that the drive number is redisplayed with each prompt as **D***x* where *x* is the current drive number.

4.5.4.11 Subtest 400, Dump Defect Lists to Service Processor Disk

du[mp_lists] [**b**][**m**|**g**|**a**]

This command is used to create a backup of the defect lists for the following reasons:

- Before moving a drive to a controller other than the Interphase 4200 SMD controller
- Whenever the disk is going to be completely overwritten for security reasons

The file can be dumped in ASCII or binary. An ASCII file is essential if the purpose is to move the drive to a Multibus system since the Multibus drive formatter can only read ASCII files to date. The binary dump capability is primarily for future use to allow a dump of the defect lists so the lists can be restored with no changes. When restoring an ASCII file, all entries become grown entries which means the information is lost as to whether the flaw originated with the manufacturer or later appeared as a grown defect. Also, a binary dump potentially occupies less room on the service processor disk.

Arguments:

- [**b**] — Dump file in binary (if not specified, dump file in ASCII)
- [**m**] — Dump only Manufacturer's Defects
- [**g**] — Dump only Grown Defects
- [**a**] — Dump all defects, default value

4.5.4.12 Subtest 400, Reformat One Track

fo[rmat] *cyl hd*

This command is used to unslip sectors, unmap a track, or recreate a bad track's headers. If automatic save is enabled (the prompt indicates *Save*), then before the track is reformatted, an attempt is made to read every logical sector (including relocated sectors). If a read fails, the options of retrying the sector, terminating the format before any destructive operation occurs, or forcing the format even though one or more sectors may not be recovered are given.

> **CAUTION**
>
> If auto-saving is disabled (*Nosave* mode) using the *toggle_save* command, the track is reformatted without preserving the track's data.

Arguments:

- *cyl hd* — Track to reformat

In the following figure, cylinder 300, head 15 is reformatted on an NEC drive (760 cylinders, 19 heads, 59 logical sectors). The track contains one slipped sector.

Figure 4–24, Restoring Track Data

```
(D3;Save)-> format 300 15
        No data has been previously saved. Saving this track
        Data save operation complete

 TRACK HEADERS BEFORE FORMAT:
48 49 50 51 52 53 54 55 56 57 58  0  1  2  3 *B  4  5  6  7  8  9 10 11 12
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47

        About to format cylinder 300  head 15.
        Flawed sectors associated with this track:
            Manufacturer's Defect list: 4
                These flaws will be reapplied to the track since this list is
                not ignored. To ignore manufacturer's defects, you must
                reformat the drive and change format options.
            Grown Defect list:          none
        Format complete.
        Restoring data to track
        Data restore operation complete

(D3;Save)-> track 300 15
 cylinder = 300  head = 15
 48 49 50 51 52 53 54 55 56 57 58  0  1  2  3 *B  4  5  6  7  8  9 10 11 12
 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
 45 46 47 48 49 50 51 52 53 54
(D3;Save)->
```

CAUTION

If auto-save is enabled and sectors are unslipped by the *format* command, data is restored to the bad sectors which could cause the data to be no longer readable. This is okay since the data is still saved. Slip the bad sectors using the *slip_sectors* command and then use the *restore_track_data* command to restore the track's data again. Also before the reformat, a prompt is displayed to enter which of the grown defects to reslip which allows the reslip to be performed in the same operation.

NOTE

If Manufacturer's Defects have been slipped on a track that is to be reformatted, the defects are reslipped automatically after the format. The only way to unslip Manufacturer's Defects is to reformat the whole disk and specify at format setup time (Subtest 300, Format Setup) that the Manufacturer's Defects are to be ignored by using the *format_options* command.

The following is another example where the entire track has been previously mapped to an alternate track because there were two grown defects (sectors 2 and 3) on the track and only one spare sector was available. The alternate track that was used is cylinder 757 head 13.

Figure 4–25, Restoring Data After Testing and Slipping Tracks

```
(D3;Save)-> format 422  1
         No data has been previously saved. Saving this track
         Data save operation complete

  TRACK HEADERS BEFORE FORMAT:
 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
 13 13 13 13 13 13 13 13 13 13

         About to format cylinder 422  head  1.
         Flawed sectors associated with this track:
             Manufacturer's Defect list: none
             Grown Defect list:          2 3
             Enter sectors from GD list you want reslipped
               separated by spaces: 2
          Track is currently mapped to cyl 757 hd 13.
          Do you want this alternate to be marked unusable [yn] (n) -> y
          Format complete.
          Restoring data to track
          Data restore operation complete

(D3;Save)-> track 422  1
 cylinder = 422  head =  1
 55 56 57 58  0  1 *B  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 45 46 47 48 49 50 51
 52 53 54 48 49 50 51 52 53 54
(D3;Save)->
```

When the format is complete, the only sector still bad is sector 2 and the track is no longer mapped to an alternate track.

4.5.4.13 Subtest 400, Display Header, Data and ECC

hd[e_display] *cyl hd sec* [**to** *cyl hd sec*]

This command displays the sector data and the header and Error Correction Code (ECC). The data is read in raw mode which means the controller does not perform error checking of the data, so an ECC error is not displayed while performing this operation.

Arguments:

- *cyl hd sec* — Starting sector.
- [**to** *cyl hd sec*]— Ending sector, default is starting sector.

> **NOTE**
>
> The sector specified for the *hde_display* command is the physical sector numbered from index. All other commands search all headers on the track until the header is found.

To illustrate how the *hde_display* command works, the track headers displayed by the *track* command and the sector displayed by the *hde_display* command are shown in the following figure:

Figure 4–26, Displaying the Track Headers

```
(D2;Save)-> track 27 5
 cylinder =  27  head =  5
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 *S  0
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 26 27 28 29 30 31 32 33 34 35

(D2;Save)-> hde 27 5 0

Phys:  27  5   0  Log:  27  5  36       Header:a81b0524 05240000 ECC:2bc18e26
000  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
020  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
040  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
060  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
080  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0a0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0c0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0e0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
100  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
120  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
140  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
160  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
180  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1a0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1c0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1e0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6

(D2;Save)->
```

The previous figure shows that the physical sector 0 corresponds to logical sector 36 on this track.

The following figure shows the result of an attempt to display header, data, and ECC for track that has been mapped to an alternate track:

Figure 4–27, Displaying a Mapped Track

```
(D2;Save)-> track 286 5
 cylinder = 286  head =  5
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12

(D2;Save)-> hde 286 5 0

Phys: 286  5   0  Log: TRK MAPPED to  757 12    Header:d11e0512 02f50000
```

The following figure shows the result of an attempt to display header, data, and ECC of an alternate track. Specifically, the figure shows the header display (*hde*) of cylinder 757 head 12 physical sector 0 from the previous figure where cylinder 286 head 5 has been mapped to cylinder 757 head 12 physical sector 0.

Figure 4–28, Displaying an Alternate Track

```
(D2;Save)-> track 757 12
 cylinder = 757  head = 12
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 *S  0
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 26 27 28 29 30 31 32 33 34 35

(D2;Save)-> hde 757 12 0

Phys: 757 12   0  Log: 286  5  36       Header:a91e0524 05240000
                       This appears to be an alternate track
```

4.5.4.14 Subtest 400, Display List of Commands and Arguments

h[elp]

This command lists the commands and their arguments.

4.5.4.15 Subtest 400, List Flaws

l[ist] [m|g|a]

This command lists flaws which have been fixed on the drive. The types of flaws to be listed can be selected as follows:

- [**m**] — List Manufacturer's Defects entries only
- [**g**] — List Grown Defects entries only
- [**a**] — List Manufacturer's Defects and Grown Defects entries, default value

Figure 4–29, Example of the List Command

```
format setup (D1;Sector)-> list

Summary of flaws on drive 1 serial number 003013
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR  SRC
  17  7  23  35032  4825      NONE USR | 237  0   1   4816    17  MAP  HNF  MAP
 237  0   0   4802   105  MAP HARD MAP |
```

The listing above is double-columned and shows the location of all known flaws for this drive. The fields in each entry have the following meanings:

CYL HD SEC	The cylinder, head and sector impacted by the flaw
BCAI	Byte count after index to the start of the flaw
LEN	Length of the flaw in bits
TYPE	MAP or blank. If MAP, the entire track has been mapped to an alternate track
ERR	The actual error that was detected during pattern test:

Table 4–13, Pattern Test Flaws

POSSIBLE VALUES	ERROR CODE	MEANING
CSUM	0x19	Header checksum error
HPAD	0x22	Invalid header pad
SOFT	0x13	Soft ECC error
HARD	0x23	Hard ECC error
HNF	0x29	Sector not found
SYNC	0x2b	Invalid sync in data field
JHDR	0x6a	Unrecognized header field
MHDR	0x6b	Mapped header error
MISC		Any other error
NONE		No error detected during pattern test

SRC Where error originated from:

Table 4–14, Sources of Flaws

POSSIBLE VALUES	MEANING
MAP	Came from original Manufacturer's Defect Map
PGM	Discovered during pattern test (Subtest 301)
USR	User input this flaw

4.5.4.16 Subtest 400, Pattern Test Range of Sectors

p[attern test] [*nnn* **times**] *cyl hd sec* [**to** *cyl hd sec*] [**with** *pat1* [*pat2* ...]]

This command allows the pattern test of selected sectors repetitively. It is useful to check part or all of a system portion of a disk (all cylinders before the diagnostic cylinder and relocated sectors). If the test is constrained to one track, the data is automatically saved and restored when the test is completed. If more than one track is selected, a warning prompt is displayed and the operation can be terminated at this point to prevent permanent overwrite of data. The automatic save can be disabled with the *toggle_save* command. Up to 15 patterns can be specified. In the list of arguments, [**to** *cyl hd sec*] may be substituted with [**to** *end*].

Arguments:

- [*nnn* **times**]— Repetition count. Possible responses include:
  - **1** — Lower bound value, default value.
  - **2,147,483,647** — Upper bound value.
- *cyl hd sec* — Starting sector.
- [**to** *cyl hd sec*]— Ending sector, default is starting sector.
- [**with** *pat1* [*pat2* ...]] — Up to 15 1-8 digit hex patterns can be specified, default is the patterns specified in Subtest 301, Format and Pattern Test or the patterns in the file *dev5130.patt* if this file exists.

The following figure shows the use of the pattern test command:

Figure 4–30, Example of Pattern Test Command

```
(D3;Save)-> patt 10 times 0 0 0 to 1 0 0 with 00000000 ffffffff
     Patterns used: 00000000 ffffffff
```

4.5.4.17 Subtest 400, Enable or Disable Write Protection

pr[otection] [**on** | **off**]

This command enables or disables writes to protected areas.

> **CAUTION**
>
> This command should be used with caution. If protection is turned off, writes can be made anywhere on the disk and can destroy unrecoverable information. If protection is on, writes are only allowed to the alternate track area or the used tracks of the defect cylinder.

On ESDI drives, write are also not allowed to the innermost cylinder which contains the original Manufacturer's Defect map.

4.5.4.18 Subtest 400, Exit Interactive Test

q[uit]

This command is used to exit the interactive test.

4.5.4.19 Subtest 400, Restore Previously Saved Track Data

r[estore_track_data]

This command can restore data from the general buffer to the track it was originally read from. The *save_track_data* command stores data into the general buffer from a selected track. In addition, the *format*, *data_verify*, and *pattern_test* commands save into the general buffer before their operation and automatically restore the data afterward. The data remains in the buffer so it can be restored again at a later time if, for instance, the first restore is bad.

If data is saved and is not restored and then an attempt is made to *quit* the interactive test, a message is displayed indicating that there is saved data and a prompt is displayed that allows reentering or exiting the interactive test.

The following figure shows an example of using the restore command:

Figure 4–31, Restoring Track Data

```
(D3;Save)-> restore
        Data has been saved for drive 3: cylinder 273  head 17
        Do you really want to restore this data? [yn] -> y
        Data restore operation complete
(D3;Save)->
```

If data is not restored, the operation is terminated and the interactive prompt is displayed.

If the current drive has been changed from drive 3 to drive 9 since the last save operation and the restore command is used, the following is displayed:

Figure 4–32, Changing Drives After a Save Operation

```
(D9;Save)-> restore
        Data has been saved for drive 3: cylinder 273  head 17
        However, drive 9 is currently selected.
        Use 'd[rive_select]' to change drives and try again.
(D9;Save)->
```

If data has not been saved before attempting to restore the data, the following is displayed:

Figure 4–33, Attempting to Restore Data Before Saving

```
        No data has been previously saved so restore aborted.
(D9;Save)->
```

4.5.4.20 Subtest 400, Save One Track of Data

sa[ve_track_data] *cyl hd*

This command saves one track of data into the general buffer. The data can be restored with the *restore_track_data* command. Data on relocated sectors is also saved so if a track is reformatted (which removes the relocated sectors) all the track's data can still be restored.

Arguments:

- *cyl hd* — Track to be saved

The following is an example of saving one track:

Figure 4–34, Saving One Track

```
(D4;Nosave)-> save 525 5
        Data has already been saved for drive 4: cyl=601, hd=14
        Are you sure you want to overwrite this data? [yn] -> y
        Data save operation complete
(D4;Nosave)->
```

The prompt in the previous figure specifies *Nosave* which indicates that the command *toggle_save* was previously entered so that saves are not automatic on destructive operations. But, *Nosave*

does not effect this command. Normally, *Save* is displayed since disabling automatic save can cause loss of data.

4.5.4.21 Subtest 400, Display Data From Range of Sectors

sec[tor_display] *cyl hd sec* [**to** *cyl hd sec*]

This command displays the data from one or more consecutive sectors.

Arguments:

- *cyl hd sec* — Starting sector.
- [**to** *cyl hd sec*] — Ending sector, default is starting sector.

The following is an example of using the *Sector_display command*:

Figure 4–35, Displaying Sector Data

```
(D1;Save)-> sector 27 5 55

        Sector  27  5 55:
000  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
020  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
040  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
060  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
080  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0a0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0c0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
0e0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
100  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
120  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
140  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
160  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
180  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1a0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1c0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6
1e0  6db66db6 6db66db6 6db66db6 6db66db6  6db66db6 6db66db6 6db66db6 6db66db6

(D1;Save)->
```

The previous example is almost identical to the one for the *hde_display* command. The major difference between the two commands is that with the *sector_display* command, the read is sensitive to data errors. In the raw read mode that *hde_display* uses, no data errors are reported. If a data error does occur while executing the *sector_display* command, then an error message is displayed, and a prompt is displayed to retry, force, or skip the read.

If a retry of the read is specified, the number of read attempts to try before displaying another error message can be specified. From 1 to 2,147,483,647 retries can be specified. Every five retries a seek is alternately performed one cylinder in and back or one cylinder out and back. Any successful read terminates the retries and the data is displayed.

If a force of the error is specified, one more read is performed with a *move bad data* indication to the controller. Whatever is read on this attempt is displayed. An error indication is not displayed if an error occurs on this read.

If a fix of the error is specified, Error Correction Code (ECC) correction is applied if the error is correctable.

If *skip* is specified, the sector is skipped. If there are any more sectors to read in the range specified, then the display of data continues with the next sector.

The following is an example of a read that fails:

Figure 4–36, Failed Read Example

```
(D1;Save)-> sector 400 22 18 to 400 22 19

dev5130 drv 1 ERROR 1e(cntlr) Soft ECC Error
 cyl:400 hd:22 sec:18

                        1 attempts to read sector failed
                        what to do [# of retries/force/skip/fix] (1) -> 100

dev5130 drv 1 ERROR 1e(cntlr) Soft ECC Error
 cyl:400 hd:22 sec:18

                        100 attempts to read sector failed
                        what to do [# of retries/force/skip/fix] (100) -> skip


        Sector 400 22 19
000  00000000 00000000 00000000 ff300020  1023eee10 99903220 b2d3d4d1 ab222290
020                     ---------------- etc ----------------
```

In the previous figure sector 18 is not displayed since it was skipped. What is displayed is the next sector, sector 19 which was read successfully on the first attempt. If `fix` had been specified for sector 18, it would have been displayed since soft ECC errors were occurring which are correctable.

4.5.4.22 Subtest 400, Display or Change Disk Serial Number

ser[ial_number] [*number*]

This command displays the drive's serial number. To change the serial number, follow the command with the new serial number.

4.5.4.23 Subtest 400, Slip One or More Sectors

sl[ip_sectors]

This command is used to slip sectors or map tracks which are generating errors or to enter defect map information to slip sectors. If track data is automatically saved (indicated by the word *Save* in the prompt), an attempt to save and restore all affected tracks is made to minimize data loss unless automatic save is disabled with the *toggle_save* command.

> **NOTE**
>
> When entering the defect map for a new disk, a disk that has just been formatted and does not yet contain a file system, the automatic save may be disabled with the *toggle_save* command before entering the *slip_sectors* command. This causes the slip to proceed much faster.

> **CAUTION**
>
> If any usable data exists on these tracks, the data is lost.

The *slip_sectors* command starts up a separate command processor which expects bad sectors or defect map information to be entered. There is a help facility which lists the available commands if **h** is entered. The help screen is shown as follows:

Figure 4–37, *Slip_Sectors* Help Screen

```
SLIP_SECTORS commands:
q[uit]                          - exit from 'slip_sectors'
c[hange_mode]                   - toggles logical sector/defect
                                  map entry mode
cyl hd sec                      - logical sector to be slipped
cyl hd bcai len                 - defect map entry to be slipped
de[lete] cyl hd sec             - delete logical sector entry
de[lete] cyl hd bcai len        - delete defect map entry
e[xecute]                       - slip the sectors now
f[ile] filename                 - get inputs from specified file
h[elp]                          - display this information
l[ist]  [m|g|a|n]               - list inputs made so far
m[ap_track] cyl hd              - map track to an alternate
!unix-command                   - execute unix command
'comment                        - comment; ignored
```

To slip sectors which are generating errors, the normal usage is to enter all sectors which need to be slipped, verify the entries using the *list* command, delete any incorrect entries with the *delete* command, and then begin the slipping with the command *execute*. New sectors can be entered at any time up until the *execute* command is entered. As a safeguard, when the *execute* command is

typed, the list of sectors is automatically displayed and a prompt is displayed to confirm the list. By replying **n** to the confirmation, the *slip_sectors* prompt is displayed with all inputs still intact.

4.5.4.24 *slip_sectors' change_mode* Command

c[hange_mode]

The *change_mode* command is used to toggle between the logical sector and defect map entry mode. The current input mode (logical sector or defect map mode) is revealed in the slip prompt as shown in the following figure:

Figure 4–38, Changing Entry Modes

```
slip (D1;Save;Sector)->c
slip (D1;Save;Defect)->
```

4.5.4.25 *slip_sectors' cyl hd sec* and *cyl hd bcai len* Commands

cyl hd sec

or:

cyl hd bcai len

These commands are used to enter the logical sector to be slipped or to enter the defect map entry to be slipped. When a defect is entered, the associated track's headers are listed as shown in the following figure:

Figure 4–39, Entering a Defect

```
slip (D1;Save;Defect)-> 293 1 572 9510
       Associated logical sectors: 55 56 57
```

The previous figure illustrates that three sectors are affected by the media flaw which starts at byte 572 from index and is 9510 bits long.

If two defects exist that affect the same sector, the logical sector is only entered in the list of flaws once. Subsequent entries of the same sector are ignored and an appropriate warning is displayed.

For instance, if the following two defects exist:

cylinder	head	bcai	bit-length
10	0	70	1
10	0	80	1

Both defects fall in sector 0 so when the second defect is entered, it is not slipped again. The slipping of sectors with two defects is shown in the following figure:

Figure 4–40, Slipping Sectors With Two Defects

```
slip (D1;Save;Defect)-> 10 0 70 1
       Associated logical sectors: 0
slip (D1;Save;Defect)-> 10 0 80 1
                        Cyl 10 Hd 0 Sec 0 already exists. Duplicate entry not created.
slip (D1;Save;Defect)-> list

Summary of new flaws on drive 1 serial number 02039
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
  10  0   0     70     1       NONE NEW |
```

NOTE

In the current version of *dev5130*, no bytes are considered part of a gap. A flaw in any byte *results* in one or more sectors being slipped.

4.5.4.26 *slip_sectors' delete* Command

de[**lete**] *cyl hd sec*

or:

de[**lete**] *cyl hd bcai len*

The *delete* commands listed above are used to delete logical sector entries or to delete defect map entries. The following figure shows how to delete bad inputs. The *list* command is used after the *delete* command to verify that the bad inputs were deleted.

Figure 4–41, Deleting Bad Inputs

```
(D1;Save)-> slip
Type 'h' for help with slip commands
slip (D1;Save;Sector)-> 17 7 23
slip (D1;Save;Sector)-> 17 7 46
slip (D1;Save;Sector)-> d 17 7 46
        Sector  17  7 46 has been removed from the inputs
slip (D1;Save;Sector)-> list

Summary of new flaws on drive 1 serial number 02039
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
  17  7  23                   NONE USR |
```

4.5.4.27 *slip_sectors' execute* Command

e[xecute]

The *execute* command is used slip the sectors specified. When the slip is executed, the sectors are slipped one track at a time or the track is mapped if not enough spare sectors exist. If automatic save is active, the data for the track is copied to a special buffer (not the general buffer used by *save_track_data*) before the slip and is restored after the slip.

To verify the slip, display the track headers once more using the *track_headers_display* command. All slipped sectors are marked with a **B*. Mapped tracks have the same number for all sectors. This number is the head number of the alternate track. An example of displaying the track headers after slipping sector 255 0 6 is shown in the following figure:

Figure 4–42, Display Slipped Track Headers

```
(D1;Save)-> track 255 0
cylinder = 255  head =  0
 0  1  2  3  4  5 *B  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56 57 58
```

If defect input mode is activated and an attempt is made to slip the sector on track 255 head 0 which has already been marked bad in the previous figure, the following figure shows what would be displayed:

Figure 4–43, Slipped Tracks Previously Marked Bad

```
slip (D1;Save;Defect)-> 255 0 3800 1
        Cyl 255 Hd 0 Sec 6 already exists. Duplicate entry not created.
```

4.5.4.28 *slip_sectors' file* Command

f[ile] *filename*

The file command is used to get inputs from a specified file. To input defects from a file, first create the file as documented in Subtest 300, Format Setup. Then type the command **file** *filename* where *filename* represents the actual filename. The defects in this file can be in byte-count-after-index or logical sector format. The file can also contain track entries which specify tracks to be mapped to alternate tracks.

4.5.4.29 *slip_sectors' help* Command

h[elp]

This command lists the commands and their arguments.

4.5.4.30 *slip_sectors' list* Command

l[ist] [**m**|**g**|**a**|**n**]

This command lists flaws which have been fixed on the drive. The types of flaws to be listed can be selected as follows:

- [**m**] — List Manufacturer's Defects entries only
- [**g**] — List Grown Defects entries only
- [**a**] — List Manufacturer's Defects and Grown Defects entries, default value
- [**n**] — List defect needs

4.5.4.31 *slip_sectors' map_track* Command

m[ap_track] *cyl hd*

To map an entire bad track, use the *map_track cyl hd* command. The test selects one of the alternate tracks for use. If data is being saved, then the original track's data is restored to the alternate track.

4.5.4.32 *slip_sectors' quit* Command

q[uit]

To exit the *slip_sectors* command at any time enter the *quit* command.

4.5.4.33 *slip_sectors'* Execute UNIX Command

!UNIX-command

This command is used to issue UNIX commands. After the UNIX command is completed, execution is returned to the interactive prompt. UNIX commands can be issued from any prompt throughout the test.

4.5.4.34 *slip_sectors'* Enter Comments

'comment

This command is used to enter a comment. To enter a comment, type the ' command and a comment. The comment is echoed so it is redirected output. This is useful to document redirected output.

4.5.4.35 *slip_sectors* Command Examples

The following figure contains examples of several *slip_sectors* commands.

Figure 4–44, Command Examples

```
slip (D1;Save;Sector)-> 17 7 36
slip (D1;Save;Sector)-> 247 12 4
slip (D1;Save;Sector)-> 256 0 12
slip (D1;Save;Sector)-> d 256 0 12
slip (D1;Save;Sector)-> 255 0 12
slip (D1;Save;Sector)-> list

Summary of new flaws on drive 1 serial number 02039
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
  17  7  23  35032  4825  MAP NONE USR | 247 12   4   2432  4825      NONE USR
  17  7  36   6644  4825  MAP NONE USR | 255  0  12   7904  4825  MAP NONE USR

slip (D1;Save;Sector)-> e

Summary of new flaws on drive 1 serial number 02039
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
  17  7  23                  NONE USR | 247 12   4                   NONE USR
  17  7  36                  NONE USR | 255  0  12                   NONE USR

       Are you sure inputs are ready for slipping? [yn] -> y

Number of spares per track: 1
 cyl hd sec  bcai  len  action              additional information
---- -- --- ----- ----- ------- ---------------------------------------------
  17  7  23 35032  4825 mapped  Alternate track used: cylinder 757 head 13
  17  7  36  6644  4825 mapped  Alternate track used: cylinder 757 head 13
 247 12   4  2432  4825

dev5130 drv 1 ERROR 23(cntlr) Hard ECC error
 cyl:247 hd:12 sec:4

                     1 attempts to read sector failed
                     what to do [# of retries/force/skip/fix] (1) -> fix
Number of spares per track: 1
 cyl hd sec  bcai  len  action              additional information
---- -- --- ----- ----- ------- ---------------------------------------------
 247 12   4  2432  4825 slipped
 255  0  12  7904  4825 mapped  Alternate track used: cylinder 757 head 12

slip (D1;Save;Sector)-> q
```

In the previous figure, the *cyl hd sec*, *delete*, *list*, *execute*, and the *quit* commands are displayed. Sector 247 12 4 is difficult to read (hard ECC error). After the first attempt fails, a request is made to fix the error. The next attempt has either a soft Error Correction Code (ECC) error or no error is reported. If an error is not fixable, specifying *force* results in one more read of the sector with the controller set to move bad data. If *skip* is selected, the remaining slips are still performed. Retries are allowed by entering the number of retries or by pressing RETURN to use the default number of retries. Every five retries a seek is alternately performed one cylinder in and back or one cylinder out and back.

In the following figure the track headers are displayed again to verify the slips:

Figure 4–45, Displaying the Track Headers

```
(D1;Save)-> track 17 7
cylinder =  17  head =  7
 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
 13 13 13 13 13 13 13 13 13 13

(D1;Save)-> track 247 12
cylinder = 247  head = 12
  0  1  2  3 *B  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58

(D1;Save)-> track 255 0
cylinder = 255  head =  0
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12
```

In the previous figure Track 17 7 and track 255 0 have been relocated to head 13 and head 12 respectively of the relocation area. The cylinder is not known but can be determined by using the *hde_display* command on any sector of the bad track. Track 247 12 has one bad sector and no spare sector.

In the following figure all flaws are redisplayed to verify the relocation actually occurred:

Figure 4–46, Display All Flaws to Verify Relocation

```
(D1;Save)-> list

BAD BLOCK TABLE FOR ccu/mb/csr/d=3/0/3f0/0

Summary of new flaws on drive 1 serial number 02039
 CYL HD SEC   BCAI   LEN TYPE  ERR SRC | CYL HD SEC   BCAI   LEN TYPE  ERR SRC
  17  7  23  35032  4825  MAP NONE USR | 247 12   4   2432  4825       NONE USR
  17  7  36   6644  4825  MAP NONE USR | 255  0  12   7904  4825  MAP NONE USR

(D1;Save)->
```

If there had already been entries in the table, they would have been displayed in the previous figure also. This list of the flaws actually comes from a copy of the table kept in main memory. However, it does reflect what is on the disk since both the *slip_sectors* and *format* commands rewrite all five copies of the defect lists before returning to the interactive prompt. To reread them off the disk before listing them, use the *status* command. It rereads all lists before displaying the status of the lists.

One other verification to perform is a *verify_format* on each of the tracks.

4.5.4.36 Subtest 400, Display Information About Device

st[atus]

This command lists the drive configuration data (all selected drives) and displays which drive is the currently active drive. The *Status* command also displays which track was last saved in the general buffer by using one of the commands: *format*, *save_track_data*, *data_verify*, or *pattern_test*. In addition, a line is displayed which indicates whether the automatic save of track data is enabled (*Save* is also shown in the prompt). If the drive is previously formatted, then the following information is also displayed:

a. The status of all copies of defect lists and track logs on the disk.

b. The revision of the formatter that originally formatted the disk.

c. The revision of the formatter that last reformatted the disk.

d. The date the drive was originally formatted.

e. The date the drive was last reformatted.

f. The serial number of the drive.

This information is from the defect list cylinder which is reread each time the *status* command is entered. The following figure shows the information that is displayed when the *status* command is entered:

Figure 4–47, Displaying Device Information

```
(D1;Save)-> status

                        DRIVE CONFIGURATION DATA
      CCU VME Cntlr Int Unit #   # Phys Log Prev
Drive  #   #   CSR Level #  Cyl Hds Sec Sec Fmtd Name          Serial #
----- --- --- ----- --- --- --- --- --- --- --- ------- ----------------------
   1   3   0  0xc00  1   0  760  19  60  59 yes DKD-206 03991
   2   3   0  0xa00  4   0  760  19  60  59 yes DKD-206 03268
   3   6   1  0xc00  3   0  842  20  46  45 yes DKD-208 04123

Drive 1 is currently selected
Saved Track:  drive 3  cylinder 300 head 12
Save of track data is automatic during destructive commands.
Status of defect lists:
Drive  1 All copies are GOOD
Drive information
        Serial number: 03991
        Original Formatter Version: 3.00 Date: Mon Mar 14 17:33:24 CST 1988
        Latest Disk Update Version: 3.00 Date: Tue Mar 15 10:23:14 CST 1988
        Format options: Manufacturer's defect list was used if available
                        Pattern test was completed using pattern set 2

(D1;Save)->
```

4.5.4.37 Subtest 400, Enable and Disable of Automatic Save

to[*ggle_save*]

This command switches between enabling and disabling of automatic save. Automatic save occurs during the commands *format*, *pattern_test*, *data_verify* and *slip_sectors*. The displayed interactive prompt indicates whether automatic save is enabled (`Save`) or disabled (`Nosave`). The *Status* command also displays the current setting of automatic save.

4.5.4.38 Subtest 400, Display Track Headers

tr[**ack_headers_display**] *cyl hd* [**to** *cyl hd*]

This command displays the sector numbers for each of the sectors on a track. The sector number that is displayed first is located physically just after the index so it will not always be zero. This is because sectors may be skewed and interleaved. For a skew of 5, head 0 logical sector 0 is the first sector after index. On head 1, logical sector 0 is the sixth sector, on head 2, logical sector 0 is the eleventh sector and so on.

Arguments:

- *cyl hd* — Starting sector.
- [**to** *cyl hd*] — Ending sector, default is starting sector.

Each track has one spare sector (unless it is used because one or more sectors were slipped on the track). The spare sector is marked *S. If any sectors have been flagged as bad, they are marked *B.

If the entire track has been remapped to an alternate track, then the head number of the alternate track is displayed. To find out what alternate track is being used, use the *hde_display* command.

The following figure shows the display of sector numbers for sectors and a ***S** for a spare sector on a track:

Figure 4–48, Displaying Sector Numbers and Spare Sector

```
(D1;Save)-> track 427 2
cylinder = 427  head =  2
50 51 52 53 54 55 56 57 58 *S  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
(D1;Save)->
```

In the previous figure, sector 0 is actually the eleventh sector on the track since it is head 2 with a skew of 5. Also the spare sector (*S) is just before sector 0.

The following figure shows the display of sector numbers for sectors and a ***B** for a bad sector on a track:

Figure 4–49, Displaying Sector Numbers and Bad Sector

```
(D1;Save)-> track 208 11
cylinder = 208  head = 11

 5 *B  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
54 55 56 57 58  0  1  2  3  4

(D1;Save)->
```

In the previous figure, one sector has been flagged bad and the spare sector has been used. Also sector zero is skewed almost to the end of the track because it is head 11 with a skew of 5.

4.5.4.39 Subtest 400, Verify a Range of Sectors

v[**erify_format**] [*nnn* **times**] *cyl hd sec* [**to** *cyl hd sec*]

This command performs repeated reads on a range of sectors in an attempt to detect read errors. The command is not data destructive. It does not perform a data compare since the data is unknown. As the *verify* is executed, a pass count is incremented on the display to indicate how far the execution has progressed. Any errors are displayed without retry and the reads continue. If any sectors have been relocated, the relocated sectors are read. In the list of arguments, [**to** *cyl hd sec*] may be substituted with [**to end**].

Arguments:

- [*nnn* **times**]— Repetition count. Possible responses include:
  - **1** — Lower bound value, default value
  - **2,147,483,647** — Upper bound value
- *cyl hd sec* — Starting sector
- [**to** *cyl hd sec*]— Ending sector, default is starting sector

4.5.4.40 Subtest 400, Execute UNIX Command

!*UNIX-command*

This command is used to issue UNIX commands. After the UNIX command is completed, execution is returned to the interactive prompt. UNIX commands can be issued from any prompt throughout the test.

The following figure shows an example of entering the **!** command and the UNIX command **ps**:

Figure 4–50, Executing UNIX Command From Interactive Test Prompt

```
(D1;Save)-> !ps
   PID TTY TIME CMD
     2 co  0:08 -
   273 co  0:00 dshell
   274 co  0:09 dev5130.t -c 4
   281 co  0:00 sh -c ps?
   282 co  0:02 ps
(D1;Save)->
```

4.5.4.41 Subtest 400, Enter Comments

'*comment*

This command is used to enter a comment. To enter a comment, type the **'** command and a comment. The comment is echoed so it is redirected output. This is useful to document redirected output.

4.6 Examples of Typical Usage of This Test

This section is designed for the casual user who has a general knowledge of disk drives and wants to do some basic operations on disks such as formatting, verifying previously formatted disks, and slipping a sector when the sector is reported bad by CONVEX UNIX. The following examples assume the drives are already cabled up, the VIOP is working, and the switches and jumpers on the controllers and drives are set properly.

4.6.1 Formatting One or More Disks at the Same Time

Up to 12 drives can be formatted at the same time while CONVEX UNIX is not running.

> **WARNING**
>
> CONVEX UNIX cannot be running concurrently with the *dev5130* test.

This example assumes all the drives selected have not been previously formatted. Therefore, the manufacturer's original defect map is read from the drive during Subtest 300, Format Setup. If any of the drives are previously formatted on a CONVEX system with a VME controller, then **y** should be entered to the `Previously formatted` prompt and (RETURN) should be entered to `Serial number` prompt so the serial number is read from the drive. If an incorrect reply is entered to the `Previously formatted` prompt, the test detects the error in Subtest 300, Format Setup and then the reply can be changed.

> **NOTE**
>
> Previously formatted means that Subtest 300, Format Setup has been completed; not that all of Class 3 subtests have been completed.

The sequence of commands to invoke the formatter are shown in the following figure:

Figure 4–51, Formatter Invocation Sequence

```
(spu)> sysreset; mminit -s; dshell (RETURN)
CONVEX DIAGNOSTIC SHELL
: test dev5130 -c 3 (RETURN)
```

The following figure shows the prompts and responses to format three drives which have never been formatted before.

Figure 4–52, Formatting Three Unformatted Drives

```
                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
 1:  Use defaults except allow destructive writes [y,n]
                                                  (y) -> RETURN

              PERIPHERAL CONFIGURATION DATA
    CCU       Chassis   Type      CSR     Int  Unit  Type
   --------  -------  --------  -----  ---  ----  -------
 1) viop 7      0      DKC-204   0xa00   1    0    DKD-206
 2) viop 7      0      DKC-204   0xc00   4    0    DKD-208
 3) viop 7      1      DKC-203   0xa00   3    0    DKD-214
Enter device 99 to begin user-defined configurations or -1 to end selection
 2:  Device selection [-1,1-3,99]                (-1) -> 1
 3:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> RETURN
 4:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> 03658
 5:  Device selection [-1,1-3,99]                (-1) -> 2
 6:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> RETURN
 7:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> 04012
 8:  Device selection [-1,1-3,99]                (-1) -> 3
 9:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> RETURN
10:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> 14539
11:  Device selection [-1,1-3,99]                (-1) -> RETURN
12:  Enter OK, or :NN to return to question NN [OK]
                                                 (OK) -> RETURN

     ------- summary of inputs prints here -------

*********************************************
* DATA DESTRUCTIVE OPERATION IS ABOUT TO START.*
* VERIFY WRITE-PROTECT IS SET ON ALL DRIVES    *
* EXCEPT THOSE YOU ARE TESTING/FORMATTING.     *
*********************************************

Then reconfirm that you want to continue [yn] -> y
```

Figure 4–52, Formatting Three Unformatted Drives (continued)

```
        Subtest 300  0:00:00 Prepare for format

Type 'h' for help with commands
format setup (D1;Sector)->continue

                     0:07:50 Reading Manufacturer's Defect Maps
        Preparing cylinder for defect lists on drive(s) 1 2 3
                     0:08:30  passed
        Subtest 301  1:40:00  passed
        Subtest 302  0:00:00  Fix flaws

Summary of all flaws on drive 1 serial number 03658:
 CYL HD SEC  BCAI   LEN TYPE  ERR SRC | CYL HD SEC  BCAI   LEN TYPE  ERR SRC
   ----------- list of all flaws for this drive prints here -----------

Summary of all flaws on drive 2 serial number 04012:
 CYL HD SEC  BCAI   LEN TYPE  ERR SRC | CYL HD SEC  BCAI   LEN TYPE  ERR SRC
   ----------- list of all flaws for this drive prints here -----------

Summary of all flaws on drive 3 serial number 14539:
 CYL HD SEC  BCAI   LEN TYPE  ERR SRC | CYL HD SEC  BCAI   LEN TYPE  ERR SRC
   ----------- list of all flaws for this drive prints here -----------

                     0:03:00  passed
        Subtest 303  0:11:00  passed
        Subtest 304  0:08:00  passed
        Subtest 305  0:00:50  passed
        Subtest 306  0:00:05  passed

************** Test started Wed Mar 16 08:03:52 1988
************** Test ended   Wed Mar 16 10:23:46 1988

test 'dev5130.t' passed
```

Approximate format times for current CONVEX drives are listed in the following table:

Table 4–15, Drive Format Times

DRIVE	TIME (hr:min)
NEC 2352 (1/2-GByte)	1:30
NEC 2363 (1-GByte)	2:30
Hitachi 514-38	1:20

If one NEC 2352 and one NEC 2363 are being formatted, the format of the drives will take 2.5 hours plus 5 minutes (5 minutes for each additional drive after the first drive). The NEC 2352 will not finish in 1.5 hours in this case because each subtest must complete on all drives before the next subtest begins. The NEC 2352 will finish each subtest before the NEC 2363 but must wait for the NEC 2363 to complete the subtest.

Subtest 306, Verify Pattern Test Error Threshold will fail if any flaws are found on a drive during the pattern test which were not in the existing defects lists. This is not normally a significant concern unless several new flaws are occurring. Inspect the list printed during Subtest 302, Fix Flaws and see how many entries are marked PGM instead of USER or MAP.

4.6.2 Verifying the Format of One or More Drives

When a disk or set of disks appear to have read problems, verify that the problem repeats by reading all the usable parts of each disk. Then cables or controllers or drives can be changed to verify the disks again. To verify the usable parts of each disk execute Subtest 304, Verify System Format. This subtest performs a read-only operation to locate any unfixed media flaws or to detect problems with the hardware.

The sequence of commands to invoke Subtest 304, Verify System Format are shown in the following figure:

Figure 4–53, Verify Format Invocation Sequence

```
(spu)> sysreset; mminit -s; dshell (RETURN)

CONVEX DIAGNOSTIC SHELL

: test dev5130 -s 304 (RETURN)
```

Upon invocation of Subtest 304, the prompts in the following figure are displayed. Reply to the prompts as shown in the following figure:

Figure 4–54, Verify Format 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

 1:  Use defaults except allow destructive writes [y,n]
                                                  (y) -> n
 2:  Are writes to other than the diagnostics cylinder allowed
    [y,n]                                         (n) -> RETURN
 3:  Are writes to the diagnostics cylinder allowed
    [y,n]                                         (n) -> RETURN
 4:  Number of errors allowed per device each subtest
    [0-65535]                                     (0) -> 100
 5:  Number of devices failed after which test aborts
    [0-65535]                                    (12) -> RETURN
 6:  Threshold for new flaws found during pattern test
    [0-65535]                                     (0) -> RETURN
 7:  Verbosity of test [0-63]                     (3) -> RETURN
 8:  If pattern testing, start after last completed pattern
    [y,n]                                         (y) -> RETURN

              PERIPHERAL CONFIGURATION DATA
     CCU    Chassis   Type     CSR    Int  Unit  Type
    ------  -------  --------  -----  ---  ----  -------
 1) viop 7    0      DKC-204   0xa00   1    0    DKD-206
 2) viop 7    0      DKC-204   0xc00   4    0    DKD-208
 3) viop 7    1      DKC-203   0xa00   3    0    DKD-214

Enter device 99 to begin user-defined configurations or -1 to end selection
 9:  Device selection [-1,1-3,99]                 (-1) -> 1
10:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> y
11:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> RETURN
12:  Device selection [-1,1-3,99]                 (-1) -> 2
13:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> y
14:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> RETURN
15:  Device selection [-1,1-3,99]                 (-1) -> 3
16:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> y
17:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> RETURN
18:  Device selection [-1,1-3,99]                 (-1) -> RETURN
19:  Enter OK, or :NN to return to question NN [OK]
                                                 (OK) -> RETURN

      ------- summary of inputs prints here -------
```

Figure 4–54, Verify Format Parameter Menu (continued)

```
        Subtest 304  0:03:26
dev5130 drv 2 ERROR 13(cntlr) Soft ECC error
  Cyl: 326 Hd: 7 Sect: 12
                     0:05:41
dev5130 drv 3 ERROR 1e(cntlr) Drive faulted
                     0:08:05  passed

************** Test started Wed Mar 16 08:03:52 1988
************** Test ended   Wed Mar 16 10:23:46 1988

test 'dev5130.t' passed
```

In the previous figure, drive 2 appears to have a new defect at cylinder 326 head 7 sector 12. The defect should be verified before slipping the sector. Refer to the next section for information about verifying a defect and slipping a sector. Drive 3 appears to have a more serious problem. The drive faulted in the middle of the subtest. The fault condition is immediately cleared and the operation is retried. The retry succeeded since the drive passed the subtest. This type of error is difficult to isolate. The drive could actually be reporting a fault condition but the problem could also be cabling or a bad controller. Subtest 400, Interactive Test should be used to isolate intermittent problems.

4.6.3 Slipping a Sector

The following example shows how the *dev5130* disk formatter can be used to map out media flaws so those portions of the disk are not used. However, before attempting to slip any sectors which have been reported bad by CONVEX UNIX, consider the following points:

1. Is the error at the reported cylinder, head and sector repeatable? Media flaws usually repeat. This example shows how to confirm a flaw. When confirming the flaw, if *dev5130* prints one of the following error messages, then the error is slippable. Otherwise, some other problem that is not media-related is causing the error.

   The following table lists the media-related errors (use code to correlate with UNIX-reported error):

Table 4–16, Media-Related Errors

ERROR CODE	ERROR DESCRIPTION
0x13	Soft ECC error
0x19	Header checksum error
0x22	Invalid header pad
0x23	Hard ECC error
0x29	Sector not found
0x2b	Invalid sync in data field
0x31	Invalid sync in header
0x6a	Unrecognized header field
0x6b	Mapped header error

2. The Manufacturer's Defect map is used on all drives when they are formatted. In the past, the map has proven to be reliable in identifying media flaws. Therefore, any new flaws indicate possible drive unreliability. If a drive has new defects, the media may be degrading. Slip the first one or two new defects but if new defects continue, a new drive should be installed.

In the following example, CONVEX UNIX has been reporting a soft Error Correction Code (ECC) error at cylinder 356 head 1 sector 19 on drive 1, an NEC 2352 1/2-GByte drive. An inspection of the file */mnt/errlog* on the service processor indicates the error has occurred three times always at the same location. The system has been brought down to the service processor and *dev5130* is about to be executed. The goal is to perform the following:

1. Check the status of the drive to see if it has had maintenance since it was originally formatted.
2. Verify the error is real and is at the same location. Also, see if old errors are located on the same surface near this flaw. For instance, the cylinder number differs by one or two from a previous error, but the head and sector are the same. This condition is an excellent indicator that a real error exists and the media flaw is impacting more than one track on a surface.
3. Slip the bad sector(s).
4. Verify the entire disk to see if any other grown errors occur.

First, invoke the interactive test as shown in the following figure:

Figure 4–55, Interactive Test Invocation Sequence

```
(spu)> sysreset; mminit -s; dshell (RETURN)
CONVEX DIAGNOSTIC SHELL
: test dev5130 -c 4 (RETURN)
```

Upon invocation of the Class 4 subtest, the prompts in the following figure are displayed. Reply to the prompts as shown in the following figure:

Figure 4–56, Interactive 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
 1:  Use defaults except allow destructive writes [y,n]
                                                  (y) -> y

              PERIPHERAL CONFIGURATION DATA
    CCU    Chassis   Type     CSR    Int  Unit  Type
   ------  -------  --------  -----  ---  ----  -------
 1) viop 7    0      DKC-204   0xa00   1    0    DKD-206
 2) viop 7    0      DKC-204   0xc00   4    0    DKD-208
 3) viop 7    1      DKC-203   0xa00   3    0    DKD-214
Enter device 99 to begin user-defined configurations or -1 to end selection
 2:  Device selection [-1,1-3,99]                (-1) -> 1
 3:  Previously formatted on CONVEX system with Interphase 4200/4201 controller
     [y,n]                                        (n) -> y
 4:  Serial number (5 to 31 characters) []
                          (Default: read from drive) -> RETURN
 5:  Device selection [-1,1-3,99]                (-1) -> RETURN
 6:  Enter OK, or :NN to return to question NN [OK]
                                                 (OK) -> RETURN

     ------- summary of inputs prints here -------

***********************************************
* DATA DESTRUCTIVE OPERATION IS ABOUT TO START.*
* VERIFY WRITE-PROTECT IS SET ON ALL DRIVES    *
* EXCEPT THOSE YOU ARE TESTING/FORMATTING.     *
***********************************************

Then reconfirm that you want to continue [yn] -> y

        Subtest 400  0:00:00 Interactive test
Reading defect lists from all selected drives...
Drive  1 All copies are GOOD

                      INTERACTIVE TEST MODE
                      FOR SMD AND ESDI DRIVES

Type 'h' for help
(D1;Save)->
```

Once the `(D1;Save)->` prompt is displayed, any interactive command can be entered.

STEP 1: First check the status of the drive.

In the following figure, the *status* command is entered to display the status of the drive.

Figure 4–57, Use of Status Command Example

```
(D1;save)-> status

                        DRIVE CONFIGURATION DATA
      CCU VME Cntlr Int Unit #   # Phys Log Prev
Drive  #   #   CSR Level #  Cyl Hds Sec Sec Fmtd Name          Serial #
----- --- --- ----- --- --- --- --- --- --- --- ------- ----------------------
   1   7   0  0xa00  1   0  760  19  60  59 yes DKD-206 03658

Drive 1 is currently selected
No track data has been saved yet.
Save of track data is automatic during destructive commands.
Status of defect lists:
Drive  1 All copies are GOOD
Drive information
        Serial number: 03658
        Original Formatter Version: 3.00 Date: Mon Mar 14 17:33:24 CST 1988
        Latest Disk Update Version: 3.00 Date: Wed May 11 14:03:14 CST 1988
        Format options: Manufacturer's defect list was used if available
                        Pattern test was completed using pattern set 2

(D1;Save)->
```

The previous figure shows that the latest disk maintenance was performed on May 11th. Check the log to see what disk maintenance was performed.

STEP 2: Verify the error is real and is at the same location.

Two steps can be used to verify the error. First, perform reads on the track in question to see if any media-related errors occur. If errors occur, the sector is bad and needs slipping. If errors do not occur, then check the current list of flaws and see if any old flaws are on the same head within one cylinder of the new flaw. If this condition is true, then a media flaw which was detected on another cylinder is marginally affecting this track so the sector should still be slipped.

The following figure shows the two steps to verify the error.

Figure 4–58, Use of Verify and List Commands Example

```
(D1;Save)-> verify 10 times 356 1 19
verify_format: pass # 4
dev5130 drv  1 ERROR 13(cntlr) Soft ECC error
  Cyl: 356 Hd: 1 Sect: 19
7
dev5130 drv  1 ERROR 13(cntlr) Soft ECC error
  Cyl: 356 Hd: 1 Sect: 19
10
(D1;Save)-> list
Summary of all flaws on drive 1 serial number 03658:
 CYL HD SEC  BCAI   LEN TYPE  ERR SRC | CYL HD SEC  BCAI   LEN TYPE  ERR SRC
   ---------- list of all flaws for this drive prints here ----------
```

The *verify* command shows whether the error repeats when performing reads only. The *list* command shows all flaws that have been slipped. Check this list for flaws near the new flaw. If executing either of these commands indicate that the flaw should be slipped, then continue to the next step. Otherwise, use the *data_verify* and *pattern_test* commands which perform writes before reads. The error may only show up when the data is written and then read.

STEP 3: Slip the bad sector(s)

To slip cylinder 356 head 1 sector 19, enter the commands and data shown in the following figure:

Figure 4–59, Use of Slip Command Example

```
(D1;Save)-> slip
Type 'h' for help with slip commands
slip (D1;Save;Sector)-> 356 1 19
slip (D1;Save;Sector)-> execute

Summary of new flaws on drive 1 serial number 03658:
 CYL HD SEC  BCAI   LEN TYPE  ERR SRC | CYL HD SEC  BCAI   LEN TYPE  ERR SRC
 356  1  19 15200  4825      NONE NEW |

        Are you sure inputs are ready for slipping? [yn] (y) -> y

 Number of spares per track: 1
 cyl hd sec  bcai  len  action              additional information
---- -- --- ----- ----- ------- ---------------------------------------------
 356  1  19 15200  4825 slipped

slip (D1;Save;Sector)-> quit
(D1;Save)->
```

The sector is now slipped.

If more than one flaw is to be slipped, enter all flaws before entering the *execute* command at the `slip (D1;Save)->` prompt.

STEP 4: Verify the entire disk to see if any other grown errors occur.

The *verify_format* command can be used to verify that the entire usable part of the disk is error-free. To verify the entire usable part of the disk, enter the *verify_format* command and data as shown in the following figure:

Figure 4–60, Use of Verify Format Command Example

```
(D1;Save)-> verify 0 0 0 to end
verify_format: pass # 1
(D1;Save)-> quit
```

Since errors did not occur when the verify was executed, the entire disk is readable. This verify may not detect intermittent errors which only a long-term read of the disk would find, but the verify does perform a quick check for flaws on the disk. If any additional flaws occur during this verify, repeat the above steps to slip the new flaws.

4.7 Manufacturer's Defect List and Grown Defect List

All drives on Interphase 4200 V/SMD controllers have the innermost cylinder reserved for five copies of the Manufacturer's Defect (MD) list and Grown Defects (GD) list. Drives on Interphase 4201 V/ESDI controllers use the next to the innermost cylinder. This is due to the fact that the original Manufacturer's Defect map is recorded on the innermost cylinder of ESDI drives and is not overwritten by this formatter. Unfortunately, SMD drives have part of the original Manufacturer's Defect map written to every track on the drive so there is no way to preserve it.

Since UNIX must not write to the defect data copies, the UNIX disk parameters file */etc/disktab* specifies one less cylinder than the disk actually has.

4.7.1 Defect Cylinder Organization on Drives

The defect cylinder on a drive has the following organization:

head 0 — 1st copy of defect data
head 1 — 2nd copy
head 2 — 3rd copy
head 3 — 4th copy
head 4 — 5th copy
heads 5-9 — continuations of 1st through 5th copies, respectively (if needed)
heads 10-14 — continuations of 1st through 5th copies, respectively (if needed)
heads 15-19 — continuations of 1st through 5th copies, respectively (if needed)

The MD list starts in sector 0 of the first track as defined above and continues on consecutive tracks. The GD list starts in the next sector following the last sector used by the MD list.

4.7.2 Format of Manufacturer's Defect List

The Manufacturer's Defect (MD) list has the following format:

```
word 0          - Magic Number - 0x492ab20e
word 1          - Version Number of formatter when Manufacturer's defect
                  list was written. The number 300 in the lower halfword
                  means version 3.00 which is the initial version of this
                  formatter.
                  The upper halfword is set to zero and is unused for now.
word 2          - Date (seconds since Jan 1, 1970)
word 3          - Number of Manufacturer's Defect Entries  (Assume m entries)
word 4-6        - First of Manufacturer's Defect Entries
     :
word 3m+4 - 3m+6 - Last of Manufacturer's Defect Entries
```

4.7.3 Format of Grown Defect List

The Grown Defect (GD) list starts in the next available sector after the Manufacturer's Defect (MD) list. It is therefore necessary to read the number of entries in the MD list before the GD list can be located.

The GD list has the following format:

```
word 0            - Magic Number - 0x92b41fe7
word 1
    halfword 0    - Control bits (in binary)
                    ssssppppp XXXXXXXm

                    where ssss is the pattern set used during pattern test.
                    The pattern sets are defined in the description of
                    Subtest 301, Format and Pattern Test.
                    pppp is the index (1-based) into the
                    pattern set. If pppp is 1111, then no pattern testing
                    has been done on this drive. If pppp is 0000, then
                    pattern testing was completed. The bit labeled 'm'
                    is set if the MD list was ignored. This is an option
                    at format setup (Subtest 300, Format Setup).

    halfword 1    - Version Number of formatter when Grown Defect List
                    was written. Two implied decimal places. I.E.- the
                    number 300 will mean version 3.00.
word 2            - Date (seconds since Jan 1, 1970)
word 3-10         - Serial Number (up to 31 characters)
word 11           - Number of Grown Defect Entries  (Assume n entries)
word 12-14        - First of Grown Defect Entries
     :
word 3n+12 - 3n+14 - Last of Grown Defect Entries
```

4.7.4 Format of Entries in the Manufacturer's Defect List and Grown Defect List

Each entry in both the Manufacturer's Defect (MD) list and Grown Defect (GD) list have the following format:

Bytes	Description
0-1	Cylinder
2	Head
3	Sector position (logical based on no slipped sectors or spares)
4	Encoded Bit field
	11110000 - Unused in *dev5130*
	00001111 - Error program detected - definitions of these bits will change depending on the formatter used
	XXXX0000 - not attempted yet or none detected
	XXXX0001 - soft ECC
	XXXX0010 - header checksum
	XXXX0011 - invalid header pad
	XXXX0100 - hard ECC
	XXXX0101 - sector not found
	XXXX0110 - invalid data sync
	XXXX0111 - invalid header sync
	XXXX1000 - unrecognizable header
	XXXX1001 - mapped header error
	XXXX1010 - this code is unused
	XXXX1011 - other error
5-7	Byte Count after Index
8	Error source - Bit field
	XXXXXX00 - User input
	XXXXXX01 - Program-detected
	XXXXXX10 - Manufacturer's Defect Map
	XXXX1XXX - Can't map this flaw - map-track command failed
	XXX1XXXX - Track is defective
	XX1XXXXX - More than 4 flaws on this track
	X1XXXXXX - New entry
	1XXXXXXX - Ignore this error
9-11	Bit length

4.8 Track Relocation Area Description

The first five sectors of each of the last five tracks in the track relocation area (track map log and alternate track area) are reserved for holding one copy of the track map log.

Each entry in the log corresponds to a track in the alternate track area. The first entry in the log corresponds to the last track - 5 on the innermost cylinder of the track relocation area. The second entry corresponds to the last track - 6 and so on.

The file */mnt/bin/lib/DB_diskfmt* defines the number of cylinders that are set aside for the track map log and alternate track area. At least one cylinder must be dedicated for the track map log and alternate track area. As of version 3.00 of this formatter, the technique for calculating the

number of cylinders is 0.005 * total number of cylinders rounded up. If */mnt/bin/lib/DB_diskfmt* defines more cylinders than are necessary to total 636 tracks (the maximum number of entries allowed in the track map log), then only the innermost 636 tracks of the alternate track area are set aside as alternate tracks. So, tracks beyond those defined in the */mnt/bin/lib/DB_diskfmt* file are not used.

> **NOTE**
>
> If any media flaws fall in the alternate track area, the track is mapped to itself.

For example, on the NEC 2363 drive, if cylinder 1021, head 20 is bad, then the 5th word of the track map log is set to cylinder 1021 head 20 so that it is not used by other tracks.

The association of entries with tracks on a 1 GByte NEC 2363 drive which has cylinders 0-1023, heads 0-26, and sectors 0-66 is the following:

Track Relocation Area	Entry	Cylinder	Head
Track map log	Log copy 1	1021	26
	Log copy 2	1021	25
	Log copy 3	1021	24
	Log copy 4	1021	23
	Log copy 5	1021	22
Alternate track area	Alt trk 0	1021	21
	Alt trk 1	1021	20
	:	:	:
	Alt trk 21	1021	0
	Alt trk 22	1020	26
	Alt trk 23	1020	25
	:	:	:
	-------- etc ---------		

The track map log has the following format:

```
word 0           - Magic Number - 0xca29e40b
word 1           - Version Number of formatter when written. The number
                   300 in the lower halfword will mean version 3.00.
                   The upper halfword is set to zero and is unused for now.
word 2           - Date (seconds since Jan 1, 1970)
word 3           - Number of Track Map Entries (one word per entry)
word 4           - Entry 0
                   Upper halfword - cylinder of bad track
                   Lower halfword - head of bad track
words 5 - 639    - Remaining 635 entries
```

4.9 Disk Parameters File, *DB_diskfmt* Description

The disk parameters file, */mnt/bin/lib/DB_diskfmt*, contains information about disk drives. This information is required to be able to format or test new drives. Each line in the file contains a number of fields separated by one or more spaces. Comments start with a # in column 1. To add new drives, create a new entry with all the fields defined, and then add the drive to the */ioconfig* file. As of the time of this printing, the *DB_diskfmt* file contains the following information for all CONVEX machines:

Figure 4–61, Contents of the *DB_diskfmt* File

```
# DB_diskfmt - file of disk parameters
# >>>> WARNING - DO NOT USE 'diskfmt' TO FORMAT! It is no longer compatible
# >>>>           with the CONVEX FORMAT! Instead, format MBUS-attached drives
# >>>>           with 'dev4110' and VME-attached drives with 'dev5130'.
# KEY FOR DRIVE NAMES (unformatted capacity is given in parentheses):
#  Name          Description            Name          Description
# DKD-001       Fujitsu Eagle (452MB) DKD-008,208  NEC 2363 (1080MB)
# DKD-002       CDC 9766 (300MB)      DKD-214      Hitachi DK514-38 (356MB)
# DKD-005,206   NEC 2352 (500MB)
#------- XYLOGICS 450/451 SMD CONTROLLER (MBUS) -------
#  a     b    c   d  e  f    g     h    i j k  l  m   n    o  p
DKD-001 2   842 20 46 45 4800 28160 1 0 1  0  0  smd mfm y
DKD-002 0   823 19 32 31 5040 20160 1 0 1  0  0  smd mfm y
DKD-005 0   760 19 60 59 4832 36288 1 0 1  0  0  smd 2-7 y
DKD-008 1 1024 27 68 67 4816 40960 1 0 1  0  0  smd 2-7 y
#------- INTERPHASE 4201 ESDI CONTROLLER (VME) --------
#  a     b    c   d  e  f    g     h    i j k  l  m   n    o  p
DKD-214 0   903 14 51 50 4736 30240 5 5 1  8  8 esdi 2-7 n
#------- INTERPHASE 4200 SMD CONTROLLER (VME) ---------
#  a     b    c   d  e  f    g     h    i j k  l  m   n    o  p
DKD-206 0   760 19 60 59 4832 36288 5 4 1 12 12  smd 2-7 n
DKD-208 0 1024 27 68 67 4816 40960 5 6 1 16 12  smd 2-7 n
# LEGEND:
# a - drive name                Must be DKD-0XX for Multibus and DKD-2XX for
#                               VMEbus
# b - disk type                 For Xylogics controller
# c - # of cylinders
# d - # of heads
# e - # of physical sectors     Number of actual sectors excluding runt
# f - # of logical sectors      Number of physical sectors (e) minus number of
#                               spares
# g - bits per sector           Number of bits between sector pulses
# h - bytes per track           Total number of unformatted bytes per track
# i - skew                      Sector offset from one head to the next
#                               Must be 1 when using Xylogics 450/451 cntlr
# j - # of relocation tracks    .5% of number of cylinders (c). Raise
#                               fractional part to next higher whole number.
#                               Ignored by dev4110 (Multibus formatter)
# k - interleave                sector separation between consecutive sectors
#                               Currently must be 1 for Xylogics 450/451 cntlr
# l - gap 1 size                Number of halfwords in gap before header
#                               (2 bytes per halfword)
#                               Ignored by dev4110 (Multibus formatter)
# m - gap 2 size                Number of halfwords in gap following header
#                               (2 bytes per halfword)
#                               Ignored by dev4110 (Multibus formatter)
# n - drive interface           Used to determine how to read manufacturer's
#                               defect map.  Currently, smd or esdi
#                               Ignored by dev4110 (Multibus formatter)
# o - data encoding scheme      Way data is encoded on the media. Used to
#                               select patterns for pattern test.
#                               Currently mfm, 2-7 or 1-7
# p - Are spares interleaved    For Xylogics, 'y'. For Interphase, 'n'.
```

4.10 Diagnostic Cylinder Description

The diagnostic cylinder is located on the first full cylinder preceding the sector forwarding area. Cylinder 840 is the diagnostic cylinder on a Fujitsu Eagle drive, and cylinder 820 is the diagnostic cylinder on a CDC SMD drive.

Each track of the cylinder is independent of the other tracks since there is a "table of contents" stored at sector 0 of each track This table describes what is stored on the remainder of that track. The format of this table is shown below:

Figure 4–62, Diagnostic Cylinder Table of Contents Format

```
/* ------------------------------------------------------------------ *
 *              Diagnostic Cylinder Definitions                       *
 * ------------------------------------------------------------------ */
#define NOT_USED        0x00000000        /* position in tbl not used     */
#define NO_HDR          0x00000001        /* this sectors header deleted  */
#define TBL_CONT        0x00000002        /* sector contains tbl of cntnts*/
#define ECC_ERR         0x00000003        /* sector written with bad ecc  */
#define PATTERN         0x00000004        /* sector written with pattern  */

 struct   tbl_cont {
    int   magic_nbr;            /* identifies this as a table of contents */
    int   version;              /* version number of this table */
    int   cyltksc;              /* sector header for this sector*/
    struct {
     int   sec_cont;          /* defines contents of corresponding sector */
     int   pattern;           /* pattern or mask used to verify contents  */
    } sec_tbl[62];
    int   checksum;          /* arithmetic tally of all the above fields */
 };
```

Each track on the cylinder is formatted in such a way that sector 0 is always on the track; sector 0 is written and verified without error when the drive is formatted.

The first field in the table contains the magic number 0x84101500. If this number is not present in the first position of a table, then someone has altered the diagnostic cylinder.

The second field in the table contains the version number. This number is an ordinal number starting with one and increasing as needed. Its purpose is to allow some measure of compatibility of newer diagnostic cylinders with older software. This number is checked if an unknown entry in the sector table (sec_tbl) is encountered. If the version number in the table is greater than the version number of the software, the diagnostic cylinder is assumed to have been formatted with a newer revision of software, and the opcode in the sector table is new to the older software. Therefore, the software is not familiar with the opcode, and the entry in the sec_tbl is ignored.

The next field in the table contains a copy of the sector header (cyltksc) for the table of contents.

The next field in the table contains the sector table (sec_tbl). This table describes what is stored on the rest of the sectors in this track. The sector table consists of 62 entries of 2 int's. Each entry (0 - 61) in this table, corresponds to a sector (0 - 61) on the track. The first position in

each entry is the sector contents (sec_cont) field. If the track has more than 62 sectors, the extra sectors are not used or checked by current software. The following table describes what is currently defined:

Table 4–17, Defined Values for Sector Contents Field

FIELD	DESCRIPTION
NOT_USED	This sector has not been initialized and should not be checked.
NO_HDR	When this sector is read, the controller should return a 0x05 sector not found error.
TBL_CONT	This sector contains a copy of the table of contents.
ECC_ERROR	This sector contains a copy of the table of contents that has been corrupted to give some form of ECC error. The pattern field contains a mask which may be xor'ed with the magic number field to correct the error. If the mask contains 11 or fewer one bits, the controller should report a soft ECC error when this sector is read. If the mask contains more than 11 one bits, the error reported should be a hard ECC error.
PATTERN	This sector has been filled with the data contained in the pattern field. No errors should be encountered when the sector is read.

The last field in the table of contents is a checksum. This is an arithmetic tally of all the bytes in the table of contents. This value is checked before any of the data in the table of contents is used.

4.11 Error Messages

The following are error messages for the *dev5130* test. The following error format is generated from a test and is not associated with a controller or drive:

```
dev5130 ERROR 56(test)-Unable to get SPU memory
```

The following error format is generated from a test and is associated with a controller but not with a specific drive:

```
dev5130 ccu/vb/csr 7/0/e00: ERROR 94(test) No cntlr interrupt in 5 seconds
```

The last error format shown below is generated from a controller and is associated with a specific drive:

```
dev5130 drv  1 ERROR 13(cntlr) Soft ECC error
 Cyl: 356 Hd: 1 Sect: 19
```

The following are possible errors:

- 0x10(cntlr) Disk not ready (or, if a DKD-214, "Write Protected")
- 0x12(cntlr) Seek error
- 0x13(cntlr) Soft ECC error
- 0x14(cntlr) Invalid Command Code
- 0x15(cntlr) Illegal fetch and execute
- 0x16(cntlr) Invalid sector in command
- 0x17(cntlr) Illegal memory type
- 0x18(cntlr) Bus timeout
- 0x19(cntlr) Header checksum error
- 0x1a(cntlr) Disk write protected
- 0x1b(cntlr) Unit not selected (or, "Powered Off")
- 0x1c(cntlr) Seek error timeout
- 0x1d(cntlr) Fault timeout
- 0x1e(cntlr) Drive faulted
- 0x1f(cntlr) Ready timeout
- 0x20(cntlr) End of medium
- 0x21(cntlr) Translation fault
- 0x22(cntlr) Invalid header pad
- 0x23(cntlr) Hard ECC error
- 0x24(cntlr) Translation error, cyl
- 0x25(cntlr) Translation error, head
- 0x26(cntlr) Translation error, sect
- 0x27(cntlr) Data overrun
- 0x28(cntlr) No index pulse on format
- 0x29(cntlr) Sector not found
- 0x2a(cntlr) ID field error-wrong head
- 0x2b(cntlr) Invalid sync in data field
- 0x2c(cntlr) No valid header found
- 0x2d(cntlr) Seek timeout error
- 0x2e(cntlr) Busy timeout
- 0x2f(cntlr) Not on cylinder
- 0x30(cntlr) RTZ timeout
- 0x31(cntlr) Invalid sync in header
- 0x3f(cntlr) No heads specified
- 0x40(cntlr) Unit not initialized
- 0x41(cntlr) Not used
- 0x42(cntlr) Gap specification error

- 0x42(test) No consecutive sectors found
- 0x43(test) Flaw log is full
- 0x47(test) Bad parm to calc_cyl
- 0x48(test) Bad parm to calc_hd
- 0x49(test) Bad parm to calc_sec
- 0x4a(test) IOP print utility memory error
- 0x4b(cntlr) Seek error
- 0x4b(test) Device FAILED
- 0x4c(test) Input parms don't allow write
- 0x4e(test) Diag. trk has all bad headers
- 0x4f(test) Diag. cylinder bad
- 0x50(cntlr) Sectors/track error
- 0x50(test) Bad data compare on diag. cyl
- 0x51(cntlr) Bytes/sector spec error
- 0x51(test) No err. Hdr-not-found expected
- 0x52(cntlr) Interleave spec factor
- 0x52(test) Soft ECC error did not occur
- 0x53(cntlr) Invalid head address
- 0x53(test) Hard ECC error did not occur
- 0x54(cntlr) Invalid cylinder addr
- 0x54(test) Data compare err on diag cyl
- 0x55(test) Setup failed - aborting test
- 0x56(test) Unable to get SPU memory
- 0x57(test) Track map log is full
- 0x58(test) No defect lists could be read
- 0x59(test) Can't read Manufacturer's defects
- 0x5a(test) New flaws exceeded user limit
- 0x5b(test) No copies successfully written
- 0x5c(test) Less than 3 good copies of defect lists
- 0x5d(cntlr) Invalid DMA xfer count
- 0x5f(test) Hardware is suspected to be bad
- 0x60(cntlr) IOPB failed
- 0x60(test) Drive is previously formatted
- 0x61(cntlr) DMA failed
- 0x61(test) Drive is not previously formatted
- 0x62(cntlr) Illegal VME address
- 0x62(test) Unable to read track headers
- 0x63(test) Unable to access main memory
- 0x64(test) Mapped track has invalid headers

- 0x65(test) Alternate track has invalid headers
- 0x66(test) Incorrect alloc of pattern space
- 0x67(test) Message received was not handshake
- 0x68(test) Incorrect handshake message format
- 0x69(test) Defect list to long to fit on disk
- 0x6a(cntlr) Unrecognized header field
- 0x6a(test) Prev. formatted drive appears unformatted
- 0x6b(cntlr) Mapped header error
- 0x6b(test) User terminated drive
- 0x6c(test) Controller firmware is out of date
- 0x6d(test) Bad data written by Format with data
- 0x6f(cntlr) No spare sector enabled
- 0x77(cntlr) Command aborted
- 0x78(cntlr) ACFAIL detected
- 0x80(test) Data compare error
- 0x81(test) Seek requested with a seek count = 0
- 0x85(test) Desired IOPB not in active chain
- 0x86(test) Message received on wrong subqueue
- 0x87(test) Wrong subqueue active and locked
- 0x88(test) Wrong subqueue active and MBS error
- 0x89(test) Wrong subqueue active, error invalid
- 0x8a(test) river executed for no reason
- 0x8c(test) ntlr busy after hard error
- 0x92(test) EGOS routine wndw_alloc returned 0
- 0x93(test) Task init attempt-error pending
- 0x94(test) No cntlr interrupt in 5 seconds
- 0x96(test) Write/Read error in controller regs
- 0x9a(test) Finished task but int. state wrong
- 0x9b(test) Controller is not present
- 0x9c(test) Sector interleave or skew error
- 0x9d(test) Bad cylinder address after seek
- 0xa0(test) Expected error did not occur
- 0xa2(test) No free IOPBs for retry
- 0xa3(test) Controller didn't finish IOPB
- 0xa4(test) Unknown error reported by cntlr
- 0xa5(test) Internal Defect log is full
- 0xa6(test) Controller doesn't respond
- 0xa7(test) Track doesn't contain flaw data
- 0xa8(test) Cntlr reporting board not ok

- 0xa9(test) Software error-Invalid verify
- 0xaa(test) Cntlr GO/BUSY set after diag cmd
- 0xab(test) Header contains bad cyl number
- 0xac(test) Header contains bad head number
- 0xad(test) Attempt to configure cntlr failed
- 0xae(test) Unexpected value in cntlr CSR
- 0xaf(test) Bad cyl, hd, or sec in IOPB
- 0xb0(test) NI with more than 1 spare
- 0xc0(cntlr) Both bits set (from appl. note)
- 0xc1(cntlr) MSE without init long (from appl. note)
- 0xff(cntlr) Command not implemented

4.11.1 Special Error Messages for *dev5130*

In some circumstances, a certain set of error messages are displayed if the peripheral is not powered on, is write protected, or is cabled incorrectly. The test gives the user several options to proceed with, depending on the circumstances. The following examples illustrate these type of error messages and user options that are presented. The first example illustrates the messages produced when the peripheral is not ready (ie., power is not on or a cable is not connected). The second example illustrates the messages produced when the peripheral is write protected and an operation attempts a write.

> **NOTE**
>
> In the following examples, all **boldface** entries are user entered information.

Figure 4–63, Drive Not Ready Error Example

```
(D1;Save)-> alternate 0 to 1000
Beginning alternate seeks (1 times)...
dev5130 drv 1 1b(cntlr) Unit not selected
  Cyl:   0 Hd: 0 # Seeks:2
  IOPB: opt  err  cyl hd/s  cnt bufh buf1 bmem inr  dma ptrh ptr1 imem skew
       8102 821b 0000 0000 0001 0001 2000 033d 0100 1000 0000 0000 0000 0000
Attempt to fix the problem. You can issue UNIX commands if it helps.
Then enter on of the following:
        <CR> to retry the operation
        'nofix' if unable to fix the problem
UNIX command, <CR> for retry, or nofix: RETURN

dev5130 drv 1 1b(cntlr) Unit not selected
  Cyl:   0 Hd: 0 Sect:  0 # Seeks:2
  IOPB: opt  err  cyl hd/s  cnt bufh buf1 bmem intr  dma ptrh ptr1 imem skew
       8102 821b 0000 0000 0001 0001 2000 033d 0100 1000 0000 0000 0000 0000
Attempt to fix the problem. You can issue UNIX commands if ti helps.
Then enter on of the following:
        <CR> to retry the operation
        'nofix' if unable to fix the problem
UNIX command, <CR> for retry, or nofix: noflx

(D1;Save)-> alternate 20 times 0 to 1000
Beginning alternate seeks (20 times)....
dev5130 drv 1 1b(cntlr) Unit not selected
  Cyl:   0 Hd: 0 # Seeks:21
  IOPB: opt  err cyl  hd/s  cnt bufh buf1 bmem inr   dma ptrh ptr1 imem skew
       8a02 821b 0000 0000 0001 0000 0000 033d 0100 1000 0000 0000 0000 0000
```

Figure 4–64, Write Protect Error Example

```
(d1;Save)-> pattern_test 1022 10 30 with 0
     Patterns used:
      00000000
        No data has been previously saved. Savin this track
        Data save operation complete
 pass #1
dev5130 drv 1 1a(cntlr) Disk write protected
  Cyl:1022 Hd:10 Sect: 30 # Seeks:2
  IOPB: opt  err  cyl hd/s  cnt bufh buf1 bmem intr  dma ptrh ptr1 imem skew
       8202 821a 03fe 0a1e 0001 0005 2000 033d 0100 1000 0000 0000 0000 0000
Attempt to fix the problem. You can issue UNIX commands if it helps.
Then enter one of the following:
        <CR> to retry the operation
        'nofix' if unable to fix the problem
UNIX command, <CR> for retry, or nofix: noflx
        Restoring data to track

dev5130 drv 1 1a(cntlr) Disk writ protected
  Cyl:1022 Hd:10 Sect: 0 # Seeks:2
  IOPB: opt  err  cyl hd/s  cnt bufh buf1 bmem intr  dma ptrh ptr1 imem skew
       8202 821a 03fe 0a00 0001 0005 2000 033d 0100 1000 0000 0000 0000 0000
                        1 attempts to write sector failed
                        what to do [# of retries/force/skip] (1) ->skip
```

Figure 4–65, Power Off Error Message

```
(D1;Save)-> verify_format 0 0 0
verify_format: pass #1
dev5130 drv 1 10(cntlr) Disk no ready
  Cyl:   0 Hd: 0 Sect:  0 # Seeks:2
IOPB: opt  err  cyl hd/s  cnt bufh buf1 bmem inr  dma ptrh ptr1 imem skew
     8102 8210 0000 0000 0001 0001 2000 033d 0100 1000 0000 0000 0000 0000
Attempt to fix the problem. You can issue UNIX commands if it helps.
Then enter on of the following:
        <CR> to retry the operation
        'nofix' if unable to fix the problem
UNIX command, <CR> for retry, or nofix: noflx
```

Appendix A
Reporting Problems

A.1 Overview

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the *contact* utility. The *contact* utility is an online system for reporting problems to the TAC. To learn *contact* by using it, enter **contact** at the system prompt and then answer the questions as they appear on the screen. To find out more about using *contact*, read through this appendix. It describes prerequisites and tips for using *contact* and the step-by-step process *contact* takes you through.

A.2 Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address the diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation problem, contact the TAC. This group stands ready to solve such problems.

A.3 The *contact* Utility

The TAC recommends using the *contact* utility to report a hardware, software, or documentation problem. The *contact* utility is an interactive utility that helps the TAC track reports and route them to the the CONVEX personnel most qualified to fix them.

After invoking *contact*, it prompts for information about the problem. When you finish your report, *contact* electronically mails it to the TAC. You are notified within 48 hours that the TAC has received your report.

A.4 Prerequisites

To use *contact* requires

- a UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- the full path name of the program or utility in question
- the version number of the program or utility in question

A.4.1 UUCP Connection

Before using *contact*, check with your system administrator to be sure there is a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX system to another. The *uucp* (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

A.4.2 Finding the Program Path Name

To determine the full path name of the program or utility in question, use the *which* command. The following screen illustrates using the *which* command to find the full path name of the loader (*ld*) utility:

```
>which ld
/bin/ld
>
```

In this example, the full path name of the loader is */bin/ld*.

For more information on the *which* command, refer to the *which*(1) man page. You can also use the *info* online information system. Enter **info which** at the system prompt. If you use the C shell (*csh*), you can also use the *whence* command to find the program path name. The *whence* command works like *which*, only faster.

A.4.3 Finding the Program Version Number

To determine the version number of the program or utility in question, use the *vers* command. The following screen illustrates using the *vers* command (enter **vers**, then the path name of the program or utility) to find the version number of the loader (*ld*) utility.

```
>vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader utility version number is 7.0.

For more information on the *vers* command, refer to the *vers*(1) man page. You can also use the *info* online information system. To do so, enter **info vers** at the system prompt.

A.5 Tips on Using the *contact* Utility

The *contact* utility is interactive and easy to use. This section lists tips to help use it efficiently. In particular, this section tells how to

- use a *.contact* file
- abort a contact session
- resubmit an aborted report
- suspend a contact session
- move from one prompt to another
- use tilde-escape sequences in the *contact* utility

A.5.1 Using a *.contact* File

When invoked, *contact* prompts for information regarding the problem. The first prompt is for your name, title, phone number, and company name. You can, however, create a *.contact* file to skip this first prompt. Follow these steps:

1. Create a *.contact* file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke *contact*, it automatically includes the *.contact* file as input for the first prompt and proceeds to the next prompt.

A.5.2 Aborting the Report

To abort a contact report, either enter the interrupt key (usually CTRL-C) or choose the abort option when prompted by the *contact* utility. Using CTRL-C to abort does not save the contents of the report. Using the abort option saves the contents of the report in a file named *dead.report* in your home directory.

A.5.3 Submitting the *dead.report* File

When aborting a contact session, the *contact* utility saves the report in a file named *dead.report* in your home directory. Using the *contact* command with the *-r* option automatically merges the contents of the *dead.report* file into the new contact session. Enter

contact -r

and *contact* finds the *dead.report* file in your home directory and merges it into the contact report. You can then edit the report. When you end the editing session, *contact* returns to the final prompt, which asks you to review, edit, submit, or abort the report.

A.5.4 Suspending a Report

Sometimes it is necessary to stop in the middle of a contact report and return to the shell (for instance, to suspend the contact session to find the program path name or version number). To suspend the contact session, press CTRL-Z. To return to the contact session, enter **fg**. Using CTRL-Z and the *fg* (foreground) command lets you switch back and forth between the *contact* utility and the shell. You cannot, however, use CTRL-Z and *fg* to switch back and forth if you are using a Bourne shell (*sh*).

A.5.5 Ending a Response

The *contact* utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press RETURN. Other prompts require more than a one-line response; to move to the next prompt, press CTRL-D.

A.5.6 Tilde-Escape Sequences

The *contact* utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. The following tilde sequences are recognized by *contact*:

~e	Start the text editor (defined in your EDITOR environment variable).
~h	Display a list of available tilde-escape sequences.
~p	Print the contact report to the terminal screen.
~r *filename*	Read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence only works for prompts that allow more a than one-line response.
~~	Insert a single tilde as the first character in the line.

A.6 Using the *contact* Utility

The *contact* utility prompts for the following information:

- your name, title, phone number, and corporate name
- the name and version of the product involved
- a one-line summary of the problem
- a detailed description of the problem
- the priority of the problem
- instructions on how to reproduce the problem
- comments about the problem
- comments about the documentation supporting the problem
- files to include in the contact report

The following is a step-by-step discussion of these prompts:

1a. To invoke the *contact* utility, enter **contact** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. The following screen illustrates the *contact* command and the system response:

```
>contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

1b. If there is a *.contact* file in your home directory, *contact* skips the first prompt. The following screen illustrates the *contact* command and the system response when a *.contact* file is in your home directory:

```
>contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

2. The *contact* utility prompts for the version number of the product. If you do not know the version number, use CTRL-Z to suspend the session. Use the *which* (or *whence* if using *csh*) and *vers* commands to find the version number of the product. Use the *fg* command to return to the session and enter the version number in the form X.X or X.X.X.X.
3. The *contact* utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Make this summary as descriptive as possible in one line.
4. The *contact* utility prompts for a detailed description of the problem. Make this description as complete as possible. Include source code and a stack backtrace whenever possible. (Refer to the *adb*(1) or *csd*(1) man page for information on obtaining a stack backtrace.) The more information provided, the quicker the TAC can isolate and solve the problem.
5. The *contact* utility prompts for the priority of the problem. The following screen illustrates this prompt and the priority levels from which to choose; you must enter a priority number.

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
>
```

6. The *contact* utility prompts for an explanation of how to reproduce the problem. Include the command syntax and options you used and anything else you did to make your program run.
7. The *contact* utility prompts for any other pertinent comments. Include any relevant information.
8. The *contact* utility prompts for suggestions regarding the documentation supporting the product. Indicate if the documentation could be revised to address the question.
9. The *contact* utility asks for the names of files necessary to reproduce the problem. The following screen illustrates the *contact* prompt and sample user response:

```
Are there any files that should be included in this report (yes | no)?
>yes
Please enter the names of the files, one to a line (^D to terminate)
>test.f
>~/subroutines/sub.f
>
```

> **NOTE**
>
> Tilde-escape sequences are not recognized in responses to this prompt. Instead, *contact* treats a tilde in this section to mean your home directory. This convention is based on use of the tilde for expanding file names in *csh*.

If the files specified are small text files, they are automatically included in the contact report. If the files are too big to be included in this report, *contact* gives further instructions on how to submit these files.

To specify a directory, combine the directory files into a single file using the *tar* command (refer to the *tar*(1) man page for further information) or enter each file name in the directory on a single line in the contact report.

10. The *contact* utility prompts you to review, edit, submit, or abort the contact report. The following screen illustrates this prompt:

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review — Review the text of your contact report. You are then prompted again to select an option.

Edit — Edit the text of the contact report. If you choose to edit the report, *contact* puts you in your default text editor.

Submit — Send the report to the CONVEX TAC. You are notified within 48 hours that the TAC has received the report. This option exits the *contact* utility and returns you to the shell environment.

Abort — Save the text of your report in a file named *dead.report* in your home directory. This option exits the *contact* utility and returns you to the shell environment.

Index

T

U

V

W

CONVEX VMEbus SMD/ESDI Disk and Formatter
(*dev5130*) Diagnostics Manual
Document No. 760-003030-000
First Edition

Reader's Forum

Please use this form to submit comments or questions concerning the clarity and service of this manual. Constructive critical comments are most welcome and help us continue in our efforts to generate quality customer documentation. Please list the page number for questions or comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
From all locations in continental U.S.	1(800)952-0379
From locations in Alaska & Hawaii	1(214)497-4379
From locations in Canada	1(800)345-2384
From all other locations	Contact nearest CONVEX office

Direct mail orders to:

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

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

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

(Fold Here Second)

(Tape or Staple)