Bootable/Standalone Diagnostics Manual

This document contains highly-sensitive confidential information that may only be viewed by employees of Solbourne Computer, Inc.

DO NOT COPY OR DISTRIBUTE THIS MANUAL.

SOLBOURNE COMPUTER, Inc. 1900 Pike Road Longmont, Colorado 80501 (303) 772-3400 Solbourne, Series4/500, and Series4/500 are trademarks of Solbourne Computer, Inc

Part Number: 101490-AC

October 1989

Copyright © 1989 by Solbourne Computer, Inc. All rights reserved. No part of this publication may be reproduced, stored in any media or in any type of retrieval system, transmitted in any form (e.g., electronic, mechanical, photocopying, recording) or translated into any language or computer language without the prior written permission of Solbourne Computer, Inc., 1900 Pike Road, Longmont, Colorado 80501. There is no right to reverse engineer, decompile, or disassemble the information contained herein or in the accompanying software.

Solbourne Computer, Inc. reserves the right to revise this publication and to make changes from time to time without obligation to notify any person of such revisions or changes.

Preface

This manual describes dg, the Solbourne Computer, Inc., standalone/test controller for the Solbourne systems. This manual contains eight sections and three appendices, as follows:

Section 1 - Introduction This section introduces the Bootable/Standalone Diagnostics program dg.

- Section 2 Getting Started with dg This section explains how to begin using dg.
- Section 3 Overview of dg Tests This section presents an overview of the dg tests available.
- Section 4 Memory Tests The suite of tests for Memory Boards are given in this section.
- Section 5 I/O Tests The various tests available for the System Board are given in this section.
- Section 6 Graphics Tests This section presents the tests currently available for Graphics Boards.
- Section 7 Miscellaneous Tests Miscellaneous tests available under dg are listed in this section.

Section 8 - Commands This section gives the user commands available when using dg.

Appendix A - MARCH Algorithm This appendix gives an explaination of how the MARCH algorithm works.

Appendix B - Moving Inversions Test Algorithm This appendix gives an explaination of how the moving inversions test algorithm works.

Appendix C - Frame Buffer Test Modes

The modes available for the frame buffer tests are given in this appendix.

Table of Contents

Section 1: Introduction	1-1
1.1 Introduction	
1.2 Related Documentation	1-1
Section 2: Getting Started with dg	2-1
2.1 Introduction	
2.2 Invoking the dg Program	2-1
2.3 Entering Commands to the dg Prompt	2-2
2.4 Using dg Commands	2-3
2.4.1 dg Commands	2-3
2.5 Starting Test Execution	2-4
2.5.1 Using the menu Command	
2.5.2 Variations of Test Execution	
2.6 Handling Test Failures	2-9
2.7 Exiting dg	2-10
Section 3: Overview of dg Tests	3-1
3.1 Introduction	
3.2 Overview of the Menu Structure	
3.3 Numerical Test Listing	3-4
Section 4: Memory Tests	
4.1 Introduction	
4.2 Test 1 - Cache Block Virtual Alias Test (affected by prompt)	4-1
4.3 Test 2 - Memory Data RAM Test (affected by prompt)	4-1
4.4 Test 3 - Memory ECC RAM Test (affected by prompt)	4-2
Section 5: System Board Tests	5-1
5.1 Introduction	5-1
5.2 ASIC Tests	5-1
5.2.1 Test 4 - I/O ASIC Register Access Test	
5.2.2 Test 5 - I/O ASIC Reset Tests	
5.2.3 Test 6 - I/O ASIC FIFO/ECC Test	5-3
5.3 LANCE (7990) Tests	5-4
5.3.1 Test 7 - 7990 LANCE Initialization Test	5-4
5.3.2 Test 8 - 7990 LANCE Internal Loopback Test	
5.3.3 Test 9 - 7990 LANCE External Loopback Test (must be prompted)	
5.3.4 Test 10 - 7990 LANCE Data Alignment Test	
5.3.5 Test 11 - 7990 LANCE Cacheable Data Merge Test	5-6
5.3.6 Test 12 - 7990 LANCE Address/Cache Data Test (affected by prompt)	
5.3.7 Test 13 - 7990 LANCE Cache Block Buswatcher Test	
5.4 SCSI (33C93) Tests	
5.4.1 Test 14 - 33C93 SBIC (SCSI) Data Path (Write Buffer) Test	
5.4.2 Test 15 - SCSI Cacheable Block Data Merge Test	
5.4.3 Test 16 - SCSI Multi-bit ECC Error Test	
5.4.4 Test 17 - SCSI Page Overflow Bit Test	
5.4.5 Test 18 - SCSI Address Counter/Cache Data Test	5-13

5.4.6 Test 19 - SCSI Cache Block Buswatcher Test	5-13
5.4.7 Test 20 - SCSI Cache Block Flush Test	5-14
5.5 Test 21 - SCSI/LANCE Data Transfer Test	5-14
5.6 VMEbus Interface Tests	5-15
5.6.1 Test 22 - VMEbus Address Map RAM Test	
5.6.2 Test 23 - VMEbus Data Path Test	
5.6.3 Test 24 - VMEbus Address Path Test	
5.6.4 Test 25 - VMEbus Data Multiplexing Test	
5.6.5 Test 26 - VMEbus Interrupt (IACK) Test	5-18
5.6.6 Test 27 - RF3500 SCSI Data Path (Write Buffer) Test	
5.6.7 Test 28 - VMEbus Cacheable Data Merging Test	
5.6.8 Test 29 - VMEbus Block Mode Data Merging Test	
5.6.9 Test 30 - VMEbus Cacheable Data Buffer Test	5-23
5.6.10 Test 31 - VMEbus Ping-Pong Data Buffers Test	
5.6.11 Test 32 - VMEbus Block Mode Address Counter Test	
5.6.12 Test 33 - VMEbus Panic Interrupt Test	
5.7 BW20 Tests	
5.7.1 Test 34 - Frame Buffer RAM Test (affected by prompt)	5-25
5.7.2 Test 35 - Frame Buffer Interrupt and Interrupt Registers Test	
5.8 Serial Port Tests	
5.8.1 Test 36 - Serial Ports Reset Test	
5.8.2 Test 37 - Serial Ports Internal Loopback Test	
5.8.3 Test 38 - System Timer Test	
5.8.4 Test 39 - Profile Timer Test	
5.9 Test 40 - RTC-58321 Real Time Clock Test	
5.5 1C5t 40 - KTC-50521 Kear Time Clock Test	5-51
Section 6: Graphics Tests	6-1
6.1 Introduction	
6.2 CG40 Tests	
6.2.1 Bt458 Tests	-
6.2.1.1 Test 41 - Bt458 Registers Test	
6.2.1.2 Test 42 - Bt458 Address Register (Autoincrement) Test	
6.2.1.3 Test 43 - Bt458 Main CLUT (Colormap) Test	
6.2.1.4 Test 44 - Bt458 Overlay CLUT (Colormap) Test	
6.2.2 Test 45 - Frame Buffer RAM Test (affected by prompt)	6-5
6.2.3 Test 46 - Frame Buffer Interrupt and Interrupt Registers Test	
6.3 CG30 Tests	
6.3.1 Registers Tests	
6.3.1.1 Test 47 - I/O Location/Control Status Registers Test	
6.3.1.2 Test 48 - I/O Location (Space bits) Register Test	
6.3.1.3 Test 49 - 8-bit Registers Test	
6.3.1.4 Test 50 - 16-bit Registers Test	
6.3.1.5 Test 51 - Dummy Registers Access Test	6-11
6.3.2 Bt458 Tests	
6.3.2.1 Test 52 - Bt458 Registers Test	6.12
6.3.2.2 Test 53 - Bt458 Address Register (Autoincrement) Test	6-12
6.3.2.3 Test 54 - Bt458 Main CLUT (Colormap) Test	6.14
6.3.2.4 Test 55 - Bt458 Overlay CLUT (Colormap) Test	0-14 2 1E
6.3.2.5 Test 56 - Colormap Pattern Test (must be prompted)	0-10
6.3.2.6 Test 57 - Frame Buffer to Colormap Test	n-1/
()) 7 Test EQ (Calashan Test Detter (
6.3.2.7 Test 58 - Colorbar Test Pattern (must be prompted) 6.3.3 Bt431 Tests	6-18

6.3.3.1 Test 59 - Address Register (Autoincrement) Test	6-1 8
6.3.3.2 Test 60 - Single Cursor Control Registers Test	6-19
6.3.3.3 Test 61 - Both Cursors to Single Cursor Control Registers Test	6-21
6.3.3.4 Test 62 - Single Cursor Pattern RAM Test	6-22
6.3.3.5 Test 63 - Both Cursors to Single Cursor Pattern RAM Test	6-23
6.3.4 Shadow RAM Tests	6-24
6.3.4.1 Test 64 - Shadow RAM Test	6-2 5
6.3.4.2 Test 65 - Shadow RAM to Bt458 Colormap Update Test	6-2 6
6.3.5 Frame Buffer RAM Tests	
6.3.5.1 Test 66 - Plane Major Mode Test (affected by prompt)	6-28
6.3.5.2 Test 67 - Pixel Major Mode Test (affected by prompt)	
6.3.5.3 Test 68 - Pixel Major to Plane Major Test	6-31
6.3.6 ROP Tests	
6.3.6.1 Registers Tests	
6.3.6.1.1 Test 69 - Single Plane Registers Test	
6.3.6.1.2 Test 70 - All Planes to Single Plane Registers Test	6-34
6.3.6.1.3 Test 71 - BTLA Mode Single Plane Registers Test	6-35
6.3.6.1.4 Test 72 - BTLA Mode All Planes to Single Plane Registers Test	
6.3.6.2 Mode Tests	
6.3.6.2.1 Test 73 - Mode-0 Read Access Test	
6.3.6.2.2 Test 74 - Mode-1 Read Access Test	
6.3.6.2.3 Test 75 - Mode-4 Read Access Test	
6.3.6.2.4 Test 76 - Mode-5 Read Access Test	
6.3.6.2.5 Test 77 - Mode-6 Read Access Test	
6.3.6.2.6 Test 78 - Mode-0 Write Access Test	
6.3.6.2.7 Test 79 - Mode-1 Write Access Test	
6.3.6.2.8 Test 80 - Mode-2 Write Access Test	
6.3.6.2.9 Test 81 - Mode-3 Write Access Test	
6.3.6.2.10 Test 82 - Mode-4 Write Access Test	
6.3.6.2.11 Test 83 - Mode-5 Write Access Test	
6.3.6.2.12 Test 84 - Mode-6 Write Access Test	
6.3.6.2.13 Test 85 - Mode-7 Write Access Test	
6.3.7 Retrace Interrupts Tests	
6.3.7.1 Test 86 - Vertical Retrace Interrupts Test	
6.3.7.2 Test 87 - Framecount Register (Counter) Test	
6.3.7.3 Test 88 - Retrace/SCC Interrupt Combination Test	
6.3.8 Serial Ports Tests	
6.3.8.1 Test 89 - Serial Ports Reset Test	6-67
6.3.8.2 Test 90 - Serial Ports Loopback Test	6-68
-	•
Section 7: Miscellaneous Tests	
7.1 Introduction	
7.2 Test 91 - Random ECC Generation Test (must be prompted)	
7.3 Test 92 - Cache Data Bus Test (must be prompted)	
7.4 Test 93 - RS-232-C Connector Loopback Test (must be prompted)	
7.5 Test 94 - IDPROM checksum tests (affected by prompt)	
7.6 Test 95 - Ethernet TFTP Read Test	
7.7 Test 96 - Disk Write/Read Test (affected by prompt)	
7.8 Test 97 - Tape Write/Read Test (must be prompted)	
7.9 Test 98 - NMI and Test Switch Test (must be prompted)	7-22
Castian & Common de	0.1
Section 8: Commands	8-1

8.1 Introduction	
Appendix A: MARCH Algorithm	A-1
A.1 Introduction	A-1
Appendix B: Moving Inversions Test Algorithm	B-1
B.1 Introduction	B-1
Appendix C: Frame Buffer Test Modes	C-1
C.1 Introduction	C-1

۰

List of Figures

Figure 3-1.	The dg menu Structure	3-3
Figure 3-2.	Tests and Test Submenus	3-4
Figure 5-1.	Pin Layout for Transceiver Connector on EXOS 202	5-5
Figure 7-1.	Solbourne Serial Port Loopback Model	7-4
Figure 7-2.	Null Modem Cable Pin Arrangement	7 -5
Figure 7-3.	Solbourne Serial Port Loopback Model	7-8
Figure 7-4.	Null Modem Cable Pin Arrangement	7-8

.

.

Section 1: Introduction

1.1 Introduction

dg is a standalone test controller for the Solbourne system. This program is used by both manufacturing and field engineering personnel to help determine which printed circuit board is defective. The primary user of this program is manufacturing.

The software for dg includes:

- The dg standalone/test controller (dg (1))
- Test control commands
- Miscellaneous commands
- dg tests

1.2 Related Documentation

Information that may be useful while using the dg program is available in the following documentation:

- Series4/600 Service Manual, Part number 101249-AA
- Series4/500 Service Manual, Part number 102161-AA
- Series4/600 Theory Manual, Part number 101250-AA
- Bootable/Standalone Multiprocessor Diagnostics Manual, Part number 101686-AB
- Extended ROM Resident Diagnostics Manual, Part number 101489-AB
- System Power On Self Test Manual, Part number 101486-AB

•

Section 2: Getting Started with dg

2.1 Introduction

This section gives step-by-step instructions and examples for getting started using dg. Additional information on using the tests from the hierarchal menu system of the dg tests is given in Section 3.

In this section, commands you enter are given in **boldface** type. Command parameters for which you substitute a value are given in *italic*. Screen output is simulated in constantwidth font.

2.2 Invoking the dg Program

The steps to follow the first time dg is invoked are given below.

The user must first bring the Solbourne system to the ROM> prompt. If UNIX is running, it must be shutdown using the halt (1) command.

1. At the ROM> prompt, type:

dg may also be invoked as follows:

If -i file is appended to the boot command, the specified file is opened and dg's internal command buffer is initialized with its contents. For additional information, refer to the fbuf(1) and xbuf(1) commands.

2. When dg is invoked, the following message is displayed:

```
DG - Standalone Diagnostic Test Controller
Version 1.1 August 1, 1989
Copyright (c) 1989 Solbourne Computer, Inc.
```

At start-up, the following tables are initialized:

- The memory configuration table and memory limits are initialized with the values found by the ROM during self-test. (See the config (1) command.)
- The frame buffer configuration table is initialized with the values found by the ROM during self-test and the default frame buffer defaults to the frame buffer in the highest numbered slot. (See the fbconfig (1) command.)
- The VMEbus table comes up as empty. (See the vmeconfig(1) command.)

2.3 Entering Commands to the dg Prompt

dg accepts input when the

DG>

prompt is displayed.

Commands and parameters are case insensitive.

Other rules for entering commands include:

1. In general more than one command can be entered in a single command line to the DG> prompt at the same time.

DG> tests 1 2 3 names on passlim 0 between 5 run

The above command line selects tests 1, 2, and 3, turns the printing of test names on, sets the pass limit to 0 (no passlim), the between count is set to 5, and begins test execution with the **run** command.

☆ ☆ ☆ NOTE ☆ ☆ ☆

It is important to remember that error messages from one test are not valid, if failures have occurred during previous tests. The errors from a test must be corrected before advancing to the next test.

- 2. Commands that process user input in an interactive mode, such as vmeconfig(1) and fbconfig(1), cause commands that follow on the command line to be ignored.
- 3. Commands must be separated by white space(s), including tabs or spaces. (Semicolons are not recognized by dg as spaces.)
- 4. If any of the command(s) entered return an error condition, all following commands are ignored and the DG> prompt is redisplayed.
- 5. If a command is unrecognized by dg, the following is displayed:

Unknown command (command name)

- 6. All command lines are terminated by a Return.
- 7. Some commands may display additional error messages if numeric values are entered incorrectly or if the numeric values are not legal. These messages identify the value that is out of range, for example

illegal address (value given)

If an illegal value is given, additional information may be displayed that identifies the

legal range of values.

- 8. Memory and I/O addresses and contents must be entered in hexidecimal format. Any value that has to do with hardware must also be entered in hexidecimal (e.g., register data, memory address, or memory data).
- 9. Counters and test numbers should be entered in decimal format (e.g., counts and limits).
- 10. The dg help(1) command can be used any time the DG> prompt is displayed. A summary of the command given as an argument to help will be displayed.

2.4 Using dg Commands

Example usage of each dg command is given in Section 8 of this manual. All commands can be used with any other commands.

2.4.1 dg Commands

dg command names and their functions follow:

- between (1) Set or display between count
- cd (1) Change to a different test directory
- config (1) Generate or display memory configuration file
- continue (1) Set or display continue on error flag
- deposit (1) Deposit data at specified address
- errlim (1) Set or display error limit
- errors (1) Display error count
- examine (1) Examine contents of memory
- fbconfig (1) Generates (or modifies) the frame buffer configuration file
- fbuf(1) Fill internal command buffer
- help (1) Display this command list or information on a specific command
- limit (1) Display or set memory test limits
- loop (1) Set or display loop on test flag
- ls (1) List contents of test directory
- menu (1) Display listing of available tests
- names (1) Enable or disable printing of test names during test execution
- next(1) Execute next selected test
- passes (1) Display pass count
- passlim (1) Set or display pass limit
- prompt(1) Set or display prompt flags
- quiet(1) Set or display error message enable flag

- quit(1) Exit from dg debugger program
- restart (1) Restart execution of selected tests
- run(1) Start execution of selected tests
- status (1) Display or reset state of modes, flags, and counts
- screenload (1) Loads a raster image file into the specified frame buffer
- tests (1) Select or display tests to be executed
- time (1) Set or display print time flag and print current date and time
- vmeconf (1) Configure VMEbus devices
- what (1) Display information about Kbus boards installed in system
- xbuf(1) Load, display, save, or execute the contents of the command buffer

2.5 Starting Test Execution

By default, when dg is invoked all the tests are selected. Tests are executed when the run(1) command is entered at the command line. For example:

DG> run

If the tests command is entered (without an argument, all the selected tests are displayed. For example:

DG> tests									
selected tests:	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	50	51	52	53	54	55	56	
	57	58	59	60	61	62	63	64	
	65	66	67	68	69	70	71	72	
	73	74	75	76	77	78	79	80	
	81	82	83	84	85	86	87	88	
	89	90	91	92	93	94	95	96	
	97	98							

The test selection can be modified any time the DG> prompt is displayed by using the tests command. For example:

```
DG> tests 9 8 4
DG> tests
selected tests: 9 8 4
DG>
```

2.5.1 Using the menu Command

The menu command identifies the test names or their functions. For example:

```
DG> menu
Menu of installed test programs (==> denotes menu):
    ==> Memory Tests
    ==> IO Tests
    ==> Graphics Tests
    ==> Miscellaneous Tests
```

Displays the top level of the menu directory. The menu command followed by the all argument gives a listing of all the tests and the menu directory in which they reside, as follows:

```
DG> menu all
Menu of installed test programs (==> denotes menu):
  ==>
      Memory Tests
     1: Cache block virtual alias test (affected by prompt)
     2: Memory Data RAM test (affected by prompt)
     3: Memory ECC RAM test (affected by prompt)
       IO Tests
  ==>
   ==>
        ASIC Tests
        4: I/O ASIC register access test
        5: I/O ASIC reset tests
        6: I/O ASIC FIFO/ECC test
          LANCE (7990) Tests
     ==>
         7: 7990 LANCE initialization test
         8: 7990 LANCE internal loopback test
         9: 7990 LANCE external loopback test (must be prompted)
        10: 7990 LANCE data alignment test
```

Continued on following page

 $(\mathbf{A}, \mathbf{A}) = (\mathbf{A}, \mathbf{A}) = (\mathbf{A}, \mathbf{A})$

`.z

Solbourne Confidential Information - Do Not Distribute

Continued from previous page 11: 7990 LANCE cacheable data merge test 12: 7990 LANCE address/cache data test (affected by prompt) 13: 7990 LANCE cache block buswatcher test ==> SCSI (33C93) Tests 14: 33C93 SBIC (SCSI) data path (Write Buffer) test 15: SCSI cacheable block data merge test 16: SCSI Multi-bit ECC error test 17: SCSI page overflow bit test 18: SCSI address counter/cache data test 19: SCSI cache block buswatcher test 20: SCSI cache block flush test 21: SCSI/LANCE data transfer test VMEbus interface Tests ==> 22: VMEbus address map RAM test 23: VMEbus data path test 24: VMEbus address path test 25: VMEbus data multiplexing test 26: VMEbus Interrupt (IACK) test 27: RF3500 SCSI data path (Write Buffer) test 28: VMEbus cacheable data merging test 29: VMEbus block mode data merging test 30: VMEbus cacheable data buffer test 31: VMEbus ping-pong data buffers test 32: VMEbus block mode address counter test 33: VMEbus panic interrupt test BW20 Tests *****=> 34: Frame Buffer RAM Test (affected by prompt) 35: Frame Buffer interrupt and Interrupt Registers Tests Serial Ports Tests ==> 36: Serial ports reset test 37: Serial ports internal loopback test 38: System Timer test 39: Profile Timer test 40: RTC-58321 real time clock test Graphics Tests ==> CG40 Tests ==> ==> Bt458 Tests 41: Registers Test 42: Address Register (Autoincrement) Test 43: Main CLUT (Colormap) Test 44: Overlay CLUT (Colormap) Test 45: Frame Buffer RAM Test (affected by prompt) 46: Frame Buffer Interrupt and Interrupt Registers Tests Continued on following page

Continued from previous page CG30 Tests ==> ==> **Registers** Tests 47: IO location/control status registers test 48: IO location (space bits) register test 49: 8-bit registers test 50: 16-bit registers test 51: Dummy registers access test ==> Bt458 Tests 52: Registers test 53: Address register (autoincrement) test 54: Main CLUT (Colormap) test 55: Overlay CLUT (Colormap) test 56: Colormap pattern test (must be prompted) 57: Frame buffer to colormap test 58: Colorbar test pattern (must be prompted) ==> Bt431 Tests 59: Address register (autoincrement) test 60: Single cursor control registers test 61: Both cursors to single cursor control registers test 62: Single cursor pattern RAM test 63: Both cursors to single cursor pattern RAM test ==> Shadow RAM Tests 64: Shadow RAM test 65: Shadow RAM to Bt458 colormap update test ==> Frame Buffer RAM Tests 66: Plane major mode test (affected by prompt) 67: Pixel major mode test (affected by prompt) 68: Pixel major to plane major test ==> ROP Tests **Registers Tests** ==> 69: Single plane registers test 70: All planes to single plane registers test 71: BTLA mode single plane registers test 72: BTLA mode all planes to single plane registers test ==> Mode Tests 73: Mode-0 read access test 74: Mode-1 read access test 75: Mode-4 read access test 76: Mode-5 read access test 77: Mode-6 read access test 78: Mode-0 write access test 79: Mode-1 write access test 80: Mode-2 write access test 81: Mode-3 write access test 82: Mode-4 write access test Continued on following page

Solbourne Confidential Information - Do Not Distribute

Continued from previous page
83: Mode-5 write access test
84: Mode-6 write access test
85: Mode-7 write access test
==> Retrace Interrupts Tests
86: Vertical retrace interrupts test
87: Framecount register (counter) test
88: Retrace/SCC interrupt combination test
==> Serial Ports Tests
89: Serial ports reset test
90: Serial ports loopback test
==> Miscellaneous Tests
91: Random ECC generation test (must be prompted)
92: Cache data bus test (must be prompted)
93: RS232 connector loopback test (must be prompted)
94: IDPROM checksum tests (affected by prompt)
95: Ethernet tftp read test
96: Disk write/read test (affected by prompt)
97: Tape write/read test (must be prompted)
98: NMI and Test switch test (must be prompted)
DG>

For more information on the hierarchal menu system of dg and test execution, refer to Section 3 and also see the tests (1), run(1), and menu (1) commands in Section 8.

2.5.2 Variations of Test Execution

This subsection discusses some of the basic variations that can be applied to test commands. There are other variations than those given here.

Two results can occur during test execution. The test can pass or the test can fail.

If the test passes, the user may use any of the available **dg** commands to "customize" the test selection and execution modes, or may exit **dg** by entering the **quit**(1) command. The available options to the user include:

- Select other tests for execution
- Tell the controller how many iterations to run using the **passlim** (1) command followed by the **restart** (1) command
- Controls whether the test names are printed using the names (1) command
- The user can also stop test execution at any time by entering a Control-C (°C). Since the tty drivers at the standalone level poll for input there may be a delay between the time the °C is entered and when the test calls the driver routine to check for input. The user should be patient and not repeatedly enter the °C command.

If the test fails, the user can do any of the following:

• Set up a oscilloscope loop by using the loop (1) and quiet (1) commands

- Continue the failing test with the run command
- Skip to the next test in the selected sequence of tests using the next (1) command
- Restart the entire sequence using the restart (1) command
- Exit dg using the quit command

2.6 Handling Test Failures

Several of the commands given in Section 8 that are used for test control can be used when test failures occur. In the following example, test 1 detects a failure and the loop and quiet commands are used to set up a scope loop.

```
DG> tests 2 run
Starting Test 2: Cache Block Virtual Alias Test (affected by prompt)
TEST 2 ERROR: Thu Dec 1 10:03:05 1988
Data error on vaddr 0x10000, paddr 0xff040000
exp = 0xff040000
act = 0xff050000
DG>
```

Note that test 2 has displayed its error message which identified the failing test case and returned to the DG> prompt. If the user wishes to evaluate this test failure by setting up a scope loop, the sequence of commands shown in the following illustration may be entered.

```
DG> loop on run

TEST 2 ERROR: Thu Dec 1 10:03:05 1988

Data error on vaddr 0x10000, paddr 0xff040000

exp = 0xff040000

act = 0xff050000

DG>
```

Note that test 2 has repeated the failing test case and has redisplayed the same error message. This suggests the presence of a solid failure. To speed up the loop and avoid having to reenter the run command, the sequence of commands in the following illustration may be entered. DG> quiet on run

(No information is displayed while the test is looping when the quiet command is invoked.)

^C
Test interrupted, type ``run'' to resume
DG>

No information is displayed while the loop and quiet flags are set. The program is not hung. It is executing in the tightest possible loop of the failing test case. A Control-C must be entered to halt the loop and return to the DG> prompt.

ኇ ኳ ኳ NOTE ኳ ኳ ኳ

It is a common mistake to forget to reset the loop and quiet flags before restarting the test sequence. This causes the first test case in the first test to be executed in a tight loop. See the loop and quiet commands in Section 8 for additional information.

2.7 Exiting dg

To exit **dg** use the **quit**(1) command.

Section 3: Overview of dg Tests

3.1 Introduction

This section presents an overview of the dg tests. For instructions on executing tests, refer to Section 2 and the run(1) and tests (1) commands in Section 8.

The information given in this section includes:

- An overview of the menu structure
- A numerical listing of all dg tests

3.2 Overview of the Menu Structure

The dg menu of tests is similar to the hierarchal tree-like structure of the UNIX file system. Figure 3-1 illustrates the menu structure of dg.



Figure 3-1. The dg menu Structure

Moving about the dg menu structure has been made easier by the installation of the UNIX-type commands cd (1) and ls (1). Refer to Section 2 and Section 8 for additional information.

Figures 3-2 and 3-3 show where the dg tests reside in the menu structure. The test names in these illustrations have been shortened. To see the full name, refer to Section 3.4.



Figure 3-2. Tests and Test Submenus



Figure 3-2. Tests and Test Submenus (Continued)

3.3 Numerical Test Listing

A numerical listing of all the dg tests is given below. The test number is given on the left side, followed by the path to the test.

- 1. memory/Cache block virtual alias test (affected by prompt)
- 2. memory/Memory Data RAM test (affected by prompt)
- 3. memory/Memory ECC RAM test (affected by prompt)
- 4. IO/ASIC/I/O ASIC register access test
- 5. IO/ASIC/I/O ASIC reset tests

- 6. IO/ASIC/I/O ASIC FIFO/ECC test
- 7. IO/ASIC/LANCE 7990/7990 LANCE initialization test
- 8. IO/ASIC/LANCE 7990/7990 LANCE internal loopback test
- 9. IO/ASIC/LANCE 7990/7990 LANCE external loopback test (must be prompted)
- 10. IO/ASIC/LANCE 7990/7990 LANCE data alignment test
- 11. IO/ASIC/LANCE 7990/7990 LANCE cacheable data merge test
- 12. IO/ASIC/LANCE 7990/7990 LANCE address/cache data test (affected by prompt)
- 13. IO/ASIC/LANCE 7990/7990 LANCE cache block buswatcher test
- 14. IO/ASIC/SCSI 33C93/33C93 SBIC (SCSI) data path (Write Buffer) test
- 15. IO/ASIC/SCSI 33C93/SCSI cacheable block data merge test
- 16. IO/ASIC/SCSI 33C93/SCSI Multi-bit ECC error test
- 17. IO/ASIC/SCSI 33C93/SCSI page overflow bit test
- 18. IO/ASIC/SCSI 33C93/SCSI address counter/cache data test
- 19. IO/ASIC/SCSI 33C93/SCSI cache block buswatcher test
- 20. IO/ASIC/SCSI 33C93/SCSI cache block flush test
- 21. IO/ASIC/SCSI/LANCE data transfer test
- 22. IO/ASIC/VMEbus/VMEbus address map RAM test
- 23. IO/ASIC/VMEbus/VMEbus data path test
- 24. IO/ASIC/VMEbus/VMEbus address path test
- 25. IO/ASIC/VMEbus/VMEbus data multiplexing test
- 26. IO/ASIC/VMEbus/VMEbus Interrupt (IACK) test
- 27. IO/ASIC/VMEbus/RF3500 SCSI data path (Write Buffer) test
- 28. IO/ASIC/VMEbus/VMEbus cacheable data merging test
- 29. IO/ASIC/VMEbus/VMEbus block mode data merging test
- 30. IO/ASIC/VMEbus/VMEbus cacheable data buffer test
- 31. IO/ASIC/VMEbus/VMEbus ping-pong data buffers test
- 32. IO/ASIC/VMEbus/VMEbus block mode address counter test
- 33. IO/ASIC/VMEbus/VMEbus panic interrupt test
- 34. IO/BW20/Frame Buffer RAM Test (affected by prompt)
- 35. IO/BW20/Frame Buffer interrupt and Interrupt Registers Tests
- 36. IO/BW20/Serial ports reset test
- 37. IO/Serial Port/Serial ports internal loopback test
- 38. IO/Serial Port/System Timer test
- 39. IO/Serial Port/Profile Timer test

- 40. IO/RTC-58321 real time clock test
- 41. Graphics/CG40/BT458/Registers Test
- 42. Graphics/CG40/BT458/Address register (autoincrement) test
- 43. Graphics/CG40/BT458/Main CLUT (colormap) test
- 44. Graphics/CG40/BT458/Overlay CLUT (colormap) test
- 45. Graphics/CG40/BT458/Colormap pattern test (must be prompted)
- 46. Graphics/CG40/BT458/Frame buffer to colormap test
- 47. Graphics/CG40/BT458/Colorbar Test Pattern (must be prompted)
- 48. Graphics/CG40/Frame Buffer RAM Test (affected by prompt)
- 49. Graphics/CG40/Frame Buffer Interrupt and Interrupt Registers Tests
- 50. Graphics/CG30/Register Tests/IO Location/Control Status Registers Test
- 51. Graphics/CG30/Register Tests/IO location (space bits) register test
- 52. Graphics/CG30/Register Tests/8-bit registers test
- 53. Graphics/CG30/Register Tests/16-bit registers test
- 54. Graphics/CG30/Register Tests/Dummy registers access test
- 55. Graphics/CG30/Bt458/Registers test
- 56. Graphics/CG30/Bt458/Address register (autoincrement) test
- 57. Graphics/CG30/Bt458/Main CLUT (colormap) test
- 58. Graphics/CG30/Bt458/Overlay CLUT (colormap) test
- 59. Graphics/CG30/Bt431/Address register (autoincrement) test
- 60. Graphics/CG30/Bt431/Single cursor control registers test
- 61. Graphics/CG30/Bt431/Both cursors to single cursor control registers test
- 62. Graphics/CG30/Bt431/Single cursor pattern RAM test
- 63. Graphics/CG30/Bt431/Both cursors to single cursor pattern RAM test
- 64. Graphics/CG30/Shadow RAM/Shadow RAM test
- 65. Graphics/CG30/Shadow RAM/Shadow RAM to Bt458 colormap update test
- 66. Graphics/CG30/Frame Buffer/Plane major mode test (affected by prompt)
- 67. Graphics/CG30/Frame Buffer/Pixel major mode test (affected by prompt)
- 68. Graphics/CG30/Frame Buffer/Pixel major to plane major test
- 69. Graphics/CG30/ROP/Register/Single plane registers test
- 70. Graphics/CG30/ROP/Register/All planes to single plane registers test
- 71. Graphics/CG30/ROP/Register/BTLA mode single plane registers test
- 72. Graphics/CG30/ROP/Register/BTLA mode all planes to single plane registers test
- 73. Graphics/CG30/ROP/Mode/Mode-0 Read access test

- 74. Graphics/CG30/ROP/Mode/Mode-1 Read access test
- 75. Graphics/CG30/ROP/Mode/Mode-4 Read access test
- 76. Graphics/CG30/ROP/Mode/Mode-5 Read access test
- 77. Graphics/CG30/ROP/Mode/Mode-6 Read access test
- 78. Graphics/CG30/ROP/Mode/Mode-0 Write access test
- 79. Graphics/CG30/ROP/Mode/Mode-1 Write access test
- 80. Graphics/CG30/ROP/Mode/Mode-2 Write access test
- 81. Graphics/CG30/ROP/Mode/Mode-3 Write access test
- 82. Graphics/CG30/ROP/Mode/Mode-4 Write access test
- 83. Graphics/CG30/ROP/Mode/Mode-5 Write access test
- 84. Graphics/CG30/ROP/Mode/Mode-6 Write access test
- 85. Graphics/CG30/ROP/Mode/Mode-7 Write access test
- 86. Graphics/CG30/Retrace Interrupt/Vertical retrace interrupts test
- 87. Graphics/CG30/Retrace Interrupt/Framecount register (counter) test
- 88. Graphics/CG30/Retrace Interrupt/Retrace/SCC interrupt combination test
- 89. Graphics/CG30/Serial Ports/Serial ports reset test
- 90. Graphics/CG30/Serial Ports/Serial ports loopback test
- 91. Miscellaneous/Random ECC generation test (must be prompted)
- 92. Miscellaneous/Cache data bus test (must be prompted)
- 93. Miscellaneous/RS232 connector loopback test (must be prompted)
- 94. Miscellaneous/IDPROM checksum test (affected by prompt)
- 95. Miscellaneous/Ethernet tftp read test
- 96. Miscellaneous/Disk write/read test (affected by prompt)
- 97. Miscellaneous/Tape write/read test (must be prompted)
- 98. Miscellaneous/NMI and test switch test (must be prompted)



Section 4: Memory Tests

4.1 Introduction

The dg tests available for memory testing are listed in this section.

4.2 Test 1 - Cache Block Virtual Alias Test (affected by prompt)

This test verifies that the virtual cache tags will reference the correct entry in the cache RAMs. This test also verifies that a reference to the same physical location through another virtual reference is invalidated (i.e., aliases are invalidated).

```
Data error at vaddr 0x10000, paddr 0xff040000
exp = 0xff040000
act = 0xff050000
```

This test performs a series of memory page writes and reads in which all physical page address from XXXX0000 to XXXXe000 (hex) are written and read using all combinations of virtual page addresses from XXXX0000 to XXXXe000 including FF space addresses.

A write to a physical page using a virtual page address creates a unique physical-to-virtual mapping within the MMU. When the physical page is accessed using a different virtual page address, the MMU must break the existing physical-to-virtual mapping so that a new one may be created. To do this, the MMU performs a "purge" operation in which the cached data block is written back to the physical location in memory and re-read into the cache at the new virtual index. This creates the new physical-to-virtual mapping.

This test insures that the MMU logic which detects the purge condition is operational and that the data (unique for each physical page) is correctly transferred between the cache and the memory system.

If the prompt flag for this test is set, the test prints out status messages that indicate which physical and logical addresses are being used (however this slows thes execution).

4.3 Test 2 - Memory Data RAM Test (affected by prompt)

This test is an extended version of the ROM power-up self-test Addressing and Data Test. The test program performs a MARCH test algorithm to verify the addressing and data paths (see the Appendix A for information on the MARCH test algorithm).

The test program determines the test area from the Kbus memory limit entry in the memory limit table (refer to the limit (1) command to modify the test area).

The total test area is blocked into eight megabytes (or less) blocks which are tested separately.

During the read-write-read sequence, the target memory block is cached and checked for correct data. The data in the cache is then complemented and the block is flushed back to memory. The target block is then re-read and verified to contain the complemented data.

Legal error codes for the Data RAM Test are:

- 0x00 Data fault exception occurred during write of memory with initial data pattern
- 0x10 Data fault exception occurred during flush of a memory block on write of memory with initial data pattern
- 0x20 Data fault exception occurred on first read of forward pass
- 0x80 Data miscompare occurred on first read on forward pass
- 0x30 Data fault exception occurred during flush of target memory block back to memory during forward pass
- 0x40 Data fault exception occurred on second read of forward pass
- 0xa0 Data miscompare occurred on second read on forward pass
- 0x50 Data fault exception occurred on first read of reverse pass
- 0xc0 Data miscompare occurred on first read on reverse pass
- 0x60 Data fault exception occurred during flush of target memory block back to memory during reverse pass
- 0x70 Data fault exception occurred on second read of reverse pass
- 0xe0 Data miscompare occurred on second read on reverse pass

An example of a Data RAM Test failure follows:

```
Error occurred in data RAM memory test
Error code = 0xe0 Virtual addr = 0x00800000
Physical addr = 0x00ea0000 Board slot = 2
A data failure was found in the second read on the reverse pass.
exp = 0x55555555
act = 0x5555555
act = 0x00000008
```

4.4 Test 3 - Memory ECC RAM Test (affected by prompt)

The test is an extended version of the ROM power-up self-test Addressing and Data Test. The test program performs a MARCH test algorithm to verify the addressing and data paths (see Appendix A for information on the MARCH test algorithm).

The Error Correction Code (ECC) test uses double word stores and loads to/from memory, the data RAM test uses word.

The test program determines the test area from the Kbus memory limit entry in the memory limit table (refer to the limit (1) command to modify the test area).

The total test area is blocked into 8 megabyte (or less) blocks which are tested separately.

During the read-write-read sequence, the target memory block is cached and checked for correct data. The data in the cache is then complemented and the block is flushed back to memory. The target block is then re-read and verified to contain the complemented data.

Legal error codes for the ECC test are:

- 0x00 Data fault exception occurred during write of memory with initial data pattern
- 0x10 Data fault exception occurred during flush of a memory block on write of memory with initial data pattern
- 0x20 Data fault exception occurred on first read of forward pass
- 0x80 Data miscompare occurred on high word in first read on forward pass
- 0x90 Data miscompare occurred on low word in first read on forward pass
- 0x30 Data fault exception occurred during flush of target memory block back to memory during forward pass
- 0x40 Data fault exception occurred on second read of forward pass
- 0xa0 Data miscompare occurred on high word in second read on forward pass
- 0xb0 Data miscompare occurred on low word in second read on forward pass
- 0x50 Data fault exception occurred on first read of reverse pass
- 0xc0 Data miscompare occurred on high word in first read on reverse pass
- 0xd0 Data miscompare occurred on low word in first read on reverse pass
- 0x60 Data fault exception occurred during flush of target memory block back to memory during reverse pass
- 0x70 Data fault exception occurred on second read of reverse pass
- 0xe0 Data miscompare occurred on high word in second read on reverse pass
- 0xf0 Data miscompare occurred on low word in second read on reverse pass

An example of an ECC test failure follows:

```
Error occurred in ECC RAM memory test

Error code = 0x40 Virtual addr = 0x00800000

Physical addr = 0x00ea0000 Board slot = 2

An exception occurred after re-read of a double on the forward pass

Exception type = data fault, FCR = 0x02, Syndrome = 0x00
```

•

Section 5: System Board Tests

5.1 Introduction

The dg tests available for System Board are listed in this section.

5.2 ASIC Tests

The following tests verify that the I/O ASIC registers can be accessed by the CPU and that the I/O ASIC can fetch good data from memory.

5.2.1 Test 4 - I/O ASIC Register Access Test

This test verifies that the I/O ASIC will retain data in internal registers and verify the access to the Western Digital (33C93) SCSI Bus Interface Controller (SBIC) and the AMD 7990 Local Area Network Controller for Ethernet (LANCE) chips. Refer to the technical manuals of these devices for additional information.

For each test, the invlid bits are masked before comparing data.

The first part of this test checks all I/O ASIC registers for data retention. Each register has a series of patterns written, read back and verified.

```
Data error in Ethernet receive address register
  exp = 0x00000000
  act = 0x00000040
```

The test continues by verifying that each of the I/O ASIC's five (5) read/write rergisters are unique. The test sequence for this test is as follows:

```
clear all registers
writing the test register to all ones
check all for proper data
write all registers to all ones
clear the test register
check all for proper data
```

The above example is repeated for all registers as the test register.

```
Data retention error in test register
test register = Ethernet receive address register
   exp = 0x0001ffff
   act = 0x0001ffef
```

In the above example, the test registers did not retain the value written to it.

```
Addressing error on ASIC
test register = SCSI DMA address register
value written = 0x00000000
register modified = SCSI control/address register
exp = 0x00007ffd
act = 0x00000000
```

In the above example, writing to the SCSI DMA address register caused the SCSI control/address register to be modified.

The next two tests verify accessability and data retention of the AMD 7990 LANCE chip and the WD 33C93 SBIC chip. Each device is presented a pattern and read back. The LANCE has a 16 bit data bus and the 33C93 has an 8 bit.

The LANCE test uses CSR1 for the test.

```
Data retention error in LANCE chip
  exp = 0xfffe
  act = 0xfefe
```

The 33C93 test uses the total cylinders, low byte, register for the test.

```
Data retention error in SCSI chip
exp = 0x08
act = 0x00
```

5.2.2 Test 5 - I/O ASIC Reset Tests

This test verifies the reset circuitry associated with the I/O ASIC. Three resets are available:

1. Reset the I/O ASIC
- 2. Reset the 33C93 SCSI Controller
- 3. Reset the VMEbus

These resets are an address that is strobed. There is a bit for each device to reset. This test verifies that each reset works and that each reset does not cause another reset to occur. (At this time there is no reliable way to verify a VMEbus reset, this portion of the test is bypassed.)

The I/O ASIC clears the R/W and prefetch bits in the SCSI control when it is reset. This is the indication that the I/O ASIC has been reset. The I/O ASIC reset is a hardware generated pulse.

```
Data retention error in ASIC SCSI control register
  exp = 0x03
  act = 0x00
```

The I/O ASIC reset did not clear by itself.

```
The ASIC was not reset when reset address was strobed
exp = 0x00
act = 0x03
```

The I/O ASIC reset did not occur.

When reset, the SCSI chip will not load a value into any register. This is the indication that it is reset.

```
SBIC chip not reset after reset latch set
    exp = non zero
    act = 0x00
SBIC chip reset after reset latch reset or data error
    exp = 0x00
    act = 0xff
```

Please refer to the I/O ASIC Specification manual, the 33C93 Technical Reference for further information.

5.2.3 Test 6 - I/O ASIC FIFO/ECC Test

This test verifies the I/O ASIC will fetch a cache block from Kbus memory and that single bit errors in any cacheline will be corrected. Every data bit in every cache line is verified to be correctable. The test verifies the following conditions:

- 1. A zero bit corrected to a one in a field of zeroes
- 2. A one bit corrected to a zero in a field of ones
- 3. A one bit corrected to a zero in a field of zeroes

4. A zero bit corrected to a one in a field of ones

The LANCE chip is used to force the I/O ASIC to fetch the test cache block.

The cache block is written to memory with one cache line with bad data and good ECC. The I/O ASIC is forced to fetch the block and the cache block is read again. The data read is verified to be corrected.

I/O ASIC FI check byte	FO/ECC error = 0x0c		
addr	exp	act	xor
0x000c0000	0x0000000000000000000000	0x00000080000000	0x00000008000000

In the above example, the I/O ASIC did not correct the bad bit in the cache line or it did not fetch the block.

5.3 LANCE (7990) Tests

Г

The following tests verify that the AMD7990 LANCE chip and the interface between the LANCE and the I/O ASIC are functional.

5.3.1 Test 7 - 7990 LANCE Initialization Test

This test verifies that the AMD 7990 LANCE chip can initialize through the I/O ASIC from Kbus memory. To verify the initialization, the LANCE status register is verified to contain the expected status.

Ignored bits are masked off.

```
LANCE initialization error
exp status = 0x0020
act status = 0x0030
mode value = 0x0002
```

In the above example, the LANCE chip received the wrong mode value.

Refer to the AMD 7990 technical manual for further information.

5.3.2 Test 8 - 7990 LANCE Internal Loopback Test

This test verifies proper operation of the LANCE with the I/O ASIC, its' address registers and interrupts to the processor. The test initializes the LANCE for internal loopback mode and sets

up a transmit packet of 32 bytes, the maximum for the LANCE, and transmits it to itself. The received packet is then verified.

The expected vector for the LANCE chip is 0x87. Interrupts are checked for valid on:

- 1. Initialization
- 2. Transmit packet
- 3. Receive packet

```
Unexpected interrupt active after LANCE initialization
   No interrupts expected
   vector received = 0x80
```

In the above example, a device that is not being tested generated an interrupt when the LANCE did something.

In the above example, a bad bit was found in the I/O ASICs' Ethernet cacheable block buffer.

5.3.3 Test 9 - 7990 LANCE External Loopback Test (must be prompted)

This test verifies the operation of the LANCE chip with the data encoding component's on the System Board.

An external loopback connector must be installed before this test is run. Figure 3-3 shows the pin layout for the transceiver connector.



Figure 5-1. Pin Layout for Transceiver Connector on EXOS 202

Time out on LANCE transmit LANCE status = 0x0033

In the above example, the external loopback connector is not connected or an external component is bad.

5.3.4 Test 10 - 7990 LANCE Data Alignment Test

This test verifies that the I/O ASIC/LANCE interface will transfer different byte counts correctly. This verifies the SE bus data multiplexer in the ASIC for proper operation.

In the above example, the I/O ASIC got a single byte transfer from the wrong side of the SEbus.

Please refer to the I/O ASIC Specification Manual and the AMD Local Area Network Controller Am7990 (Lance) Technical Manual for further information.

5.3.5 Test 11 - 7990 LANCE Cacheable Data Merge Test

This test verifies the I/O ASIC function of merging new data with existing data when the new data does not start on a 32 byte (cache block) boundary.

The test begins by initializing the test cache block with a known pattern. The test does an internal loopback of 8 bytes (all zeroes) into the prepared cache block at a known offset. Since the LANCE appends 4 byte of ECC to the data, these bytes are expected. The cache block is verified to contain the 8 bytes of zeroes and that none of the other bytes were modified. This procedure is repeated for every 16 bit word position in a cache block (14).

addr	exp	act	xor
0xff0f0180	0xfffe000000000000	0x00000000000000 000	0xfffe000000000000
0xff0f0188	0x0000f5f4f3f2f1f0	0x00000000000000000	0x0000f5f4f3f2f1f0
0xff0f0190	Oxefeeedecebeae9e8	0x00000000000000000	0xefeeedecebeae9e8
0xff0f0198	0xe7e6e5e4e3e2e1e0	0x00000000000000000	0xe7e6e5e4e3e2e1e0

In the above example, the I/O ASIC did not merge the data in RAM, but rather used what was already in the cacheable buffer.

Refer to the Kbus Specification Manual, I/O ASIC Specification Manual, and AMD Local Area Network Controller Am7990 (LANCE) Technical Manual for further information.

5.3.6 Test 12 - 7990 LANCE Address/Cache Data Test (affected by prompt)

This test verifies the operation and selection of the I/O ASIC/LANCE interface transmit registers (TX1 and TX2) and verifies the data integrity of the LANCE cache block buffer. The test verifies that:

- 1. TX address registers are unique and do not interact
- 2. The value in the TX address registers is presented correctly
- 3. The LANCE cache block buffer has no data failures

The LANCE is set up with 2 receive descriptors/buffers and 2 transmit descriptors/buffers, one uses TX1, the other TX2.

The test starts with TX2 pointing to a fixed transmit buffer and TX1 to a moving buffer. Psuedo random data is set into the buffer for TX1 and the TX2 data buffer is set to zero. An internal loopback is done using TX1. The data is verified, and the receive buffer is returned to the LANCE. A loopback is then done with TX2 using random data and TX1 zero, the data checked and buffer returned. This procedure is repeated using TX2 pointing to the moving buffer and TX1 the fixed.

The moving buffers address is updated, checked for valid and the procedure is repeated until all test addresses are complete.

If this test is prompted, the address increments in the valid range, else the address is taken from a pattern table.

Data error	in LANCE cache block		
addr	exp	act	xor
0x00fc00c0	0x43856bc2d88ef179	0x438d6bc2d88ef179	0x0008000000000000

In the above example, the I/O ASIC has a bad bit in the LANCE cache buffer. The I/O ASIC used wrong TX buffer on transmit. The I/O ASIC transmitted a packet using the wrong TX address register.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the AMD Local Area Network Controller Am7990 (Lance) Technical Manual for further information.

5.3.7 Test 13 - 7990 LANCE Cache Block Buswatcher Test

This test verifies the operation of the LANCE interface cacheable buffer buswatcher.

The test begins by turning the CPU's ECC correction off. This allows the test to determine where the data is coming from. A cache block is written to the test memory address with one cache line with bad data and good ECC. The device is forced to fetch the block and the cache block is read again.

If the I/O ASIC provides the data, it will be corrected. If memory supplies it, it will not be corrected. The corrected data is all 00's.

The test is repeated for each test address.

I/O ASIC/LA	NCE buswatcher error		
addr	exp	act	xor
0x00ff9000	0x0000000000000000	0x0800000000000000	0x0800000000000000

In the above example, the I/O ASIC did not supply the data when it was read.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the AMD Local Area Network Controller Am7990 (Lance) Technical Manual for further information.

5.4 SCSI (33C93) Tests

The following tests verify that the Western Digital 33C93 chips and the interface between the SBIC and I/O ASIC are functional.

5.4.1 Test 14 - 33C93 SBIC (SCSI) Data Path (Write Buffer) Test

This test verifies the operation of the WD 33C93 SBIC chip (SCSI) with the I/O ASIC.

The chip is set with a SCSI id of 7 (highest priority) and interrupts are verified to occur and be the correct vector.

The test sequence is as follows:

- The test waits for a minimum of 5 seconds to allow the device to recover from the SCSI bus reset.
- The test performs a SCSI TEST UNIT READY command. This first command will fail with a check condition status and a sense key of 0x06. This error is expected and accounted for, it is the drive saying it was reset.
- The data to transfer to the device is set up, an incrementing pattern of 512 bytes is used.
- The data is transferred to the device using the SCSI WRITE BUFFER command.
- The data is transferred from the device using the SCSI READ BUFFER command.
- The data read is verified to be the same data that was sent.

An example of a data failure follows:

SCSI transf	er error		
addr	exp	act	xor
0x000c0000	0 x0001020304050607	0x0001020384050607	0x000000008 0000000

The "Timeout on access to target 0 after reset" error specifes that SCSI device address 0 and Logical Unit Number (LUN) 0 could not be selected. Verify the device:

- 1. Cables are not damaged and connected
- 2. Is addressed as device 0
- 3. Is addressed as LUN 0

SCSI errors return the command that was issued. The commands used are:

command	device	description
0x00	0x0 0	SCSI command, TEST UNIT READY
0x3b	0x0 0	SCSI command, WRITE BUFFER
0x3c	0x00	SCSI command, READ BUFFER

Errors returned from the device will display:

- 1. SCSI status
- 2. Target status
- 3. Command phase
- 4. SBIC chips status (auxilliary status)

For example:

SCSItargetcommandauxilliarystatusstatusphasestatusexp = 0x020x020x020x02act = 0x020x020x020x02

Please refer to the 33C93 Technical Reference and the Small Computer Systems Interface (SCSI) specification for further information.

5.4.2 Test 15 - SCSI Cacheable Block Data Merge Test

This test verifies the I/O ASIC function of merging new data with existing data when the new data does not start on a 32 byte (cache block) boundary.

The test begins by initializing the test cache block with a known pattern. The test reads one zero byte from the device buffer into the prepared cache block at a known offset. The cache block is verified to contain the zero byte and that none of the other bytes were modified. This procedure is repeated for every byte position in a cache block (31).

addr	exp	act	xor
Oxff 0f0000	0xff00fdfcfbfaf9f8	0x 00000000000000000	0xff00fdfcfbfaf9f8
Oxff0f0008	0xf7f6f5f4f3f2f1f0	0x 00000000000000000	0xf7f6f5f4f3f2f1f0
Oxff0f0010	0xefeeedecebeae9e8	0x 00000000000000000	0xefeeedecebeae9e8
Oxff0f0018	0xe7e6e5e4e3e2e1e0	0x00000000000000000	0xe7e6e5e4e3e2e1e0

In the above example, the I/O ASIC did not merge the data in RAM, but rather used what was already in the cacheable buffer.

Please refer to the Kbus Specification Manual, I/O ASIC Specification Manual, 33C93 Technical Reference, and the Small Computer Systems Interface (SCSI) Specification for further information.

5.4.3 Test 16 - SCSI Multi-bit ECC Error Test

This test verifies the operation of the I/O ASIC when a multi-bit ECC error in ram is detected. The five areas verified are:

- 1. The I/O ASIC will detect a multi-bit ECC error
- 2. The I/O ASIC will set the scsi_err bit in the interrupt status register
- 3. The I/O ASIC will continue to fetch the block and set the scsi_err bit when a zero is written to the scsi_err bit
- 4. The I/O ASIC will reset the scsi_err bit when a new address is written
- 5. The I/O ASIC will reset the scsi_err bit when the multi-bit error condition in the cache block referenced has been cleared

The test begins by setting up a cache block with a multi-bit ECC error. A write to the SCSI device is attempted using the address of the error block. Operations 1 - 4 above are verified. The test is repeated to verify operation 5.

SCSI ECC error was not reset by re-writing address

In the above example, the I/O ASIC did not clear the error when the address register was set to a good cache block in memory.

SCSI ECC error was not reset by writing good data

Solbourne Confidential Information - Do Not Distribute

In the above example, the I/O ASIC did not clear the error when good data was written to the test cache block.

Refer to the Kbus Specification Manual, the I/O ASIC Specification manual, the 33C93 Technical Reference and the the Small Computer Systems Interface (SCSI) specification for further information.

5.4.4 Test 17 - SCSI Page Overflow Bit Test

This test verifies the I/O ASIC function halting a SCSI transfer when the transfer crosses a system page address boundary (every 8192 bytes).

The test verifies that during a SCSI read operation the I/O ASIC will:

- Stop at the page boundary
- Hold off the SCSI chip transfer, the transfer did not complete
- The address register contains the correct value
- The page_overflow bit in the status register is set
- A page overflow interrupt is generated
- Data in the next system page is not modified

The test verifies that during a SCSI write operation the I/O ASIC will:

- Stop at the page boundary
- Hold off the SCSI chip transfer, the transfer did not complete
- The address register contains the correct value
- The page_overflow bit in the status register is set
- A page overflow interrupt is generated

The read test writes 8K of zeroes to the devices buffer and sets a byte of 0xff to the first byte of the next page. The test then reads 8K bytes and sets the address to system page boundary + 4K. The read is then started and a timeout is set. When the I/O ASIC encounters a page boundary, all conditions listed above are verified and the read is allowed to complete.

The write test sets the address to system page boundary + 4K. The write is then started and a timeout is set. When the I/O ASIC encounters a page boundary, all conditions listed above are verified and the write is allowed to complete.

SCSI read completed with page overflow active

In the above example, the I/O ASIC flagged the overflow but did not stop the write to memory.

SCSI read completed without page overflow active

In the above example, the I/O ASIC did not detect the page overflow condition.

```
Timeout on page overflow interrupt
```

In the above example, the I/O ASIC did not interrupt the processor on a page overflow condition.

```
Unexpected vector after page overflow on read
    exp = 0x83
    act = 0x87
```

In the above example, the I/O ASIC is interrupting on the wrong line or there is a failure on the System Board.

```
Page overflow did not prevent write to next page
    exp = 0xff
    act = 0x00
```

In the above example, the I/O ASIC did not prevent a write to the next address on a page overflow.

```
Address miscompare after a page overflow on read
exp = 0x000f1fff
act = 0x000f2000
```

In the above example, the SCSI address incrementor in the I/O ASIC is bad.

Page overflow bit active after clear on read

In the above example, the I/O ASIC did not clear the overflow bit after the write completed.

```
Write to memory did not complete after a page overflow0);
    exp = 0x00
    act = 0xff
```

In the above example, the I/O ASIC did not continue the transfer when a new address was written.

```
SCSI write completed with page overflow active0);
```

In the above example, the I/O ASIC flagged the page overflow but did not stop on the condition.

SCSI write completed without page overflow active

In the above example, the I/O ASIC did not stop on a page overflow condition.

```
Time out on SCSI write after a page overflow page overflow bit inactive
```

In the above example, the I/O ASIC did not continue the transfer when a new address was written, if the page overflow bit is "active," the I/O ASIC did not clear the bit.

Page overflow bit active after clear on write

In the above example, the I/O ASIC did not clear the overflow bit after the write completed.

Please refer to the Kbus Specification Manual, I/O ASIC Specification Manual, and WD 33C93 SBIC Technical Manual for further information.

5.4.5 Test 18 - SCSI Address Counter/Cache Data Test

This test verifies the operation of the SCSI address counter and checks the SCSI cache block buffer for data errors. The test is a write/read loop to/from the SCSI devices buffer.

The test writes and reads 8192 bytes of data to the SCSI devices buffer on each pass. The data used is random and the data is checked after every read. Data errors may be from address or cache buffer faults.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, and the WD 33C93 SBIC Technical Manual for further information.

5.4.6 Test 19 - SCSI Cache Block Buswatcher Test

This test verifies the operation of the SCSI interface cacheable buffer buswatcher.

The test begins by turning the CPU's ECC correction off. This allows the test to determine where the data is coming from. A cache block is written to the test memory address with one cache line with bad data and good ECC. The device is forced to fetch the block and the cache block is read

again.

If the I/O ASIC provides the data, the data will be corrected. If memory supplies it, it will not be corrected. The corrected data is all 00's.

The test is repeated for each test address.

In the above example, the I/O ASIC did not supply the data when it was read.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, and the WD 33C93 SBIC Technical Manual for further information.

5.4.7 Test 20 - SCSI Cache Block Flush Test

This test verifies the operation of the SCSI interface cacheable buffer flush when the flush address is strobed.

The test begins by writing a known pattern to the SCSI device. The data is then read back with the last byte being written to the first byte of the cache block buffer. The SCSI address register is set to the next to last byte of the cache block. The SCSI flush address is strobed and the test data is verified to be correct.

The test is repeated for each byte offset in the cache block (1-31).

SCSI cache bytes int	flush error o cache block = 1		
addr	exp	act	xor
0x00ffe000	0xc300000000000000	0x00000000000000000000	0xc300000000000000

In the above example, the I/O ASIC did not flush the data when the flush address was strobed.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, and the WD 33C93 SBIC Technical Manual for further information.

5.5 Test 21 - SCSI/LANCE Data Transfer Test

This test verifies the operation of the I/O ASIC when the SCSI and LANCE devices are running at the same time.

The test is a loop. Each device is serviced as it has finished a write, read or transmit. The test writes and reads up to 8192 bytes of data for the SCSI device and an internal loop of 32 bytes on the LANCE device. Data is checked after 10 iterations of the SCSI device to keep the SE bus collisions high.

If the test is prompted, the number of SCSI write/reads before a data check is wanted and the number of data check iterations wanted is asked for. Entering "q" or "Q" to any of these prompts exits the test. The test defaults to 10 for each loop.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the WD 33C93 SBIC Technical Manual, and the AMD Local Area Network Controller Am7990 (Lance) Technical Manual for further information.

5.6 VMEbus Interface Tests

The following tests verify that the VMEbus interface on the System Board is functional. This includes tests for the I/O ASIC VMEbus port, the VMEbus address map, and the ability to run VMEbus cycles. Some of the tests in this subsection require VMEbus boards to be installed. Refer to the test descriptions and the vmeconfig command to find the command to be used to enable them.

5.6.1 Test 22 - VMEbus Address Map RAM Test

This test verifies that the VMEbus address map RAM on the System Board is accessible via RIO transactions. It does not verify that Kbus/VMEbus address translations can be performed.

The VMEbus map RAM is a 2048 word memory and is accessible at low address 83000000 and high address 83ffe000. The adjacent locations in the RAM are at 2000 hexadecimal increments.

Currently, each RAM location is 32 bits. However, only four buts can be read. This test only checks four bits from each location in the RAM.

Strategy: Write random byte sequence to 2048 VMEbus map RAM locations. Read and verify contents of VMEbus map RAM.

Possible error messages follow:

```
VMEbus address map write/read error
RIO address = 0xYYYYYYY
exp = EEEEEEEE
act = AAAAAAAA
xor = XXXXXXXX
where:
YYYYYYYY - RIO address where error occurred
EEEEEEEE - value expected to be read from address
AAAAAAAA - actual value read from address
XXXXXXXX - xor of expected and actual values
```

If this test fails, the control logic for accessing the RAM should be checked to be functional. If the control logic is operational, then insure the RAM chips are receiving address and data correctly.

5.6.2 Test 23 - VMEbus Data Path Test

This test verifies correct data path access to the VMEbus using RIO cycles. This test requires a VMEbus Memory Board installed in the VMEbus backplane and the boards' presence listed in the VMEbus configuration table (see the command 'vmeconfig' for more information).

The test program writes and reads the VMEbus RAM board as a 16 bit device and an 8 bit device. The 8 bit device test verifies that the correct VMEbus data strobe line is used and that the unused data strobe does not. The test is run for each data strobe line.

```
VMEbus data error, 16 bit accesses
address = 0x86100000
act = 0xfffe
exp = 0xffff
xor = 0x0001
VMEbus data error, 8 bit accesses
address = 0x86100000
act = 0xff02
exp = 0xff00
xor = 0x0002
```

5.6.3 Test 24 - VMEbus Address Path Test

This test verifies that address lines on the VMEbus backplane do not interact with each other. This test requires a VMEbus Memory Board installed in the VMEbus backplane and the boards' presence listed in the VMEbus configuration table (see the **vmeconfig** (1) command for more information).

The test writes and reads the VMEbus RAM board as a 16 bit device. The test program sets the entire VMEbus RAM to 0x0000, then each single bit address line is set to 0xffff. For example:

address	contents
0x000000	0x0000
0x000002	0xffff
0x000004	Oxffff
0x000006	0x0000
0x000008	0xffff
0x00000a	0x000x0
0x00000c	0x0000
0x00000e	0x000x0
0x000010	0xffff
0x000012	0x0000
•	•
•	•
0x0fffff	0x0000

In this way any address line that interacts with another one will write the opposite data to the location.

The test is repeated using a single low bit in the address.

```
VMEbus addressing error
address = 0x86100000
act = 0xffff
exp = 0x0000
```

5.6.4 Test 25 - VMEbus Data Multiplexing Test

This test verifies the RIO data mutiplexing of the I/O ASIC to the VMEbus. It ensures the data going to any VMEbus device or ram will be presented in the correct position and without error. This test requires a VMEbus ram board to be installed and configured (see command 'vmeconf' for further information) before this test will run.

The first part of this test is a simple data path check, verifying the 32 bit data path. This check is made to both positions of the 64 bit Kbus data path. After the 32 bit path is known to work, the 16 and 8 bit transfers are done using 32 bit transfers to verify proper transfer and placement of the 16 or 8 bit data in all positions of the 64 bit data path.

```
VMEbus data path or data muxing error
  exp = 0xfffffff
  act = 0x00000000
  xor = 0xfffffff
  position 32 - 63
```

In the above example, the I/O ASIC did not multiplex the data properly on a 32 bit access on the high order 32 bits, 32 - 63, of the bus.

```
VMEbus 16 bit data muxing error
exp = 0x0000 0xffff 0x0000 0x0000
act = 0x0000 0x0000 0xffff 0x0000
```

In the above example, the I/O ASIC did not multiplex the data properly on a 16 bit access.

In the above example, the I/O ASIC did not multiplex the data properly on a 8 bit access.

```
VMEbus 8 or 32 bit data muxing error
exp = 0x00000000 0x00ff0000
act = 0x00ff0000 0x0000000
```

In the above example, the I/O ASIC did not multiplex the data properly on an eight-bit access.

Refer to the Kbus Specification Manual and the VMEbus Specification, Revision C.1 for further information.

5.6.5 Test 26 - VMEbus Interrupt (IACK) Test

This test verifies than when a VMEbus device interrupts, the system will see the correct interrupt level and the correct interrupt response.

This test causes the Ciprico Rimfire 3500 VMEbus-to-SCSI card to generate an interrupt for each combination of interrupt priority level and VMEbus vector.

☆ ☆ ☆ NOTE ☆ ☆ ☆

If the VMEbus configuration table has not been configured with a SCSI board, the test will not be executed and a message printed warning the user that the test was skipped. This message(s) appears as follows:

No SCSI board found in configuration table Test skipped.

If incorrect memory or VMEbus configuration tables are used while running this test, invalid results are produced.

Example of possible error messages follow:

```
    RF3500 C_IDENTIFY command returned error condition
    Correct vector returned on VMEbus IACK cycle on IRQ level 1
but no interrupt was received
vector = 0x01
expected interrupt = IRQ1
```

This error indicates the correct VMEbus vector was received but no VMEbus interrupt was received from the System Board.

```
3) Correct vector returned on VMEbus IACK cycle on IRQ level 2
but wrong interrupt was received
vector = 0x02
expected interrupt = IRQ2
received interrupt = IRQ1
```

This error indicates the correct VMEbus vector was received but an interrupt associated with the wrong VMEbus IRQ level was transmitted by the System Board.

```
4) Incorrect vector was returned from IACK on IRQ level 2
Received interrupt on IRQ level 2
expected vector = 0x84
received vector = 0x82
```

This error indicates the wrong VMEbus vector was returned on the VMEbus interrupt acknodledge cycle. The System Board transmitted the interrupt for the IRQ level.

```
5) Bus Timeout occurred attempting to IACK on IRQ level 2
Received interrupt on IRQ level 1
expected vector = 0x84
no vector received on any other IRQ level
```

This error indicates that a bus timeout occurred (no VMEbus device responded) on the VMEbus interrupt acknowledge cycle. The System Board transmitted the interrupt for the indicated IRQ level.

6) Bus Timeout occurred attempting to IACK on IRQ level 2 Received interrupt on IRQ level 1 expected vector = 0x84 vector 0x82 was received on IRQ level 1

This error indicates that a bus timeout occurred while performing a VMEbus interrupt acknowledge cycle on IRQ level 2. However, a VMEbus device did acknowledge on IRQ level 1. wrong vector number.

7) VMEbus interrupt status bit was inactive on VMEbus interrupt IRQ level 2

This error indicates the System Board did not show an active VMEbus interrupt in the status register when the interrupt was active.

8) VMEbus interrupt status bit was active after a VMEbus interrupt IRQ level 4

This error indicates that the System Board shows an active VMEbus interrupt in the status register when the interrupt was not active.

5.6.6 Test 27 - RF3500 SCSI Data Path (Write Buffer) Test

This test verifies functionality of the VMEbus emulator on the System Board and the connection of the SCSI device to the Ciprico RIMFIRE 3500 SCSI (Ciprico) board.

This test requires a Ciprico VMEbus SCSI board installed in the VMEbus backplane and the boards' presence listed in the VMEbus configuration table (see the command 'vmeconfig' for more information).

The test sequence follows:

- 1. The Ciprico SCSI board is reset and the test waits for the board to complete self tests.
- 2. The board options are set with a SCSI id of 7.
- 3. The unit options are set with a retry limit of 3.
- 4. The test waits for a minimum of 5 seconds to allow the device to recover from the SCSI bus reset.
- 5. The test performs a SCSI TEST UNIT READY command. This first command will fail with a check condition status and a sense key of 0x06. This error is expected, it is the drive saying it was reset.
- 6. The size of the devices buffer is queried and is used for the transfer count. This count is displayed.
- 7. The data to transfer to the device is set up, an incrementing pattern is used.
- 8. The data is transferred to the device using the SCSI WRITE BUFFER command.
- 9. The data is transferred from the device using the SCSI READ BUFFER command.
- 10. The data read is verified to be the same data that was sent.
- 11. The data transfer, write-read-verify sequence, is repeated when loop is on.

An example of a data failure follows:

```
Data error on SCSI transfer
transfer offset = 350
exp = 0x02
act = 0x03
xor = 0x01
```

Errors in this test can be from:

- The System board will not address the Ciprico SCSI (run VMEbus Data and Address path tests to verify).
- The System board will not allow the Ciprico board to address system RAM (run System Board VMEbus address map RAM test to verify mapping RAM). This is not the only cause for this condition.
- The 'Error on first selection of device' error specifies that SCSI device address 0 and Logical Unit Number (LUN) 0 could not be selected. Verify the device:
 - 1. cables are not damaged and connected
 - 2. is addressed as device 0
 - 3. is addressed as LUN 0

SCSI errors return the command that was issued and the device that was addressed. The commands used are shown in the following table.

Command	Device	Description
0x00	0x00	SCSI command, TEST UNIT READY
0x07	0xff	Ciprico board command, set general options
0x07	0xff	Ciprico board command, set device options
0x3b	0x00	SCSI command, WRITE BUFFER
0x3c	0x00	SCSI command, READ BUFFER

Errors returned from the device will display Ciprico board error status (error code) and various SCSI status bytes. Refer to the Ciprico RIMFIRE 3500 Product specification and the Small Computer Systems Interface specification for further information.

Example of an error on a command follows:

```
Error status returned after command
error code = 0x4b SCSI status = 0x02 SCSI flags = 0x00
info byte 3 = 0x00 4 = 0x00 5 = 0x00 6 = 0x00
Command 0x3c on device 0x00
```

5.6.7 Test 28 - VMEbus Cacheable Data Merging Test

These tests verify the I/O ASIC function of merging new data with existing data when the new data does not start on a 32 byte (cache block) boundary on the VMEbus interface. The test uses the Ciprico RF3500 SCSI interface card and a SCSI device.

The block mode data merging test uses block mode transfers to verify the merging of data into the ping-pong buffers.

The test begins by initializing the test cache block with a known pattern. The test reads one zero byte from the device buffer into the prepared cache block at a known offset. The cache block is verified to contain the zero byte and that none of the other bytes were modified. This procedure is repeated for every byte position in a cache block (31).

In the above example, the I/O ASIC did not merge the data in RAM, but rather used what was already in the cacheable buffer.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.6.8 Test 29 - VMEbus Block Mode Data Merging Test

These tests verify the I/O ASIC function of merging new data with existing data when the new data does not start on a 32 byte (cache block) boundary on the VMEbus interface. The test uses the Ciprico RF3500 SCSI interface card and a SCSI device.

The block mode data merging test uses block mode transfers to verify the merging of data into the ping-pong buffers.

The test begins by initializing the test cache block with a known pattern. The test reads one zero byte from the device buffer into the prepared cache block at a known offset. The cache block is verified to contain the zero byte and that none of the other bytes were modified. This procedure is repeated for every byte position in a cache block (31).

In the above example, the I/O ASIC did not merge the data in RAM, but rather used what was already in the ping-pong buffers.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.6.9 Test 30 - VMEbus Cacheable Data Buffer Test

These tests verify the I/O ASIC's data buffers for data integrity. The tests also verify data transfers of 16 and 32 bits to/from the VMEbus. The test uses the Ciprico RF3500 SCSI interface card and a SCSI device.

The test writes and reads random data to the devices buffer. After the read, the data is verified. Cacheable data transfers 8 Kbytes, block mode data (using the ping-pong buffers) transfers 16 Kbytes. The write/read loop is done 50 times for the 16 bit and 50 times for the 32 bit transfers.

In the above example, the I/O ASIC's cacheable buffer has a bad bit.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.6.10 Test 31 - VMEbus Ping-Pong Data Buffers Test

These tests verify the I/O ASIC's data buffers for data integrity. The tests also verify data transfers of 16 and 32 bits to/from the VMEbus. The test uses the Ciprico RF3500 SCSI interface card and a SCSI device.

The test writes and reads random data to the devices buffer. After the read, the data is verified. Cacheable data transfers 8 Kbytes, block mode data (using the ping-pong buffers) transfers 16

Kbytes. The write/read loop is done 50 times for the 16 bit and 50 times for the 32 bit transfers.

```
VMEbus ping-pong buffer error<br/>data size = 32 bitaddrexpaddrexp0xff1000000xc4a57fefd8290b380xff1000000xc4a57fefd8290b38
```

In the above example, an I/O ASIC's ping-pong buffer is bad.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.6.11 Test 32 - VMEbus Block Mode Address Counter Test

This test verifies the functionality of the I/O ASIC's internal block mode address counter. The test uses the Ciprico RF3500 SCSI interface card and a SCSI device.

The test writes and reads a block of random data to the SCSI device and varies the starting address. This forces the address counter to start at every legal value and create the next block mode address.

The read area of memory is cleared before and the data is verified after each read. The test uses 16 bit and 32 bit transfers.

```
        VMEbus block mode address counter error

        data size = 16 bit

        addr
        exp

        0xff100000
        0xc4a57fefd8290b38

        0xff100008
        0x9af55427815463fb

        0xff100008
        0x9af55427815463fb
```

In the above example, the I/O ASIC's block mode address counter added a count on a 16 bit transfer.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.6.12 Test 33 - VMEbus Panic Interrupt Test

This test verifies the I/O ASIC function of presenting an interrupt to the system when data was taken from the VMEbus device but the address is illegal. The test uses the Ciprico RF3500 SCSI

interface card and a SCSI device.

The test verifies that:

- The panic interrupt bit is set
- A panic interrupt is generated with the correct vector
- The panic interrupt bit can be reset by writing a zero to it

The test begins by initializing the Kbus RAM and two 8 Kbyte partions for the VMEbus map ram. 256 bytes of data is written to the SCSI device. The second VMEbus map ram is set to point to non-existant memory. A read of 256 bytes is started at an address 252 bytes from the nonexistant page. When the read reaches the non-existent page, the I/O ASIC does not have an address for the data and a panic interrupt is generated.

```
VMEbus panic indicator bit not set
```

In the above example, the I/O ASIC did not detect the error condition or the latch is bad.

VMEbus panic interrupt did not occur

In the above example, the I/O ASIC did not set the interrupt.

```
Wrong vector received for VMEbus panic interrupt
  exp = 0x83
  act = 0x85
```

In the above example, the I/O ASIC is connected to the wrong interrupt line.

VMEbus panic indicator bit would not reset

In the above example, the I/O ASIC is bad.

Please refer to the Kbus Specification Manual, the I/O ASIC Specification Manual, the VMEbus Specification, Revision C.1, and the Small Computer Systems Interface (SCSI) specification for further information.

5.7 BW20 Tests

The following tests verify that the monochrome frame buffer on the System Board is functional. Note that these tests do not verify that the video output signal is functional. This must be done visually.

5.7.1 Test 34 - Frame Buffer RAM Test (affected by prompt)

This test verifies the integrity of the frame buffer RAMs using the MARCH test algorithm (see Appendix A).

The test begins by printing:

```
Testing BW20 Board at Slot 7
Testing Frame Buffer:
Address range 0 to 1fff...
```

If the first frame buffer access results in a data fault, the test prints:

```
Frame Buffer does not respond to address 0xa7000000
verify that a valid frame buffer configuration exists
Test skipped.
```

The user should use the **fbconfig**(1) command to verify the monochrome frame buffer is configured correctly. If it is configured correctly, then this message should be considered an error. If not configured correctly, use the **fbconfig** command to correct the problem and re-run the test.

If in the frame buffer configuration table (see fbconfig) specifies that the monochrome board under test is connected to a low-resolution monitor, then the frame buffer addresses 0 to 0x1fff are tested. If high resolution is specified, then frame buffer addresses 0 through 0x3fff are tested.

If the test is prompted, the test is performed using all the RIO access types supported by the frame buffer under test. In order words, if the frame buffer supports quad-byte RIO accesses, then the test is executed in entirely using quad-byte RIO accesses, then repeated using double-byte RIO accesses, then repeated again using byte RIO accesses. If the test is not prompted, only quad-byte accesses are used.

For each access type (identified above) the test is executed five times using different complementary pattern pairs: The first pass uses the 00/ff complement pair in each byte of the data. The second pass uses the 0f/f0 complement pair in each byte of the data. The third pass uses the cc/33 complement pair in each byte of the data. The fourth pass uses the 99/66 complement pair. The fifth pass uses the 55/aa complement pair in each byte of the data.

If an incorrect frame buffer configuration table is used while running this test, invalid results will be produced.

Possible error messages follow:

This error may reflect RIO addressing problems or problems with the frame buffer RAM components/support logic.

This error may reflect RIO addressing problems or problems with the frame buffer RAM components/support logic.

5.7.2 Test 35 - Frame Buffer Interrupt and Interrupt Registers Test

This test verifies the correct operation of the selected monochrome frame buffer board to generate interrupts.

The test enables interrupts on the monochrome frame buffer board and makes sure that they are received by the CPU.

☆ ☆ ☆ NOTE ☆ ☆ ☆

If the frame buffer configuration table has not been configured, the test will not be executed and a message printed warning the user that the test was skipped. This message(s) will appear as follows:

Frame buffer configuration table has not been initialized Test skipped.

If an incorrect frame buffer configuration table is used while running this test, invalid results are produced.

Possible error messages follow:

```
Timeout occurred while waiting to receive interrupt
from frame buffer board in slot 7
   expected vector = 0x85
DG>
```

or:

```
Incorrect interrupt received from frame buffer board in slot 7
  expected vector = 0x85
  received vector = 0x84
DG>
```

These errors indicates the interrupt register cannot be accessed on the slow bus or the register is not functioning properly.

5.8 Serial Port Tests

The following tests verify the functions associated with the serial ports (port A, port B, keyboard, and mouse) on the System Board.

5.8.1 Test 36 - Serial Ports Reset Test

This test verifies the reset state of the Z8530 Serial Communication Controller (SCC) chip for serial ports A and B as well as the mouse and keyboard ports. This test also verifies that all ports are accessible.

ት ት ት NOTE ት ት ት

This test uses the environment variable "CONSOLE" in order to determine which ports to test. In order to test the keyboard and mouse ports, the CONSOLE variable must be set to be one of the serial ports. For example, setting "CONSOLE=zs(,1,)" allows testing both the keyboard and mouse. This variable is set while at the ROM prompt.

Possible error messages follow:

```
Serial Port A reset test failure
failing register = RR0
exp = 44
act = c0
xor = 84
Serial Port B reset test failure
failing register = RR0
exp = 44
act = c0
xor = 84
```

The Serial Port A reset test failure indicates that either the port A can not be accessed or reset. The user should make sure that command data is being written to the Z8530 chip and that status information is actually being read.

The Serial Port B reset test failure indicates that either the port B can not be accessed or reset. The user should make sure that command data is being written to the Z8530 chip and that status information is actually being read.

5.8.2 Test 37 - Serial Ports Internal Loopback Test

This test performs a loopback test for each channel in the Z8530 SCC chip for serial ports A and B as well as the keyboard and mouse ports.

☆ ☆ ☆ NOTE ☆ ☆ ☆

This test uses the environment variable "CONSOLE" in order to determine which ports to test. In order to test the keyboard and mouse ports, the CONSOLE variable must be set to be one of the serial ports. For example, setting "CONSOLE=zs(,1,)" allows testing both the keyboard and mouse. This variable is set while at the ROM prompt.

Possible error messages follow:

```
Serial Port A loopback test failure

TX character = ff

RX character = 00

xor = ff

Serial Port B loopback test failure

TX character = 01

RX character = 00

xor = 01
```

When Serial Port A internal loopback test failure occurs, the user must verify that the clock signal is supplied by the mouse port to the chip. A failure may indicates the SCC chip is malfunctioning.

When Serial Port B internal loopback test failure occurs, the user must verify that the clock signal is supplied by the mouse port to the chip. A failure may indicates the SCC chip is malfunctioning.

5.8.3 Test 38 - System Timer Test

The System Timer Test verifies that the System Board can generate system timer interrupts.

To operate the system timer, the z8530 SCC chip for the keyboard must be re-programmed such that the baud rate generator is enabled and the TRxC pin is configured to output the baud rate generator signal. Because re-programming the keyboard port conflicts with normal user input, this test will only execute when the CONSOLE ROM environment variable is set to zs() (serial port A or B). If the console variable is set to something other than zs(), the test will print the following message:

```
To run test 14 CONSOLE must be set to zs()
Test skipped.
```

If the console device is one of the serial ports, then this test correctly programs the keyboard port and enables the System Board interrupt register. If a Kbus timeout error occurs while the keyboard port is being setup, one of the following messages will be printed and the test halts:

```
Timeout error occurred while resetting the Z8530
Port = KBD
```

Timeout error occurred while programming KBD port

The test then waits for a system timer interrupt to occur. If no system timer interrupt occurs within a few milliseconds, the test will print the following error message and halt:

```
Timeout waiting for system timer interrupt
interrupt count = 0
```

If an interrupt occurs, the test verifies that the correct vector (0x8d) is received. If the wrong vector is received, the test will print the following message and halt:

```
Incorrect System Board interrrupt vector received
   number of interrupts received = 1
   expected vector = 0x8d
   received vector = 0x8c
```

The test continues to receive system timer interrupts until 1000 interrupts have been received.

5.8.4 Test 39 - Profile Timer Test

The Profile Timer Test verifies that the System Board can generate profile timer interrupts.

To operate the profile timer, the RTS pin of the z8530 SCC chip must be set to true. Setting the RTS pin does not conflict with the normal mouse port programming. Therefore this test executes no matter what the ROM environment CONSOLE variable is set to.

The test writes the mouse port to set the RTS pin. If a Kbus timeout occurs while the mouse port is being accessed the following message will be printed and the test will halt:

Timeout occurred while setting RTS bit in WR5 of MOUSE port

The test then enables System Board interrupts and waits for a profile timer interrupt to occur. If no profile timer interrupt occurs within a few milliseconds, the test will print the following error message and halt:

```
Timeout waiting for profile timer interrupt
interrupt count = 0
```

If an interrupt occurs, the test verifies that the correct vector (0x8f) is received. If the wrong vector is received, the test will print the following message and halt:

```
Incorrect System Board interrrupt vector received
number of interrupts received = 1
expected vector = 0x8f
received vector = 0x8d
```

The test continues to receive profile timer interrupts until 1000 interrupts have been received.

5.9 Test 40 - RTC-58321 Real Time Clock Test

This is a test of the RTC-58321 chip on the System Board. This test verifies that the real time clock (RTC) internal registers can be accessed and that the clock is counting. The RTC is connected to the slow bus is located at RIO address 17020000.

This test consists of two parts:

- 1. Tests the RTC registers by writing test patterns and then reading them back for verification.
- 2. Tests verifies that writing the RTC chip can count by loading the registers such that a rollover causes a ripple from the least significant digit to the most significant digit.

Possible error messages follow:

```
RTC register write/read failure
RTC register address = RR
write pattern = P
exp = E
act = A
xor = X
where:
    RR - RTC register number
    P - pattern written to RTC register
    E - value expected to be read from RTC register
    A - actual value read from RTC register
    X - xor of expected and actual values
```

This error message implies that there may be a problem accessing the RTC chip from the slow bus.

```
Clock state rollover test failure (24 hr mode)

S MI HR W D MO YR

Initial state = 59 59 b2 6 31 12 99

Expected state = 04 00 80 0 01 01 00

Actual state = AA AA AA AA AA AA

where:

AA - actual state of RTC register
```

If this error occurs, an RTC chip failure has occurred.

6.1 Introduction

The graphics tests available under dg are given in this section.

6.2 CG40 Tests

The following sections contain descriptions of the tests for the CG40 Board.

6.2.1 Bt458 Tests

The following tests verify that the Brooktree Bt458 RAMDAC chip is accessable and functional. Note that only the S/W visible functions of the RAMDAC are tested. The RGB output signals must be verified visually.

6.2.1.1 Test 41 - Bt458 Registers Test

This test verifies that the Bt458 registers on the CG40 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared.

Registers tested are:

- 1. Read mask register
- 2. Blink mask register
- 3. Command register
- 4. Test register

Refer to Appendix C if frame buffer configuration table contains multiple CG40 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXX
Bt458 REG TESTED RIO write access error
Slot = SLOT
RIO address = XXXXXXX
Data error in Bt458 REG TESTED
Slot = SLOT
RIO address = XXXXXXX
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG40 Board under test
REG TESTED	RIO address of Bt458 register under test
xxxxxx	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values

6.2.1.2 Test 42 - Bt458 Address Register (Autoincrement) Test

The Bt458 Address Register has the ability to autoincrement to the next location as sequential read or writes are performed to the main and overlay colormaps. This test verifies that this register does increment by first initializing it to point to the first entry in the main colormap, performing three successive reads and then checking that the address register has incremented. This check is done for all 256 entries of the main colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG40 boards and the test is prompted.

Solbourne Confidential Information – Do Not Distribute

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Bt458 Colormap Palette Data Register RIO read access error
Slot = SLOT
RIO address = XXXXXXXX
Autoincrement error in Bt458 Colormap Address Register
Slot = SLOT
RIO address = XXXXXXXX
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG40 Board under test
XXXXXXX	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values

6.2.1.3 Test 43 - Bt458 Main CLUT (Colormap) Test

The Bt458 chip has a 254 * 24 color lookup table that is used by the CG40 board to select color information. This test verifies the integrity of this RAM by performing a simple write, read, compare test on all 256 entries of the main colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG40 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXX
Bt458 Colormap Palette Data Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Data error in Bt458 Main CLUT
Slot = SLOT
RIO address = XXXXXXX
CLUT index = II
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
xxxxxx	RIO address of Bt458 register under test
II	Index into the CLUT
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values

6.2.1.4 Test 44 - Bt458 Overlay CLUT (Colormap) Test

The Bt458 chip contains an internal 4 * 24 color lookup table that is normally used to select the colors of the overlay planes. This test verifies the integrity of this RAM by performing a simple write, read, compare test on all entries of the overlay colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG40 boards and the test is prompted.

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXX
Bt458 Colormap Overlay Data Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Data error in Bt458 Overlay CLUT
Slot = SLOT
RIO address = XXXXXXXX
CLUT index = II
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
xxxxxx	RIO address of Bt458 register under test
II	Index into the CLUT
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values

6.2.2 Test 45 - Frame Buffer RAM Test (affected by prompt)

This test verifies the integrity of the color frame buffer RAMs using the MARCH test algorithm (see Appendix A).

The test first prints:

Testing CG40 Board at Slot X

The test may be executed in two modes depending on the state of the prompt(1) flag for this test. If the prompt flag is set (true), the test presents a menu with choices of plane(s) to test. The user is ask to select the plane(s) that are to be tested. Following is an example of how this works:

```
Plane to test:
    1) First Overlay Plane
    2) Second Overlay Plane
    3) Image Plane
    4) All of the above
Which Plane?
```

When the prompt flag is not set (false), all planes (both overlay planes and the image plane) are tested.

The first overlay plane resides in addresses 0 to 0x1fff, the second overlay plane resides in addresses 0x20000 through 0x3fff, and the image plane in addresses 0x100000 through 0x3ffff.

If prompted, the test is performed using all the RIO access types supported by the frame buffer under test. In order words, if the frame buffer supports quad-byte RIO accesses, then the test is executed in entirely using quad-byte RIO accesses, then repeated using double-byte RIO accesses, then repeated again using byte RIO accesses. If not prompted only quad-byte accesses are used.

For each access type (identified above) the test is executed five times using different complementary pattern pairs: The first pass uses the 00/ff complement pair in each byte of the data. The second pass uses the 0f/f0 complement pair in each byte of the data. The third pass uses the cc/33 complement pair in each byte of the data. The fourth pass uses the 99/66 complement pair. The fifth pass uses the 55/aa complement pair in each byte of the data.

☆ ☆ ☆ NOTE ☆ ☆ ☆

If the frame buffer configuration table has not been configured, the test will not be executed and a message printed warning the user that the test was skipped. This message(s) will appear as follows:

Frame buffer configuration table has not been initialized Test skipped.

If an incorrect frame buffer configuration table is used while running this test, invalid results will be produced.

Possible error messages follow:
```
Frame buffer RAM data error (on read of background pattern)
Slot = SS RAM address = AAAAAAAA
exp = XXXXXXX
act = YYYYYYYY
where:
            SS - slot number of board
AAAAAAAA - frame buffer address where error occurred
XXXXXXXX - value expected to be read from address
YYYYYYYY - actual value read from address
```

This error may reflect RIO addressing problems or problems with the frame buffer RAM components/support logic.

```
Frame buffer RAM data error (on read of complement pattern)
Slot = SS RAM address = AAAAAAA
exp = XXXXXXX
act = YYYYYYY
where:
            SS - slot number of board
AAAAAAA - frame buffer address where error occurred
XXXXXXXX - value expected to be read from address
YYYYYYYY - actual value read from address
```

This error may reflect RIO addressing problems or problems with the frame buffer RAM components/support logic.

6.2.3 Test 46 - Frame Buffer Interrupt and Interrupt Registers Test

This test verifies the correct operation of the color frame buffer boards to generate interrupts. It contains two tests for the frame buffer board interrupt logic.

The first test case is a register write/read test of the frame buffer board interrupt registers.

The second part of this test, enables interrupts in the frame buffer board and makes sure that they are received.

☆ ☆ ☆ NOTE ☆ ☆ ☆

If the frame buffer configuration table has not been configured, the test will not be executed and a message printed warning the user that the test was skipped. This message(s) will appear as follows:

Frame buffer configuration table has not been initialized Test skipped.

If an incorrect frame buffer configuration table is used while running this test, invalid results are produced.

Possible error messages follow:

```
IVR register write/read error for COLOR board in slot 5
  RIO address = 0x15060000
  exp = 0x7f
  act = 0x00
  xor = 0x7f
DG>
```

This error indicates the interrupt register cannot be accessed on the slow bus or the register is not functioning properly.

6.3 CG30 Tests

The following sections contain descriptions of the tests for the CG30 Board.

6.3.1 Registers Tests

The following tests verify that the CG30 Board-level control registers are functional.

6.3.1.1 Test 47 - I/O Location/Control Status Registers Test

This test verifies that the I/O Location and the Control/Status registers on the CG30 board can be accessed and checks both registers for data retention. Each register is tested for a series of patterns which are written, read back and compared.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
RIO read access error

Slot = SLOT

Register = REG TESTED

RIO address = XXXXXXXX

Data error in REG TESTED

Slot = SLOT

RIO address = XXXXXXXX

exp = EE

act = AA

xor = OO
```

SLOT	Slot number of CG30 board under test
REG TESTED	Register under test
xxxxxx	RIO address under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values

6.3.1.2 Test 48 - I/O Location (Space bits) Register Test

The I/O Location register determines which I/O space the frame buffer and compatibility registers reside in. This test verifies that the space bits are funtional by setting the bits to the space as determined in the frame buffer configuration table and performing a RIO read transaction to the Compatible Space Status register. In this manner the I/O space control logic in this board is verify for minimal operation, since this is needed to operate the CG30 board.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error
Slot = SLOT
Register = Compatible Space Status Register
RIO address = XXXXXXXX
```

Where:

SLOTSlot number of CG30 board under testXXXXXXXRIO address of register

6.3.1.3 Test 49 - 8-bit Registers Test

This test verifies that the 8-bit registers on the CG30 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared.

Registers tested are:

- 1. Colormap Address Register
- 2. HW Cursor 1 Address Low Register
- 3. HW Cursor 1 Address High Register
- 4. HW Cursor 2 Address Low Register
- 5. HW Cursor 2 Address High Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error

Slot = SLOT

Register = REG TESTED

RIO address = XXXXXXXX

Data error in REG TESTED

Slot = SLOT

RIO address = XXXXXXXX

exp = EE

act = AA

xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
REG TESTED	Register under test
xxxxxxx	RIO address of register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.1.4 Test 50 - 16-bit Registers Test

This test verifies that the 16-bit registers on the CG30 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared.

Registers tested are:

- 1. Compatible Space Status Register
- 2. Compatible Space Plane Mask Register
- 3. Retrace Interrupt Register
- 4. Serial Port Interrupt Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error

Slot = SLOT

Register = REG TESTED

RIO address = XXXXXXXX

Data error in REG TESTED

Slot = SLOT

RIO address = XXXXXXXX

exp = EE

act = AA

xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
REG TESTED	Register under test
xxxxxxx	RIO address of register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.1.5 Test 51 - Dummy Registers Access Test

Dummy registers are not used in this implementation of a CG30 compatible frame buffer. Reading or writing to these locations will not cause a bus timeout. This test checks that accesses to these registers do not in fact result in a bus timeout for all types of accesses (byte, double byte and quad byte).

Registers tested are:

- 1. Dblbuf Register
- 2. DMAbase Register
- 3. DMAwidth Register

- 4. CG30 Interrupt Vector Register
- 5. CG30 ID Register
- 6. Auxiliary Status Register
- 7. ROP Mode Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error
Slot = SLOT
Dummy Register = REG TESTED
RIO address = XXXXXXXX
RIO access = RIO TYPE
```

Where:

SLOT	Slot number of CG30 board under test
REG TESTED	Dummy register under test
XXXXXXXX	RIO address of register under test
RIO TYPE	Byte, doublebyte, or quadbyte

6.3.2 Bt458 Tests

The following tests verify that the Brooktree Bt458 RAMDAC chip on the CG30 Board is accessable and functional. Note that the RGB video outputs on the Bt458 can not be verified. This must be done visually.

6.3.2.1 Test 52 - Bt458 Registers Test

This test verifies that the Bt458 registers on the CG30 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared.

Registers tested are:

- 1. Read mask register
- 2. Blink mask register
- 3. Command register
- 4. Test register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information – Do Not Distribute

Possible errors detected by this test:

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Bt458 REG TESTED RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Data error in Bt458 REG TESTED
Slot = SLOT
RIO address = XXXXXXXX
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
REG TESTED	Bt458 register under test
xxxxxxx	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.2.2 Test 53 - Bt458 Address Register (Autoincrement) Test

The Bt458 Address Register has the ability to autoincrement to the next location as sequential read or writes are performed to the main and overlay colormaps. This test verifies that this register does increment by first initializing it to point to the first entry in the main colormap, performing 3 successive reads and then checking that the address register has incremented. This check is done for all 256 entries of the main colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Bt458 Colormap Palette Data Register RIO read access error
Slot = SLOT
RIO address = XXXXXXXX
Autoincrement error in Bt458 Colormap Address Register
Slot = SLOT
RIO address = XXXXXXXX
exp = EE
act = AA
xor = OO
```

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.2.3 Test 54 - Bt458 Main CLUT (Colormap) Test

The Bt458 chip has a 254 * 24 color lookup table that is used by the CG30 board to select color information. This test verifies the integrity of this RAM by performing a simple write, read, compare test on all 256 entries of the main colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information – Do Not Distribute

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Bt458 Colormap Palette Data Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Data error in Bt458 Main CLUT
Slot = SLOT
RIO address = XXXXXXXX
CLUT index = II
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.2.4 Test 55 - Bt458 Overlay CLUT (Colormap) Test

The Bt458 chip contains an internal 4 * 24 color lookup table that is normally used to select the colors of the overlay planes. This test verifies the integrity of this RAM by performing a simple write, read, compare test on all entries of the overlay colormap.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Bt458 Colormap Overlay Data Register RIO write access error
Slot = SLOT
RIO address = XXXXXXXX
Data error in Bt458 Overlay CLUT
Slot = SLOT
RIO address = XXXXXXXX
CLUT index = II
exp = EE
act = AA
xor = OO
```

Where:

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address of Bt458 register under test
II	Index into the CLUT
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.2.5 Test 56 - Colormap Pattern Test (must be prompted)

The Bt458 chip (index 0) is incremented from 0 through 255 (all intensities) for red, green and blue. The frame buffer is enabled with all planes initialized to zero value. The result is a smooth transition of intensities for each color. This test writes variable data to the Bt458 and provides a vehicle for subjective evaluation of the Bt458 color outputs.

Refer to Appendix C if the frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Bt458 Colormap Address Register RIO write access error
Slot = SLOT
RIO address = xxxxxxxx
Bt458 Colormap Palette Data Register RIO write access error
Slot = SLOT
RIO Address = xxxxxxx
```

SLOTSlot number of CG30 board under testXXXXXXXRIO address of Bt458 register under test

6.3.2.6 Test 57 - Frame Buffer to Colormap Test

All locations of the Bt458 are written with data that equals their address. Each location of the frame buffer is written with data that equals the last byte of its address. The frame buffer is enabled which displays a grayscale image on the monitor. The Bt458 is read back and the data received is compared to the frame buffer data - byte by byte. This test checks the serial data latches between the frame buffer and the Bt458. The image gives a subjective method to evaluate the correlation between the frame buffer and the Bt458.

Refer to Appendix C if the frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Frame Buffer Read Access Error - RIO Timeout
Slot = SLOT
RIO Address = xxxxxxx
Frame Buffer Index to Colormap Data Error
Slot = SLOT
Frame Buffer Address = xxxxxxx
exp = EE
act = AA
```

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address of Bt458 register under test
EE	Expected value when read
AA	Actual value read

6.3.2.7 Test 58 - Colorbar Test Pattern (must be prompted)

This is a subjective test that outputs a color bar to the monitor and some registration marks. The intention of the test is to display known colors so that the Bt458 outputs can be intuitively evaluated. The registration marks allow alignment of the image on a monitor.

Refer to Appendix C if the frame buffer configuration table contains multiple CG30 boards and the test is prompted.

There are no errors detected by this test.

6.3.3 Bt431 Tests

The following tests verify that the Brooktree Bt431 cursor generation chip on the CG30 Board is accessable and functional. Note that the cursor output signals can not be verified by software. This must be done visually.

6.3.3.1 Test 59 - Address Register (Autoincrement) Test

The Bt431 Address Register has the ability to autoincrement to the next location as sequential read or writes are performed to the pattern RAM. This test verifies that this register does increment by first initializing it to point to the first entry in the pattern RAM, successive reads and then checking that the address register has incremented. This check is done for all entries of the pattern RAM, on both cursor chips.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Bt431 Address Low Register RIO write access error
  Slot = SLOT
 HW Cursor = C
 RIO address = XXXXXXXX
Bt431 Address High Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 RAM Data Register RIO read access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Autoincrement error in Bt431 Address Register
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
С	Hardware cursor under test
xxxxxxx	RIO address of Bt431 register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.3.2 Test 60 - Single Cursor Control Registers Test

This test verifies that the Bt431 registers on the CG30 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared. The test is performed for both Bt431 chips.

Bt431 Registers tested are:

- 1. Command Register
- 2. Cursor (x) Low Register
- 3. Cursor (x) High Register
- 4. Cursor (y) Low Register

- 5. Cursor (y) High Register
- 6. Window (x) Low Register
- 7. Window (x) High Register
- 8. Window (y) Low Register
- 9. Window (y) High Register
- 10. Window Width Low Register
- 11. Window Width High Register
- 12. Window Height Low Register
- 13. Window Height High Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Bt431 Address Low Register RIO write access error
Slot = SLOT
HW Cursor = C
RIO address = XXXXXXX
Bt431 Control Register RIO write access error
Slot = SLOT
HW Cursor = C
RIO address = XXXXXXX
Data error in Bt431 REG TESTED
Slot = SLOT
HW Cursor = C
RIO address = XXXXXXX
exp = EE
act = AA
xor = OO
```

SLOT	Slot number of CG30 board under test	
С	Hardware cursor under test	
REG TESTED Bt431 register under test		
xxxxxxx	RIO address of Bt431 register under test	
EE	Expected value of register	
AA	Actual value read from register	
00	XOR of expected and actual values	

6.3.3.3 Test 61 - Both Cursors to Single Cursor Control Registers Test

This tests verifies that the Bt431 registers on both cursor chips can be simultaneously written. Each Bt431 register on the Both Cursor Registers is written simultaneously, each cursor register is then read and the data compared with the value that written to the Both Cursor Register.

Bt431 Registers tested are:

- 1. Command Register
- 2. Cursor (x) Low Register
- 3. Cursor (x) High Register
- 4. Cursor (y) Low Register
- 5. Cursor (y) High Register
- 6. Window (x) Low Register
- 7. Window (x) High Register
- 8. Window (y) Low Register
- 9. Window (y) High Register
- 10. Window Width Low Register
- 11. Window Width High Register
- 12. Window Height Low Register
- 13. Window Height High Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
Bt431 Address Low Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 Control Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 Control Register RIO read access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Data error in Bt431 REG TESTED
  Slot = SLOT
  HW Cursor = C
  Write RIO address = WW
  Read RIO address = RR
  exp = EE
  act = AA
  xor = 00
```

Where:

SLOT	Slot number of CG30 board under test
С	Hardware Cursor under test
REG TESTED	Bt431 register under test
WW	RIO address of Bt431 register written
RR	RIO address of Bt431 register read
EE	Expected value when read
AA	Actual value read
00	XOR of expected and actual values.

6.3.3.4 Test 62 - Single Cursor Pattern RAM Test

•

This test verifies the integrity of the pattern RAM internal to the Bt431 chip. The test performs a simple write, read, compare test on the entire pattern RAM on both Bt431 components.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Bt431 Address Low Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 Address High Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 RAM Data Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Data error in Bt431 Pattern RAM
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
  RAM index = II
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
С	Hardware cursor under test
xxxxxxx	RIO address of Bt431 register under test
II	Index into the Pattern RAM
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.3.5 Test 63 - Both Cursors to Single Cursor Pattern RAM Test

This test verifies that both Bt431 chip pattern RAM's can be written to simultaneously when using the Both Cursor Registers. The test writes to the Both Cursor Pattern RAM register (which should result in each Bt431 being written), each Bt431 chip pattern RAM is read and then compared with the value written. This sequence is performed for a series of patterns.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
Bt431 Address Low Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 Address High Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 RAM Data Register RIO write access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Bt431 RAM Data Register RIO read access error
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
Data error in Bt431 Pattern RAM
  Slot = SLOT
  HW Cursor = C
  RIO address = XXXXXXXX
  RAM index = II
  exp = EE
  act = AA
  xor = 00
```

Where:

SLOT	Slot number of CG30 board under test
С	Hardware cursor under test
xxxxxxx	RIO address of Bt431 register under test
II	Index into the Pattern RAM
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.4 Shadow RAM Tests

The following tests verify that the CG30 colormap shadow RAM is functional.

6.3.4.1 Test 64 - Shadow RAM Test

The colormap shadow RAM is used to load the color map used for the frame buffer. It contains 256 one byte entries for each of red, green and blue. This test verifies the integrity of the shadow RAM using the MARCH test algorithm (see Appendix B) using double-byte accesses.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Colormap Shadow RAM RIO write access error
During tagging of RAM
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Colormap Shadow RAM RIO write access error
During writing of pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Colormap Shadow RAM RIO read access error
During read of complement pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Colormap Shadow RAM RIO read access error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Colormap Shadow RAM Data error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
Colormap Shadow RAM Data error
During read of complement pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

- SLOT Slot number of CG30 board under test
- XXXXXXXX RIO address where error occured
- EE Expected value of register
- AA Actual value read from register
- 00 XOR of expected and actual values

6.3.4.2 Test 65 - Shadow RAM to Bt458 Colormap Update Test

The colormap shadow RAM is copied to the Bt458 colormap during vertical retrace when the Update Colormap (UC) bit on the Compatible Space Status Register is set. This test verifies that this functionality is under operation by first writing to the shadow RAM, turning the UC bit at the beginning of the vertical retrace (which should result in the Bt458 main colormap being updated), turning the UC bit off, then reading athe Bt458 and comparing it to the data written into the shadow RAM. This sequence is performed for a series of patterns.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Colormap Shadow RAM RIO write access error
  Slot = SLOT
 RIO address = XXXXXXXX
Timeout on Vertical Blanking Bit (waiting for HIGH)
  on Control/Status Register
  Slot = SLOT
 RIO address = XXXXXXXX
Timeout on Vertical Blanking Bit (waiting for LOW)
  on Control/Status Register
  Slot = SLOT
 RIO address = XXXXXXXX
Bt458 Colormap Address Register RIO write access error
  Slot = SLOT
  RIO address = XXXXXXXX
Bt458 Colormap Palette Data Register RIO read access error
  Slot = SLOT
  RIO address = XXXXXXXX
Data error in Bt458 Main CLUT Update
 from Colormap Shadow RAM during Vertical Retrace
  Slot = SLOT
  RAM address = XXXXXXXX
  CLUT index = II
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address where error occured
II	Index into the CLUT
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.5 Frame Buffer RAM Tests

The following tests verify that the eight planes in the CG30 Board frame buffer are accessable in the plane major and pixel major modes.

6.3.5.1 Test 66 - Plane Major Mode Test (affected by prompt)

In plane major access, each of the eight planes in the frame buffer can be access independently. The organization is one bit per pixel. Each plane consists of 128 Kbytes of memory.

This test verifies the integrity of the frame buffer RAM using the MARCH test algorithm (see Appendix B). The test is performed using all the RIO access types. When the prompt flag is not set, only quad-byte accesses are performed, otherwise all access types (quad-byte, double-byte, byte) are used.

If the state of the prompt flag is not set, all planes will be tested in sequence from plane 0 to plane 7. In the other hand, if the prompt flag is set, a menu is presented which choices of plane(s) to test. The user is asked to select the plane(s) that are to be tested. Following is an example of how this works;

```
Plane to test:

1) Plane 0

2) Plane 1

3) Plane 2

4) Plane 3

5) Plane 4

6) Plane 5

7) Plane 6

8) Plane 7

9) All of the above

Which Plane ?
```

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Plane Major Access of Plane P RAM RIO write access error
During tagging of RAM
  Slot = SLOT
 RAM RIO address = XXXXXXXX
Plane Major Access of Plane P RAM RIO write access error
During writing of pattern
 Slot = SLOT
 RAM RIO address = XXXXXXXX
Plane Major Access of Plane P RAM RIO read access error
During read of complement pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Plane Major Access of Plane P RAM RIO read access error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Plane Major Access of Plane P RAM Data error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
Plane Major Access of Plane P RAM Data error
During read of complement pattern
  Slot = SLOT
 RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

P	Plane under test
SLOT	Slot number of CG30 board under test
XXXXXXX	RIO address where error occured
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.5.2 Test 67 - Pixel Major Mode Test (affected by prompt)

In pixel major access, all planes are accessed simultaneously. The organization is one pixel per byte. Plane 0 correspond to bit 0 of the byte and plane 7 corresponds to bit 7 of the byte. The entire frame buffer consists of 1 Mbyte of memory.

This test verifies the integrity of the frame buffer RAM using the MARCH test algorithm (see Appendix A). The test is performed using all the RIO access types. When the prompt flag is not set, only quad-byte accesses are performed, otherwise all access types (quad-byte, double-byte, byte) are used.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Pixel Major Access Frame Buffer RAM RIO write access error
During tagging of RAM
  Slot = SLOT
 RAM RIO address = XXXXXXXX
Pixel Major Access Frame Buffer RAM RIO write access error
During writing of pattern
  Slot = SLOT
 RAM RIO address = XXXXXXXX
Pixel Major Access Frame Buffer RAM RIO read access error
During read of complement pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Pixel Major Access Frame Buffer RAM RIO read access error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
Pixel Major Access Frame Buffer RAM Data error
During read of background pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
Pixel Major Access Frame Buffer RAM Data error
During read of complement pattern
  Slot = SLOT
  RAM RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address where error occured
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.5.3 Test 68 - Pixel Major to Plane Major Test

This test verifies that accesses in Pixel major mode and plane major mode both refer to the same memory address. The conversion from pixel mode to plane mode is done by testing the data path for a series of patterns and for all combinations of enabled/disabled planes in the Plane Mask register.

The plane mask register affects both read and write operations in pixel mode. Only planes whose corresponding bit in the Plane Mask register is set will be modified by write operations. The test writes in pixel mode and reads in plane mode, checking for correct data.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Data error in Pixel Major to Plane Major Frame Buffer Access
 while Plane P was enabled
  Slot = SLOT
 Pixel Major Access address = XXXXXXXX
 Plane Major Access address = YYYYYYYY
  exp = EE
  act = AA
 xor = 00
Data error in Pixel Major to Plane Major Frame Buffer Access
 while Plane P was disabled
  Slot = SLOT
 Pixel Major Access address = XXXXXXXX
 Plane Major Access address = YYYYYYY
  exp = EE
  act = AA
  xor = 00
```

Where:

P	Plane under test
SLOT	Slot number of CG30 board under test
xxxxxxx	Pixel Major Mode RIO address where error occured
YYYYYYY	Pixel Major Mode RIO address where error occured
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6 ROP Tests

The following sections contain test descriptions for the Raster Operation Processor (ROP) chips on the CG30 Board.

6.3.6.1 Registers Tests

The following tests verify that the ROP chips are accessable and functioning correctly.

6.3.6.1.1 Test 69 - Single Plane Registers Test

This test verifies that the ROP registers on the CG30 board can be accessed and checks these registers for data retention. Each register is tested for a series of patterns which are written, read back and compared. The test is performed on each plane's ROP chip.

Registers tested are:

- 1. Destination Register
- 2. Source 1 Register
- 3. Source 2 Register
- 4. Pattern Register
- 5. Mask 1 Register
- 6. Mask 2 Register
- 7. Shift Value Register
- 8. Function Register
- 9. Width Register
- 10. OP Counter Register
- 11. Flag Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
ROP TESTED Register RIO write access error

ROP Register = ROP REG

Slot = SLOT

RIO address = XXXXXXXX

Data error in ROP TESTED Register

ROP Register = ROP REG

Slot = SLOT

RIO address = XXXXXXX

exp = EE

act = AA

xor = OO
```

SLOT	Slot number of CG30 board under test
ROP TESTED	Name of ROP under test
ROP REG	ROP register under test
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.1.2 Test 70 - All Planes to Single Plane Registers Test

This tests verifies that the ROP registers on all ROP chips may be simultaneously written. Each ROP register on the All ROP register is written, each plane ROP chip register is then read and the data compared with the value that written to the All ROP register.

Registers tested are:

- 1. Destination Register
- 2. Source 1 Register
- 3. Source 2 Register
- 4. Pattern Register
- 5. Mask 1 Register
- 6. Mask 2 Register
- 7. Shift Value Register
- 8. Function Register
- 9. Width Register
- 10. OP Counter Register
- 11. Flag Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
All Planes ROP Register RIO write access error
  ROP Register = ROP REG
  Slot = SLOT
  RIO address = XXXXXXXX
ROP TESTED Register RIO read access error
  ROP Register = ROP REG
  Slot = SLOT
  RIO address = XXXXXXXX
Data error in ROP TESTED Register
  ROP Register = ROP REG
  Slot = SLOT
  Write RIO address = WWWWWWWW
  Read RIO address = RRRRRRRR
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
ROP TESTED	Name of ROP under test
ROP REG	ROP register under test
wwwwwww	RIO address of ROP register written
RRRRRRR	RIO address of ROP register read
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.1.3 Test 71 - BTLA Mode Single Plane Registers Test

This test verifies that accesses to the ROP chips using the Byte Transfer Load Alternate (BTLA) mode follows the specified format. In this access mode the data path is tested for a series of patterns for each individual ROP chip.

A series of patterns are written, read back, and then compared in order to verify that the correct action took plane.

Registers tested are:

- 1. Destination Register
- 2. Source 1 Register
- 3. Source 2 Register

- 4. Pattern Register
- 5. Mask 1 Register
- 6. Mask 2 Register
- 7. Shift Value Register
- 8. Function Register
- 9. Width Register
- 10. OP Counter Register
- 11. Flag Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
ROP TESTED Register RIO write access error
 ROP Register = ROP REG
  Slot = SLOT
 RIO address = XXXXXXXX
ROP TESTED Register RIO read access error
 ROP Register = ROP REG
  Slot = SLOT
 RIO address = XXXXXXXX
Data error in ROP TESTED Register
 ROP Register = ROP REG
  Slot = SLOT
 Write RIO address = WWWWWWWW
 pattern = PP
 Read RIO address = RRRRRRRR
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
ROP TESTED	Name of ROP under test
ROP REG	ROP register under test
WWWWWWW	RIO address of ROP register written
RRRRRRR	RIO address of ROP register read
PP	Pattern written to register
EE	Expected value of register

- AA Actual value read
- 00 XOR of expected and actual values

6.3.6.1.4 Test 72 - BTLA Mode All Planes to Single Plane Registers Test

This tests verifies that the ROP registers on all ROP chips may be simultaneously written using Byte Transfer Load Alternate mode. Each ROP register on the All ROP register space is written, each plane's ROP chip register is then read and the data compared with the value that written thru the All ROP register space.

The plane mask register affects both read and write operations in pixel mode. Only planes whose corresponding bit in the Plane Mask register is set will be modified by write operations. This condition is checked by writing a series of patterns for all combinations of enabled/disabled planes in the Plane Mask register.

Registers tested are:

- 1. Destination Register
- 2. Source 1 Register
- 3. Source 2 Register
- 4. Pattern Register
- 5. Mask 1 Register
- 6. Mask 2 Register
- 7. Shift Value Register
- 8. Function Register
- 9. Width Register
- 10. OP Counter Register
- 11. Flag Register

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

U

```
BTLA All Planes ROP Register RIO write access error
  ROP Register = ROP REG
  Slot = SLOT
  RIO address = XXXXXXXX
ROP TESTED Register RIO write access error
  ROP Register = ROP REG
  Slot = SLOT
  RIO address = XXXXXXXX
Data error in ROP TESTED Register
 while Plane P was enable
 ROP Register = ROP REG
  Slot = SLOT
  Write RIO address = WWWWWWWW
  pattern = PP
  Read RIO address = RRRRRRRR
  exp = EE
  act = AA
  xor = 00
Data error in ROP TESTED Register
 while Plane P was disable
  ROP Register = ROP REG
  Slot = SLOT
  Write RIO address = WWWWWWWW
  pattern = PP
  Read RIO address = RRRRRRRR
  exp = EE
  act = AA
  xor = 00
```

SLOT	Slot number of CG30 board under test
P	Number of Plane under test
ROP TESTED	Name of ROP under test
ROP REG	ROP register under test
wwwwwwww	RIO address of ROP register written
RRRRRRR	RIO address of ROP register read
PP	Pattern written to register
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2 Mode Tests

The following tests verify the operation of the ROP chips in each operation mode.

6.3.6.2.1 Test 73 - Mode-0 Read Access Test

This test verifies that read accesses to the frame buffer using the ROP mode address space during ROP mode 0 in fact result in a load of the ROP destination register from the frame buffer. This test verifies the address and data paths between ROPs and the frame buffer. The test performs a simple sequence of writes to the frame buffer in plane mode, reads from the ROP mode space, and assures that the ROP destination register gets loaded with the pattern that was written in the frame buffer.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error
during ROP Mode-0 Frame Buffer Access
  Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
 Slot = SLOT Plane = PLANE
 RIO address = PPPPPPPP
ROP Destination Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP Destination Register data error
 during ROP Mode-0 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Destination Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test

RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.2 Test 74 - Mode-1 Read Access Test

This test verifies that read accesses to the frame buffer using the ROP mode address space during ROP mode 1 results in a load of the ROP destination register from the frame buffer. This test verifies the address and data paths between ROPs and the frame buffer. The test performs a simple sequence of writes to the frame buffer in plane mode, reads from the ROP mode space, and assures that the ROP destination register gets loaded with the pattern that was written in the frame buffer.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
RIO read access error
during ROP Mode-1 Frame Buffer Access
 Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
 RIO address = PPPPPPP
ROP Destination Register RIO read access error
 Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP Mode Access data error
during ROP Mode-1 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
 ROP Mode Access RIO address = RRRRRRR
 Plane Major Access RIO address = PPPPPPP
 ROP Destination Register RIO address = XXXXXXXX
 exp = EEEE
 act = AAAA
 xor = 0000
ROP Destination Register data error
 during ROP Mode-1 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Destination Register RIO address = XXXXXXXX
  exp = EEEE
  act = AAAA
  xor = 0000
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
RRRRRRR	ROP Mode RIO address
PPPPPPP	Plane Major RIO address
XXXXXXXX	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register

XOR of expected and actual values

6.3.6.2.3 Test 75 - Mode-4 Read Access Test

00

This test verifies that read accesses to the frame buffer using the ROP mode address space during ROP mode 4 results in a load of the ROP destination and source registers from the frame buffer. This test verifies the address and data paths between ROPs and the frame buffer. The test performs a simple sequence of writes to the frame buffer in plane mode, reads from the ROP mode space, and assures that the ROP source and destination register get loaded with the pattern that was written in the frame buffer.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.
```
ROP Shift Value Register RIO write access error
  Slot = SLOT
              Plane = PLANE
 RIO address = XXXXXXXX
RIO read access error
during ROP Mode-4 Frame Buffer Access
  Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
 RIO address = PPPPPPP
ROP Destination Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP Source 2 Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP Destination Register data error
 during ROP Mode-4 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Destination Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
ROP Source 2 Register data error
 during ROP Mode-4 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Source 2 Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test

RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.4 Test 76 - Mode-5 Read Access Test

This test verifies that read accesses to the frame buffer using the ROP mode address space during ROP mode 5 results in a load of the ROP destination register from the frame buffer. This test verifies the address and data paths between ROPs and the frame buffer. The test performs a simple sequence of writes to the frame buffer in plane mode, reads from the ROP mode space, and assures that the ROP destination register gets loaded with the pattern that was written in the frame buffer.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
ROP Shift Value Register RIO write access error
  Slot = SLOT
              Plane = PLANE
  RIO address = XXXXXXXX
RIO read access error
 during ROP Mode-6 Frame Buffer Access
  Slot = SLOT Plane = PLANE
  RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
  RIO address = PPPPPPP
ROP Source 2 Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP Source 2 Register data error
 during ROP Mode-6 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Source 2 Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
RRRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

-6.3.6.2.5 Test 77 - Mode-6 Read Access Test

This test verifies that read accesses to the frame buffer using the ROP mode address space during ROP mode 6 results in a load of the ROP source register from the frame buffer. This test verifies the address and data paths between ROPs and the frame buffer. The test performs a simple sequence of writes to the frame buffer in plane mode, reads from the ROP mode space, and assures that the ROP source register gets loaded with the pattern that was written in the frame buffer.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO read access error
 during ROP Mode-0 Frame Buffer Access
  Slot = SLOT Plane = PLANE
  RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
 Slot = SLOT Plane = PLANE
 RIO address = PPPPPPP
ROP Destination Register RIO read access error
 Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP Destination Register data error
 during ROP Mode-0 Read Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Destination Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
XXXXXXXX	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.6 Test 78 - Mode-0 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 0 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
ROP Destination Register RIO write access error
  Slot = SLOT
              Plane = PLANE
  RIO address = XXXXXXXX
RIO write access error
 during ROP Mode-0 Frame Buffer Access
  Slot = SLOT Plane = PLANE
  RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
 Slot = SLOT Plane = PLANE
 RIO address = PPPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-0 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-0 Write Access to Frame Buffer
  Slot = SLOT
               Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address

PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.7 Test 79 - Mode-1 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 1 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
ROP Destination Register RIO write access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
RIO write access error
during ROP Mode-1 Frame Buffer Access
  Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
  RIO address = PPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-1 Write Access to Frame Buffer
  Slot = SLOT
              Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-1 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address

PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.8 Test 80 - Mode-2 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 2 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

4

```
RIO write access error
during ROP Mode-2 Frame Buffer Access
  Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
 RIO address = PPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP ALU Output Register data error
during ROP Mode-2 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-2 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
 Plane Major Access RIO address = PPPPPPP
 ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register

- AA Actual value read from register
- 00 XOR of expected and actual values

6.3.6.2.9 Test 81 - Mode-3 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 3 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
RIO write access error
 during ROP Mode-3 Frame Buffer Access
 Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
 Slot = SLOT Plane = PLANE
 RIO address = PPPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-3 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-3 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register

- AA Actual value read from register
- 00 XOR of expected and actual values

6.3.6.2.10 Test 82 - Mode-4 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 4 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
ROP Destination Register RIO write access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP Source 1 Register RIO write access error
 Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP Source 2 Register RIO write access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
RIO write access error
 during ROP Mode-4 Frame Buffer Access
 Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO read access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
 RIO address = PPPPPPPP
ROP ALU Output Register RIO read access error
 Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-4 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-4 Write Access to Frame Buffer
  Slot = SLOT
               Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

and the second second

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.11 Test 83 - Mode-5 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 5 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
ROP Destination Register RIO write access error
 Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
RIO write access error
during ROP Mode-5 Frame Buffer Access
 Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO read access error
during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
 RIO address = PPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-5 Write Access to Frame Buffer
 Slot = SLOT
              Plane = PLANE
 ROP function = SRC FUNCTION DST
 ROP Mode Access RIO address = RRRRRRR
 Plane Major Access RIO address = PPPPPPP
 ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-5 Write Access to Frame Buffer
 Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address

PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.12 Test 84 - Mode-6 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 6 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Solbourne Confidential Information - Do Not Distribute

```
ROP Source 1 Register RIO write access error
 Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
ROP Source 2 Register RIO write access error
  Slot = SLOT Plane = PLANE
 RIO address = XXXXXXXX
RIO write access error
 during ROP Mode-6 Frame Buffer Access
 Slot = SLOT Plane = PLANE
 RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
  RIO address = PPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-6 Write Access to Frame Buffer
  Slot = SLOT
              Plane = PLANE
  ROP function = SRC FUNCTION DST
 ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-6 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

Where:

PLN ACC TYPEName of Plane under testSLOTSlot number of CG30 board under test

PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register
AA	Actual value read from register
00	XOR of expected and actual values

6.3.6.2.13 Test 85 - Mode-7 Write Access Test

This test verifies that write accesses to the frame buffer using the ROP mode address space during ROP mode 7 results in a load of the frame buffer with the correct result for the AND, OR, and XOR logical operations performed by each individual ROP on a series of patterns. This test verifies the address and data paths between ROPs and the frame buffer as well as some basic ROP chip functionality.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
RIO write access error
 during ROP Mode-7 Frame Buffer Access
  Slot = SLOT Plane = PLANE
  RIO address = RRRRRRRR
RIO write access error
 during PLN ACC TYPE
  Slot = SLOT Plane = PLANE
  RIO address = PPPPPPPP
ROP ALU Output Register RIO read access error
  Slot = SLOT Plane = PLANE
  RIO address = XXXXXXXX
ROP ALU Output Register data error
 during ROP Mode-7 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
 ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
RAM Data error during ROP Mode-7 Write Access to Frame Buffer
  Slot = SLOT Plane = PLANE
  ROP function = SRC FUNCTION DST
  ROP Mode Access RIO address = RRRRRRR
  Plane Major Access RIO address = PPPPPPP
  ROP Output Register RIO address = XXXXXXXX
  exp = EE
  act = AA
  xor = 00
```

PLN ACC TYPE	Name of Plane under test
SLOT	Slot number of CG30 board under test
PLANE	Plane under test
FUNCTION	Logical function (AND, OR, XOR)
RRRRRRR	ROP Mode RIO address
PPPPPPPP	Plane Major RIO address
xxxxxxx	RIO address of ROP register under test
EE	Expected value of register

AA	Actual value read from register
----	---------------------------------

00 XOR of expected and actual values

6.3.7 Retrace Interrupts Tests

The following tests verify the functions associated with the interrupt logic on the CG30 Board.

6.3.7.1 Test 86 - Vertical Retrace Interrupts Test

This test verifies the correct operation of the frame buffer to generate vertical retrace interrupts as well as the interrupt related bits on the Compatible Space Status and Control/Status registers.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Interrupt pending bit did not get set
on Compatible Space Status Register
 Slot = SLOT
Timeout occured while waiting for vertical retrace interrupt
 Slot = SLOT
  expected vector = EE
Vertical blanking bit did not get set
on Control Status Register
  Slot = SLOT
 RIO address = XXXXXXXX
Vertical blanking bit did not get set
on Compatible Space Status Register
  Slot = SLOT
 RIO address = XXXXXXXX
Incorrect vertical retrace interrupt vector received
  Slot = SLOT
  expected vector = EE
  received vector = RR
```

Where:

SLOT Slot number of CG30 board under test

XXXXXXXX RIO address of register under test

- EE Expected vector to be received after interrupt
- RR Actual vector received after interrupt

6.3.7.2 Test 87 - Framecount Register (Counter) Test

This test verifies the correct operation of the frame buffer to generate vertical retrace interrupts as well as its ability to maintain the framecount register. The framecount register is an eight bit register thats allows the CPU to count vertical periods. It is incremented at the beginning of every vertical blanking period. When it reaches the count of 255, it is rolled over to zero. This tests checks that the interrupt pending bit on the Compatible Space Status register gets set and verifies that the framecount register is incremented and resets back to zero after reaching the count of 255.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

Possible errors detected by this test:

```
Timeout occured while waiting for vertical retrace interrupt
  Slot = SLOT
  expected vector = VV
Incorrect vertical retrace interrupt vector received
  Slot = SLOT
  expected vector = VV
  received vector = RR
Framecount Register did not increment
after a vertical retrace interrupt occurred
  Slot = SLOT
  exp = EE
  act = AA
Framecount Register did not reset
back to zero after count reached 255
  Slot = SLOT
  read = AA
```

SLOT	Slot number of CG30 board under test
xxxxxxx	RIO address of register under test
vv	Expected vector to be received after interrupt
RR	Actual vector received after interrupt
EE	Expected count to be read from Framecount register
RR	Actual count read from Framecount register

6.3.7.3 Test 88 - Retrace/SCC Interrupt Combination Test

This test verifies the correct operation of the interrupt prioritization logic on the CG30 Board when simultaneous vertical retrace and SCC interrupts are pending.

The test programs the SCC (internal loopback mode) and enables vertical retrace interrupts. The test then waits until a vertical retrace is about to occur and writes a character to the SCC port. When the vertical retrace does occur, there should be three interrupts pending: SCC Transmit buffer empty; Vertical Retrace; and SCC Receive character available.

The test verifies that the interrupt are received in the correct order and that the received vectors are correct.

The following errors could occur:

```
RIO timeout error occurred while programming the Z8530
Port = PORT
```

Where:

PORT CG30 Mouse, CG30 Keyboard

The above error indicates a data fault occurred while the test was programming the Z8530 SCC port.

```
ERROR occurred on SCC write
Port = PORT
```

Where:

ERRORTimeout error, Framing error, Parity error, or Data overrun errorPORTCG30 Mouse, CG30 Keyboard

The above error indicates an error occurred when writing a character to be transmitted by the specified port.

```
Timeout waiting for 1st interrupt from CG30 board
   Slot = SLOT
```

Where:

SLOT Slot number of CG30 Board under test

The above error indicates that the CPU timedout waiting for the 1st (SCC) interrupt to be received.

```
1st interrupt was not the SCC vector
Slot = SLOT
expected vector 0x8d
received vector 0xVV
```

SLOT Slot number of CG30 Board under test

vv Actual received interrupt vector

The above error indicates that the interrupts were received out of order.

```
Timeout waiting for 2nd interrupt from CG30 board
   Slot = SLOT
```

Where:

SLOT Slot number of CG30 Board under test

The above error indicates that the CPU timed out waiting for the 2nd (vertical retrace) interrupt to be received.

```
2nd interrupt was not the vertical retrace vector
Slot = SLOT
expected vector 0x85
received vector 0xVV
```

Where:

SLOT	Slot number of CG30 Board under test
vv	Actual received interrupt vector

The above error indicates that the interrupts were received out of order.

```
Timeout waiting for 3rd interrupt from CG30 board
Slot = SLOT
```

Where:

SLOT Slot number of CG30 Board under test

The above error indicates that the CPU timed out waiting for the 3rd (SCC) interrupt to be received.

```
3rd interrupt was not the SCC vector
Slot = SLOT
expected vector 0x8d
received vector 0xVV
```

SLOT Slot number of CG30 Board under test

VV Actual received interrupt vector

The above error indicates that the interrupts were received out of order.

ERROR occurred on SCC read Port = PORT

Where:

ERROR Timeout error, Framing error, Parity error, or Data overrun error

PORT CG30 Mouse or CG30 Keyboard

The above error indicates an error occurred when reading the character received by the specified port

6.3.8 Serial Ports Tests

The following tests verify that the Z8530 SCC chips on the CG30 Board are accessable and functional. Note that the interface between the SCC and the keyboard and mouse is not tested at this time. This must be done manually by connecting a keyboard and mouse and verifying the correct operation.

6.3.8.1 Test 89 - Serial Ports Reset Test

This test verifies the reset state of the Z8530 Serial Communication Controller (SCC) chip for both keyboard and mouse ports. This test also verifies that both ports are accessible.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
Timeout occurred while resetting the Z8530
Port = PORT TESTED
Timeout occurred while reading register REG TESTED
Port = PORT TESTED
PORT TESTED reset test failure
failing register = REG TESTED
exp = EE
act = AA
xor = OO
```

PORT TESTED	Name of CG30 serial port under test
REG TESTED	Name of Z8530 failing register
EE	Expected value to be read from address
AA	Actual value read from address
00	XOR of expected and actual values

6.3.8.2 Test 90 - Serial Ports Loopback Test

This test performs a loopback test for each channel in the Z8530 Serial Communication Controller (SCC) chip for both keyboard and mouse ports. This test also verifies that the interrupt logic for both serial ports is functional.

Refer to Appendix C if frame buffer configuration table contains multiple CG30 boards and the test is prompted.

```
RIO timeout occurred while resetting the Z8530
 Port = PORT TESTED
RIO timeout occurred while programming the Z8530
 Port = PORT TESTED
Timeout while waiting for Receive Character Available interrupt
 Port = PORT TESTED
  expected vector = EE
Incorrect Receive Character Available interrupt vector received
 Port = PORT TESTED
  expected vector = EE
  received vector = RR
Internal Loopback Data Compare Error
 Port = PORT TESTED
  TX character = TT
  RX character = XX
  xor
              = 00
Receive Character Available interrupt pending inactive
  Port = PORT TESTED
  Z8530 interrupt vector (RR2) = ZZ
Timeout while waiting for Transmit Buffer Empty interrupt
 Port = PORT TESTED
  expected vector = EE
Incorrect Transmit Buffer Empty interrupt vector received
  Port = PORT TESTED
  expected vector = EE
  received vector = RR
Transmit Buffer Empty interrupt pending inactive
  Port = PORT TESTED
  Z8530 interrupt vector (RR2) = ZZ
```

PORT TESTEDName of CG30 serial port under testEEExpected value to be received after interruptRRActual value received after interruptTTValue transmitted to SCCXXValue received from SCC00XOR of expected and actual values

ZZ

Contents of Z8530 Status Register

Section 7: Miscellaneous Tests

7.1 Introduction

The miscellaneous dg tests are given in this section. These are special case tests which require external test fixtures (loopback connectors) and/or manual interaction.

7.2 Test 91 - Random ECC Generation Test (must be prompted)

This test is actually tool which should be used when intermittent ECC errors occur in the system and it is desired to exercise the ECC generation and checking logic on the CPU Board.

The test generates 8 Kbytes of random data in the CPUs internal cache, flushes the random data out to memory, then reads the data back into the cache. When the data is written to memory, the CPU board generates an ECC byte which is stored in memory along with the data. When the data is read, the CPU must generate a new ECC byte based on the data being read and compare it with the ECC byte returned from the memory system.

The advantage of using random data patterns is that slow or pattern sensitive components in the ECC generation and checking path on the CPU board may be located.

This test is not executed unless the prompt command is first used to enable it. When prompted, the test prints:

Generate random ECC bytes? (default yes):

If the user enters no then the test asks the following question:

Enter target ECC byte:

The target ECC byte supplied by the user to this question will be the ECC byte produced by all 8 Kbytes of random data. In other words, the test will generate 8 Kbytes of random data with each 64-bit pattern mapping to the target ECC byte. This mode of operation may be used when a specific ECC pattern is suspected. Note that the test takes considerably longer to generate 8 Kbytes of random data in this mode since many patterns which don't map to the target ECC byte are discarded.

If the user enters yes (or return) to the first question, then the test asks the following question:

Enter random seed (default 0):

The value that is entered will become the seed value which determines what sequence of random data patterns which will be generated. By changing the seed value the sequence of patterns may be changed.

The test then asks for the number of repetitions to be performed as follows:

```
Enter number of repetitions (0 means forever, default 1000):
```

One repetition consists of 8 Kbytes of random data being generated. If 0 is entered, then the test will repeat forever until the user enters Control-C (^C).

If the test fails, one of the following error message will be displayed:

```
Data fault occurred accessing block at address 0xaaaaaaa
FVAR = 0xaaaaaaaa
```

or,

```
ECCS fault occurred accessing block at address 0xaaaaaaaa
FPAR = 0xaaaaaaaa
FES = 0xbb
```

or,

```
Data error occurred at address 0xaaaaaaaa
exp = 0xeeeeeeee
act = 0xddddddd
```

where 0xaaaaaaaa is the access address, bb is the contents of the FES register, 0xeeeeeeee is the expected data, and 0xddddddd is the actual data.

7.3 Test 92 - Cache Data Bus Test (must be prompted)

Test 92 is similar to Test 91 (Random ECC Generation Test) except that alternating walking 1 and walking zero data patterns on alternating cache lines are used as the data patterns.

The test generates 8 Kbytes of alternating patterns in the CPUs internal cache, flushes the data out to memory, then reads the data back into the cache. This is intended to test the ability of the cache data bus drivers to switch between all ones and all zeros for all 64 bit positions. The test is executed with ECC enabled to detect any ECC single or double bit errors.

This test is not executed unless the prompt command is first used to enable it. When prompted the test prints:

```
Enter number of repetitions (0 means forever, default 1000):
```

One repetition consists of 8 Kbytes of data being created. If 0 is entered, the test repeats forever until the user enters Control-C ($^{\circ}$ C).

If the test fails, one of the following error messages are displayed:

```
Data fault occurred accessing block at address 0xaaaaaaaa
FVAR = 0xbbbbbbbb
```

This indicates that a data fault exception occurred when the cpu accessed address "Oxaaaaaaaaa."

```
ECCS fault occurred accessing block at address 0xaaaaaaaa
FPAR = 0xbbbbbbbb
FES = 0xccccccc
```

This indicates that an ECC single bit exception occurred when the cpu accessed the cache block at address "0xaaaaaaaaa."

```
Data error occurred at address 0xaaaaaaaa
Pass N
exp = 0xeeeeeeee
act = 0xaaaaaaaa
```

This indicates that the cpu read incorrect data at the specified address. "Pass" indicates how many repetitions were completed when the error occurred.

7.4 Test 93 - RS-232-C Connector Loopback Test (must be prompted)

This test requires that a loopback connector be placed across serial port A and serial port B on the System Board. The test will not be executed unless it has been prompted (see the **prompt** man page).

☆ ☆ ☆ NOTE ☆ ☆ ☆

System Boards at revision DA or later operate differently than earlier System Boards. Specifically, the DTR output of the serial ports was previously used to switch the sense of the RX and TX data pins and RTS was used to drive DTR to the connector. On Revision DA and later boards, this functionality was removed and the DTR pin is really transmitted as DTR.

Because of this, the RS-232-C Connector loopback test was modified to automatically check the revision of the System Board and modify Part2 of the test accordingly.

The first part of the documentation for this test deals with System Boards built prior to Revision DA. The second part details the test for boards at or above Revision DA.

쇼 쇼 쇼 NOTE 쇼 쇼 쇼

System Boards Prior to Revision DA

This test contains two parts:

- 1. Verifies receive data (RxD) and transmit data (TxD) signal continuity
- 2. Verifies RTS-CTS, and DSR-DTR continuity.

Continuity for all signals is verified from port A to port B and from port B to port A. Figure 7-1 shows the proper loopback connector signal arrangement.

It is important to note that the looped-back signals are transmitted at RS-423-A signal levels. Although the serial ports are designed to accept RS-232-C signal levels, this test does not verify that RS-232-C signal levels can be received. To verify this, it is recommended that an RS-232-C terminal device be connected and used to drive each serial port. Figure 7-2 shows the null modem cable pin arrangement.



Figure 7-1. Solbourne Serial Port Loopback Model



Figure 7-2. Null Modem Cable Pin Arrangement

The test begins by resetting both serial ports and programming them for normal operation. If a timeout error occurs while these operations are being performed, one of the following error messages may be displayed:

```
Timeout error occurred while resetting the Z8530
for serial ports A and B
Timeout occurred while programming serial port A
Timeout occurred while programming serial port B
```

Part 1 of the test verifies that characters can be transmitted from port A to port B and from port B to port A. All characters in the range 0x20 through 0x7e are transmitted in each direction.

During this part of the test, one of the following error messages may be displayed:

```
Kbus timeout occurred on transmit
    Transmit port = Serial Port N
   Receive port = Serial Port N
Timeout error occurred on transmit
    Transmit port = Serial Port N
   Receive port = Serial Port N
    TX character = XX
Kbus timeout occurred on receive
    Transmit port = Serial Port N
   Receive port = Serial Port N
Parity error occurred on receive
    Transmit port = Serial Port N
   Receive port = Serial Port N
Serial Port external data loopback test failure
    Transmit port = Serial Port N
    Receive port = Serial Port N
    TX character = XX
   RX character = XX
    xor
                  = XX
```

In the above example, *N* means either A or B; XX is any 2-digit hex number.

Part 2 of the test verifies transmission of the modem control signals for both true and false signal levels from port A to port B and from port B to port A. The following is a description of how the test is performed:

- 1. Port A RTS is set to FALSE and port B RTS is set to TRUE.
- 2. Port A CTS and SYNC inputs are read.
- 3. Port B CTS and SYNC inputs are read.
- 4. Port A CTS is verified to be TRUE.
- 5. Port A SYNC is verified to be TRUE.
- 6. Port B CTS is verified to be FALSE.
- 7. Port B SYNC is verified to be FALSE.

During this portion of the test one of the following error messages may be displayed:

Timeout occurred while setting PORTA<RTS> false Timeout occurred while setting PORTB<RTS> true Timeout occurred while reading RR0 in PORTA Timeout occurred while reading RR0 in PORTB True state not transmitted from PORTB<RTS> to PORTA<CTS> True state not transmitted from PORTB<RTS> to PORTA<SYNC> False state not transmitted from PORTA<RTS> to PORTB<CTS> False state not transmitted from PORTA<RTS> to PORTB<CTS>

Next, the state of the modem control signals are reversed and the opposite states are verified:

- 1. Port A RTS is set to TRUE and port B RTS is set to FALSE.
- 2. Port A CTS and SYNC inputs are read.
- 3. Port B CTS and SYNC inputs are read.
- 4. Port A CTS is verified to be FALSE.
- 5. Port A SYNC is verified to be FALSE.
- 6. Port B CTS is verified to be TRUE.
- 7. Port B SYNC is verified to be TRUE.

During this portion of the test one of the following error messages may be displayed:

Solbourne Confidential Information - Do Not Distribute

Timeout occurred while setting PORTA<RTS> true Timeout occurred while setting PORTB<RTS> false Timeout occurred while reading RR0 in PORTA Timeout occurred while reading RR0 in PORTB False state not transmitted from PORTB<RTS> to PORTA<CTS> False state not transmitted from PORTB<RTS> to PORTA<SYNC> True state not transmitted from PORTA<RTS> to PORTB<CTS> True state not transmitted from PORTA<RTS> to PORTB<CTS>

System Boards at or After Revision DA

This test contains two parts:

- 1. Verifies receive data (RxD) and transmit data (TxD) signal continuity
- 2. Verifies RTS-CTS, and DSR-DTR continuity.

Continuity for all signals is verified from port A to port B and from port B to port A. Figure 7-3 shows the proper loopback connector signal arrangement.

It is important to note that the looped-back signals are transmitted at RS-423-A signal levels. Although the serial ports are designed to accept RS-232-C signal levels, this test does not verify that RS-232-C signal levels can be received. To verify this, it is recommended that an RS-232-C terminal device be connected and used to drive each serial port. Figure 7-4 shows the null modem cable pin arrangement.


Figure 7-3. Solbourne Serial Port Loopback Model



Figure 7-4. Null Modem Cable Pin Arrangement

The following is a descriptino of how the test is performed:

- 1. Port A RTS is set to FALSE and DTR is set to TRUE.
- 2. Port B RTS is set to TRUE and DTR is set to FALSE.
- 3. Port A CTS and SYNC inputs are read.
- 4. Port B CTS and SYNC inputs are read.
- 5. Port A CTS is verified to be TRUE.
- 6. Port A SYNC is verified to be FALSE.
- 7. Port B CTS is verified to be FALSE.
- 8. Port B SYNC is verified to be TRUE.

During this part of the test one of the following error messages may be displayed:

Timeout occurred while setting PORTA<DTR> true Timeout occurred while setting PORTB<RTS> true Timeout occurred while reading RR0 in PORTA Timeout occurred while reading RR0 in PORTB True state not transmitted from PORTB<RTS> to PORTA<CTS> False state not transmitted from PORTB<DTR> to PORTA<SYNC> False state not transmitted from PORTA<RTS> to PORTA<CTS> True state not transmitted from PORTA<RTS> to PORTA<SYNC>

Next, the state of the modem control signals are reversed and the opposite states are verified:

- 1. Port A RTS is set to TRUE and DTR is set to FALSE.
- 2. Port B RTS is set to FALSE and DTR is set to TRUE.
- 3. Port A CTS and SYNC inputs are read.
- 4. Port B CTS and SYNC inputs are read.
- 5. Port A CTS is verified to be FALSE.
- 6. Port A SYNC is verified to be TRUE.
- Port B CTS is verified to be TRUE.
- 8. Port B SYNC is verified to be FALSE.

During this part of the test one of the following error messages may be displayed:

Timeout occurred while setting PORTA<RTS> true Timeout occurred while setting PORTB<DTR> true Timeout occurred while reading RR0 in PORTA Timeout occurred while reading RR0 in PORTB False state not transmitted from PORTB<RTS> to PORTA<CTS> True state not transmitted from PORTB<DTR> to PORTA<SYNC> True state not transmitted from PORTA<RTS> to PORTA<CTS> False state not transmitted from PORTA<RTS> to PORTA<CTS>

7.5 Test 94 - IDPROM checksum tests (affected by prompt)

This test is used to verify that the contents of the IDPROM on the selected board(s) contains the correct pattern. Verification is performed by summing up the contents of all the bytes on the IDPROM and by making sure that the least significant byte of the result is of zero value.

This test may be executed in two modes depending on the state of the prompt flag for this test. If the prompt flag is set, the test presents a menu of checksum tests specific to each of the following boards: System Board, Memory Board, and Graphics Board. The user is then ask to select the test(s) that are to be executed. Following is an example of this mode of execution:

```
Available tests are:
    1) System Board IDPROM checksum test
    2) Memory Board IDPROM checksum test
    3) Graphics Board IDPROM checksum test
    4) All of the above
Test number:
```

When the prompt flag is not set, all of the available tests will be executed.

쇼 쇼 쇼 NOTE 쇼 쇼 쇼

If either the memory or frame buffer configuration tables are not configured, tests for the above mention boards will not be executed and a message printed warning the user that the test was skipped.

Possible errors detected by this test:

```
Timeout error occurred reading IDPROM size
from BOARD TYPE board in slot SLOT
  IDPROM base address = BB
 header number = H
 header offset = 00
  address of size field = SS
Zero size field read from IDPROM
on BOARD TYPE board in slot SLOT
  IDPROM base address = BB
 header number = H
 header offset = 00
Timeout error occurred while generating checksum
on BOARD TYPE board in slot SLOT
  IDPROM base address = BB
 header number = H
 header offset = 00
  failing address = FF
Timeout error occurred reading optional header field
on BOARD TYPE board in slot SLOT
  IDPROM base address = BB
 header number = H
 header offset = 00
  failing address = FF
IDPROM checksum error for header H
on BOARD TYPE board in slot SLOT
  IDPROM base address = BB
  header offset = 00
  low byte of checksum should be 0x00
  actual checksum = CC
```

Where:

BOARD TY	PE Class of board under test
SLOT	Slot number of board under test
BB	Base address of IDPROM under test
н	IDPROM header number
00	Offset addres of header from IDPROM base
FF	RIO address of failure
сс	Expected checksum
SS	RIO address of size byte in IDPROM

7.6 Test 95 - Ethernet TFTP Read Test

The ethernet tftp read test verifies the Solbourne workstation's ability to read a file from a host servers's file system.

The file system on the server must have a "/usr/stand" directory which contains the "tftpread.data" file. The host server must also be set up to be a tftp file server. This means that the /etc/hosts and /etc/ethers file must contain entries which map the ethernet address of the workstation under test to its internet address.

The "DIAGSERVER" ROM environment variable on the Solbourne workstation under test must be set to accurately point to the directory containing tftpread.data on the server. For example, "DIAGSERVER = tftp.ex(,,1)/usr/stand". tftp.ex is the name associated with the EXELAN ethernet device, tftp.ei is the name associated with the IOASIC ethernet device.

If the "DIAGSERVER" ROM environment variable is not set, the test will print:

```
Environment variable DIAGSERVER not initialized Test Skipped.
```

When the test attempts to open the file on the remote file system a "rarp" (reverse arp) operation is performed in which the workstation sends out its Ethernet address and the server responds by sending back the internet address of the requesting workstation along with the internet address of the responding server.

If the server fails to respond, a string of messages like the following will be seen as the driver keeps retrying the rarp using its ethernet address. If the server never responds, the rarp request will timeout.

```
rarp: requesting internet address for 8:0:14:20:38:92
rarp: timeout on request
```

If any of the above situations arise, the user should make sure that the server and workstation under test have been setup correctly and that all Ethernet cable connections are secure.

If the server responds normally, several "rarp" messages will be seen on the console:

```
rarp: using IP address 192.9.201.140 = C009C98C
rarp: server at IP address 192.9.201.27 = C009C91B
```

In the above messages, the first message represents IP address of the workstation and the second message represents the IP address of the responding server.

If the initial open of the file pointed to by the "DIAGSERVER" variable fails, the test will print the following message and halt.

tftp.ex(,,1)/usr/stand/tftpread.data cannot be opened

If the file open is successful, the test reads the data packets sent from the server and compares the data with a re-generated copy of the data. If a read fails, the test will print the following message:

```
Read Error: tftp.ex(,,1)/usr/stand/tftpread.data, fd = 3, count = 0
```

Where fd is the file descriptor returned from the file open operation and count is the number of bytes returned from the read call. The read operation always requests four bytes at a time.

If there are no errors returned from the read operation, the test compares the data read with expected data. If there is a data miscompare, the test will print the following messages and halt:

```
Data Error: tftp.ex(,,1)/usr/stand/tftpread.data, fd = 3
byte number 0;
    exp = 0x08a6c2b1
    act = 0xff08a62c
```

A data miscompare error indicates that the data stream was corrupted by the Ethernet device (EXELAN or ASIC) during the time it was received from the network cable and placed in system memory. In any case, the entire data path (cable, Ethernet device and System Board) should be suspect.

7.7 Test 96 - Disk Write/Read Test (affected by prompt)

The Disk write/read test verifies the ability of the Solbourne workstation to write data to a disk and read it back. The test is device independent in that it will run on any disk device supported by the standalone driver (SCSI or SMD).

The test operates in two distinct modes: prompted, and non-prompted (see the prompt(1) command on how to prompt tests).

In non-prompted mode, the test performs 128 Kbyte writes and reads on the disk partition specified by the DEFAULTSWAP ROM environment variable. The DEFAULTSWAP variable must be valid for the test to operate correctly. In addition, incorrectly setting the DEFAULTSWAP device could result in corruption of disk data.

If the DEFAULTSWAP variable has not been initialized, the test will print the following message:

```
Environment variable DEFAULTSWAP not initialized
Test Skipped.
```

If the DEFAULTSWAP variable has been initialized. The test attempts to open the specified device. If the open fails for any reason, the test will print the following message and halt (assuming DEFAULTSWAP is set to sd.si(,1)):

Default swap device sd.si(,,1) cannot be opened

Once the specified disk device has been successfully opened, the test calls a driver function which returns the number of disk blocks associated with the specified partition. If the specified partition is of zero size, the test prints the following message and halts:

Null size for device partition sd.si(,,1)

If any of the above messages are displayed, the user should verify that the DEFAULTSWAP variable has been correctly set up and that the signal and power cableing on the target disk system is connected and intact.

Once passed the above checks, the test fills a 128 Kbyte block of physical memory with a repeating pattern of 0xb6d9, writes the data buffer to disk block 0 then reads the disk data back into another 128 Kbyte memory buffer.

If the call to the write function returns an error, the test will print the following message and halt:

```
Write Error: device sd.si(,,1), fd = 3, count = 0x0
target block = 0
block address = 0x00000000
transfer size = 0x00020000
write buffer = 0xff0c0000 - 0xff0dffff
```

After the disk write has been performed, the test zeroes out the buffer that will be used during the read, then reads the disk into the buffer.

```
Read Error: device sd.si(,,1), fd = 3, count = 0x0
target block = 0
block address = 0x00000000
transfer size = 0x00020000
read buffer = 0xfffe0000 - 0xffffffff
```

After the disk read has been completed, the test verifies that the write buffer data matches the read buffer data. If there is a data miscompare, the test will print the following message and halt:

```
Data Compare Error: device = sd.si(,,1), fd = 3
target block = 0
block address = 0x00000000
write buffer = 0xff0c0000 - 0xff0dffff
read buffer = 0xffe0000 - 0xffffffff
error occurred at buffer offset 0x100
    expected: (0xff0c0100) = 0xb6
    actual: (0xfffe0100) = 0x00
```

In prompted mode, the test asks the operator explicit questions about how the test should operate.

The following sequence shows the questions which are asked:

```
*** CAUTION ***
Data on the disk partition you specify will be corrupted
Please specify device name:
```

• The user must enter the name of the device to test (for example, sd.sr(,,1)). If the user enters "quit" the test will be halted, otherwise the test will attempt to open the specified device. If the open fails, the following message will be printed and the user will be prompted to specify another device name.

Device sd.si(,,1) cannot be opened

• If the open is successfull, but the specified devicename has a zero block size, the following message will be printed and the user will be prompted to specify another device name.

```
Cannot use disk partition sd.si(,,1) for testing sd.si(,,1) has zero size
```

• If the user entered a valid device name with a non-zero size, then the test continues asking the following questions. Default answers are always displayed. The user accepts the default answer by simply entering a return. If the user enters an invalid response, then an error message is printed and the user is re-prompted for the information.

```
Enter maximum disk block (default is block 66149):
Enter minimum disk block (default is block 0):
Enter Read/Verify retry count (default 0):
Random patterns (default yes):
```

• If the user elects to not use test-generated random patterns, then he/she is asked to enter the data patterns to be used as follows:

```
Enter length of byte pattern (1-80):
Byte 0 of pattern:
Byte 1 of pattern:
Byte 2 of pattern:
Byte 3 of pattern:
```

• If random patterns are selected, then the user is asked if the entire buffer should be random or if the same random byte should be used to fill the entire buffer:

```
Randomize entire buffer? (default no):
```

• Once the test has the data pattern information, the user is asked if verbose mode should be used. In verbose mode more information is printed during the testing phase.

```
Verbose? (default no):
```

• The user is then asked to select the area of physical memory to be used to contain the write and read buffers:

```
Start of buffer area (default is 0x000e0000):
End of buffer area (default is 0x00fffff):
```

- If illegal addresses are entered to the above questions, an error message is printed and the user is asked to select the buffer area again.
- The test then prompts for two patterns to be ORed into the write and read buffer addresses used by the test (for example 0x1f used as an OR pattern will always result in addresses with the least significant 5 bits equal to 0x1f).

```
Enter OR pattern for write buffer address (default is 0):
Enter OR pattern for read buffer address (default is 0):
```

• The number of repetitions controls how many iterations of the test are performed. If zero is selected, the test will run forever or until an error occurs.

Enter number of repetitions (0 means forever, default 1):

After all the above information has been entered, the test selects a target disk block to write at random between minimum block and maximum block, a transfer size (in bytes) at random between 1 byte and the maximum transfer size implied by the minimum and maximum disk blocks, and write and read data buffers at random within the start and end of the specified buffer area. The transfer size is always selected to be within the boundarys of minimimum block and maximum block. The read and write buffers are never allowed to overlap.

When the disk test begins, the following message is printed:

Testing sd.si(,,1):

During the test, one of the following error messages could be displayed depending on the type of error (write, read, compare):

```
Write Error: device sd.si(,,1), fd = 3, count = 0x0
target block = 36504
block address = 0x011d3000
transfer size = 0 \times 00003686
write buffer = 0xffc04840 - 0xffc07ec8
Read Error: device sd.si(,,1), fd = 3, count = 0x0
target block = 36504
block address = 0 \times 011d3000
transfer size = 0 \times 00003686
read buffer = 0xffa3a6fe - 0xffa3dd86
Data Compare Error:
                      device sd.si(,,1), fd = 3
target block = 36504
block address = 0 \times 011d3000
transfer size = 0 \times 00003686
write buffer = 0xffc04840 - 0xffc07ec8
read buffer = 0xffa3a6fe - 0xffa3dd86
error occurred at buffer offset 0x0
  expected: (0xffc07ec8) = 0xd9
  actual: (0xffa3dd86) = 0x00
Run of 1 incorrect byte(s) (end of buffer reached)
1 byte(s) of zero pattern located
```

When the test has completed, the following status message containing the number of errors encountered is displayed:

Done: 0 Read errors, 0 Write errors, 0 Verify errors

7.8 Test 97 - Tape Write/Read Test (must be prompted)

The Tape write/read test verifies the ability of the Solbourne workstation to write data to a tape and read it back. The test is device independent in that it will run on any tape device supported by the standalone driver.

The is test is interactive and must be prompted in order to run. When the test is not prompted, the following message is printed:

Test 31 skipped - use PROMPT command to enable

If the test is prompted, the user is prompted for the following information:

Enter tape device name:

The user must enter the name of the device to test (for example, st.si(,4,)). If the user enters "quit" the test will be halted, otherwise the test will attempt to open the specified device. If the open fails, the following message will be printed and the user will be prompted to specify another device name.

Device st.si(,4,) cannot be opened

If the device is successfully opened, the user is asked to insert a scratch tape and enter return:

Please install a scratch tape and hit return when ready to continue

When the user enters return, the test continues asking the following questions. Default answers are always displayed. The user accepts the default answer by simply entering a return. If the user enters an invalid response, then an error message is printed and the user is re-prompted for the information.

The following sequence shows the questions which are asked:

```
Enter maximum block number (default is block 66149):
Enter Read/Verify retry count (default 0):
Random patterns (default yes):
```

Note that block 0 is always used as the minimum block number for this test so that only the maximum block number has an effect on the maximum transfer size. However, the maximum transfer size is limited by the test to less than or equal to 1Mbyte.

If the user elects to not use test-generated random patterns, then he/she is asked to enter the data patterns to be used as follows:

```
Enter length of byte pattern (1-80):
Byte 0 of pattern:
Byte 1 of pattern:
Byte 2 of pattern:
Byte 3 of pattern:
```

If random patterns are selected, then the user is asked if the entire buffer should be random or if the same random byte should be used to fill the entire buffer:

```
Randomize entire buffer? (default no):
```

Once the test has the data pattern information, the user is asked if verbose mode should be used. In verbose mode more information is printed during the testing phase.

Verbose? (default no):

The user is then asked to select the area of physical memory to be used to contain the write and read buffers:

```
Start of buffer area (default is 0x000e0000):
End of buffer area (default is 0x00ffffff):
```

If illegal addresses are entered to the above questions, an error message is printed and the user is asked to select the buffer area again.

The test then prompts for two patterns to be ORed into the write and read buffer addresses used by the test (for example 0x1f used as an OR pattern will always result in addresses with the least significant 5 bits equal to 0x1f).

```
Enter OR pattern for write buffer address (default is 0):
Enter AND pattern for write buffer address (default is 0xffffffff):
Enter OR pattern for read buffer address (default is 0):
Enter AND pattern for read buffer address (default is 0xffffffff):
Enter OR pattern for for transfer length (default is 0):
Enter AND pattern for for transfer length (default is 0xffffffff):
```

The number of repetitions controls how many iterations of the test are performed. If zero is selected, the test will run forever or until an error occurs.

```
Enter number of repetitions (0 means forever, default 1):
```

Once the above information is entered and processed, the test selects a transfer size (in bytes) at random between 1 byte and the maximum transfer size implied by the minimum and maximum block numbers, and write and read data buffers at random within the start and end of the specified buffer area. The transfer size is always selected to be within the boundarys of minimimum block and maximum block. The read and write buffers are never allowed to overlap.

When the test begins, the following message is printed:

Solbourne Confidential Information - Do Not Distribute

Testing st.si(,4,):

The test then begins the following operations:

- Open device for write
- Write random data of random size (up to 1 megabyte)
- Close the device
- Open the device for read
- Read the data back
- Close the device
- Verify the random data

During the test, one of the following error messages could be displayed depending on the type of error (write, read, compare):

```
Device st.si(,4,) cannot be opened for write
Write Error: device st.si(,4,), fd = 3, count = 0x0
target block = 36504
block address = 0x011d3000
transfer size = 0 \times 00003686
write buffer = 0xffc04840 - 0xffc07ec8
Device st.si(,4,) cannot be opened for read
Read Error: device st.si(,4,), fd = 3, count = 0x0
target block = 36504
block address = 0x011d3000
transfer size = 0 \times 00003686
read buffer = 0xffa3a6fe - 0xffa3dd86
Data Compare Error:
                      device st.si(,4,), fd = 3
target block = 36504
block address = 0x011d3000
transfer size = 0 \times 00003686
write buffer = 0xffc04840 - 0xffc07ec8
read buffer = 0xffa3a6fe - 0xffa3dd86
error occurred at buffer offset 0x0
  expected: (0xffc07ec8) = 0xd9
  actual: (0xffa3dd86) = 0x00
Run of 1 incorrect byte(s) (end of buffer reached)
1 byte(s) of zero pattern located
```

When the test has completed, the following status message containing the number of errors encountered is displayed:

Done: 0 Read errors, 0 Write errors, 0 Verify errors

7.9 Test 98 - NMI and Test Switch Test (must be prompted)

This test verifies that the Test and NMI switches on the Solbourne workstation are functional. The is test is interactive and must be prompted in order to run. When the test is not prompted, the following message is printed:

```
Test 32 skipped - use PROMPT command to enable
```

If the test is prompted, the following messages are printed:

Toggle the NMI and Test	switches (enter	^C when done)
Test Switch	NMI Counter	
NORMAL	0	

The above messages show the state of the Test switch when the test was invoked, and the number of times the NMI button has been pressed. When the user moves the Test switch to the DIAG position, the message changes to:

Test Switch	NMI Counter
DIAG	0

Each time the user presses the NMI switch, the NMI counter value will be incremented by 1.

Test Switch	NMI Counter	
DIAG	1	

The user should always place the test switch back into the DIAG position before halting the test by entering °C. If the switch is left in the DIAG position, the following warning messge is printed:

WARNING: test switch left in "diag" position

Section 8: Commands

8.1 Introduction

This section offers printed copies of man pages for all commands associated with dg(1). The commands are presented in the UNIX man page reference format.

A summary of command usage is displayed on-line when **dg** is running by typing:

DG> ?.

The following is a listing of the **dg** commands available in this section:

SECTION 1: COMMANDS

between (1) cd (1) config(1) continue(1) deposit(1) **dg**(1) errlim(1) errors(1) examine(1) **fbcon**fig(1) fbuf(1) help(1)limit(1) loop(1)**ls** (1) menu(1) names(1) next(1) passes (1) passlim(1) prompt(1) quiet(1)quit(1)restart (1) run(1)screenload(1) status (1) tests (1) time(1) vmeconf(1) what(1) xbuf(1)

between - Set or display between count

SYNOPSIS

between [count]

DESCRIPTION

between sets or displays the current setting of the between count. **between** suppresses printing test completed messages to the screen until *count* passes have completed.

When the status (1) reset command is used, the between count is reset to 1.

OPTION

count Specifies the number of test passes that must be completed before a completion message is displayed. By default the **between** *count* is always set to 1.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example illustrates how to set and redisplay the between count.

```
DG> between 4
DG> between
Between count = 4
DG>
```

SEE ALSO

dg (1), passlim (1), status (1)

cd - change to a different test directory

SYNOPSIS

cd [test directory]

DESCRIPTION

cd changes to a different dg(1) test directory, performing functionality similar to that done by the UNIX cd (1) command.

EXAMPLES

User input in the examples is shown in **boldface** type.

In the following sequence of commands, the user moves down the tree structure of the dg test suite using the cd command.

```
DG> menu

Menu of installed test programs (==> denotes menu):

==> Memory Tests

==> IO Tests

==> Graphics Tests

==> Miscellaneous Tests

DG> cd graphics

DG> ls

Menu of : /Graphics Tests

==> CG40 Tests

==> CG30 Tests

DG> cd CG30

DG>
```

SEE ALSO

dg (1), **l**s (1)

config - Display memory configuration file

SYNOPSIS

config

DESCRIPTION

config displays the memory configuration. When **dg** is invoked it creates a memory configuration table based on the memory configuration information saved in the diagnostic RAM during the power-up self-tests.

EXAMPLES

User input in the examples is shown in **boldface** type.

In the following example, config is entered at the dg prompt. The current contents of the configuration table are displayed.

DG> **config** Memory Configuration:

5 board(s)	totaling 80 Mbytes	
Slot 1	16 Mbytes	Base address = 00000000
Slot 2	16 Mbytes	Base address = 01000000
Slot 3	16 Mbytes	Base address = 02000000
Slot 4	16 Mbytes	Base address = 03000000
Slot 5	16 Mbytes	Base address = 04000000

DG>

SEE ALSO dg (1)

continue - Set or display continue on error flag

SYNOPSIS

continue [on | off]

DESCRIPTION

continue sets or displays the continue-on-error flag. If no parameters are specified, continue displays the current setting of the continue-on-error flag.

The continue flag commands tests to continue executing after a test failure occurs. Tests are designed to check the continue flag to determine if test execution should be halted (the default condition) or if the next test case should be executed.

OPTIONS

- on Turns on the continue-on-error flag.
- off Turns off the continue-on-error flag.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example causes the current error message enable flag to be displayed.

```
DG> continue
continue = off
DG>
```

The following example illustrates how the continue flag is changed and redisplayed.

```
DG> continue on
DG> continue
continue = on
DG>
```

SEE ALSO

dg(1), status(1)

deposit - Deposit data at specified address

SYNOPSIS

deposit [-b | h | w] [addr_range] = value

DESCRIPTION

deposit writes data to an address or range of addresses.

OPTIONS

[-b | h | w]

Specifies the width of the data to be examined.
-b - byte (8 bits)
-h - half word (16 bits)
-w - word (32 bits)
If the width is not specified, a width of -b (1 byte) is assumed.

addr_range

One of the following forms: addr - the location addr addr #count - count locations starting from addr addr1 :addr2 - all locations from addr1 to addr2.

=value Value to be written to the specified address.

EXAMPLES

The following example writes 32 bits of data (zero) to address ff000000 hex.

DG> deposit -w 0xff000000=0 DG>

SEE ALSO

dg(1), **examine**(1)

dg - description of the standalone diagnostic test controller

SYNOPSIS

dg

DESCRIPTION

dg is a standalone/test controller. The test controller provides the commands necessary to randomly select and execute any or all of the available test programs. The operator has control over test execution and can command test programs to loop on error or repeat execution indefinitely.

The following is a list of the dg commands with the shortest possible abbreviation in capital letters. Command names and abbreviations are not case sensitive.

The acceptable commands follow (bold, uppercase letters represent the abbreviated usage of the command name):

? Print summary of dg commands between Set or display between count Change to a different test directory cd Display memory configuration config continue Set or display continue on error flag deposit Deposit data at specified address errlim Set or display error limit errors Display error count examine Examine contents of memory fbconfig Displays the frame buffer configuration fbuf Fill internal command buffer help Print summary of dg commands limit Display or set memory test limits loop Set or display loop on test flag List contents of test directory ls Display listing of the available tests menu Enable or disable printing of test names during test exenames cution Execute next selected test next passes Display pass count passlim Set or display pass limit Set or display prompt flags prompt Set or display error message enable flag quiet Exit from dg program quit

restart	Restart execution of selected tests
run	Start execution of selected tests
screenlo	bad
	Loads a raster image file into the specified frame buffer
status	Display or reset state of modes, flags and counts
tests	Select or display tests to be executed
time	Set or display print time flag and display current date and time
vmeconf	Configure VMEbus devices
what	Display information about Kbus boards installed in system

xbuf Display, save, or execute the contents of the command buffer

DG(1)

errlim - Set or display error limit

SYNOPSIS

errlim [limit]

DESCRIPTION

limit

errlim sets or displays the current setting of the test error limit.

OPTION

Specifies the number of test errors that can occur before test execution is halted. By default, the *limit* is set to zero (no error limit). However, the error limit may be changed by specifying a new limit value. The limit value must be entered in unsigned decimal format and be between 0 and 2,147,483,647, inclusive.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example illustrates how to display the current error limit.

```
DG> errlim
Error limit = 0
DG>
```

The following example illustrates how to change and re-display the error limit.

```
DG> errlim 100
DG> errlim 
Error limit = 100
DG>
```

SEE ALSO

dg (1), errors (1), status (1)

errors - Display error count

SYNOPSIS

errors

DESCRIPTION

errors displays the number of test errors that have occurred since the last run(1) command.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example illustrates how to display the error count.

DG> **errors** Total test errors = 0 DG>

SEE ALSO

dg(1), errlim(1), status(1)

examine - Examine contents of memory

SYNOPSIS

examine [-b | h | w] [addr_range]

DESCRIPTION

examine reads data from the specified address or addresses.

OPTIONS

[-b | h | w] Specifies the width of the data to be examined. -b - byte (8 bits) -h - half word (16 bits) -w - word (32 bits)

addr_range

One of the following forms: addr - the location addr addr #count - count locations starting from addr addr1 : addr2 - all locations from addr1 to addr2. If range is not specified, the address range used on the previous examine command is used.

EXAMPLES

The following example shows how to examine a byte from location 17000000 hex.

```
DG> examine -b 0x17000000
(0x17000000): 0x3d
DG>
```

SEE ALSO

dg(1), **deposit**(1)

fbconfig - displays the frame buffer configuration file

SYNOPSIS

fbconfig

DESCRIPTION

The frame buffer configuration is read from the diagnostic RAM when dg is invoked. The board must be configured in descending slot order.

EXAMPLE

User input in the example is shown in **boldface** type.

DG> fbconfig

Frame Buffer Configuration:

1 graphics board(s): Slot IO address Board Type Resolution 1 8100000 monochrome low Slot number of default board to test: 1 DG>

SEE ALSO dg (1)

fbuf - fill internal command buffer

SYNOPSIS

fbuf

DESCRIPTION

fbuf fills the internal command buffer. The internal command buffer can be displayed, executed, or initialized from a file using the xbuf (1) command. When fbuf is entered, the fbuf: prompt is displayed and each line that is typed is saved into the command buffer. When finished entering commands, the fbuf command is entered again to close the command buffer and return the the DG> prompt.

EXAMPLE

User input in the example is shown in **boldface** type.

In the following example, fbuf opens the command buffer. When opened, the fbuf: prompt is displayed. The command buffer is filled first with a tests command to select tests 1 through 6, then with a **run** command. Finally, the command buffer is closed by reentering the fbuf command and the DG> prompt returns.

```
DG> fbuf
fbuf: tests 1 2 3 5 6
fbuf: run
fbuf: fbuf
DG>
```

SEE ALSO

dg(1), xbuf(1)

help - Display command list or information on a specific command

SYNOPSIS

help [command ...]

DESCRIPTION

The help command with no arguments causes a list of command and command usages to be displayed. This is equivalent to the ? command.

The help command with an argument causes the command usage for the specified command to be displayed.

OPTIONS

command

name of command for which help is desired.

EXAMPLE

The following example causes the command usage for the tests command to be displayed:

```
DG> help tests
```

```
Usage: tests [ all | [ test ... ] [ test:test ... ] [
menu_path ] [ * ] ]
```

```
DG>
```

SEE ALSO

dg(1)

limit - Display or set memory test limits

SYNOPSIS

limit[reset | memname[low high | reset]]

DESCRIPTION

limit displays or sets the memory test limits of the system. By default, limit displays all the memory limits.

limit is set to the amount of installed memory for each memory devices in the system. Memory devices include physical memory, VMEbus address map memory, and VMEbus resident memory boards.

The test programs examine the memory limits to determine how much memory to test.

OPTION

reset Resets the limits back to the default settings. The default settings are determined by the amount of installed memory. For example, as set by the config (1) command.

low high

low is the first address and high is the last address to test, inclusive.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example displays the current limit settings for all the memory devices.

DG> limit		
Memory limits:	LOW	HIGH
mem =	5ffff	8fff
vmemap =	40	7ff
vmemem =	3	5ff
DG>		

The following example resets the memory limits to their default values.

DG> limit reset DG> limit		
Memory limits:	LOW	HIGH
mem =	40000	lfffff
vmemap =	20	7ff
vmemem =	0	7ffff
DG>		

The following example sets the memory limits for physical memory to the range 40000 through 2ffffff hex and set the VMEbus address map limits to 20 through ff hex, inclusive.

DG> limit mem 40000 2ffffff			
DG> limit	vmemap	20 ff	
DG> limit			
Memory lim	its:	LOW	HIGH
mem	=	40000	2ffff
vmema	p =	20	ff
vmeme	m =	0	7fff
DG>			

The following example resets only the VMEbus address map limits to their default values. The physical memory values are not modified.

DG> limit vmemap	reset	
DG> limit		
Memory limits:	LOW	HIGH
mem =	40000	2fffff
vmemap =	20	7ff
vmemem =	3	5ff
DG>		

DG>

SEE ALSO

config(1), dg(1)

loop - Set or display loop on test flag

SYNOPSIS

loop [on | off]

DESCRIPTION

loop sets or displays the loop on error flag. If no parameters are specified, **loop** displays the current setting of the loop flag.

The loop flag commands tests to loop on the failing test case in the event a test error occurs. Tests are designed to halt when errors occur so that the loop command may be entered.

OPTIONS

on Turns on the loop flag.

off Turns off the loop flag.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example causes the current loop flag do be displayed.

DG> loop loop = off DG>

The following example illustrates how the loop flag is changed and re-displayed.

```
DG> loop on
DG> loop
loop = on
DG>
```

SEE ALSO

dg (1), status (1)

ls - list contents of test menu

SYNOPSIS

ls

DESCRIPTION

Is lists the contests of a test menu, performing functionality similar to that done by the UNIX Is (1v) command. The information displayed is in a different format that the UNIX command of the same name.

EXAMPLE

User input in the examples is shown in **boldface** type.

The following example illustrates how the user lists the contents of the current menu using the ls command. The current menu is displayed as the Graphics Tests and the submenus of CG40 Tests and CG30 Tests are shown. Using the ***** as an argument to the tests command, all the tests in both submenus are selected for execution.

```
DG> 1s
Menu of : /Graphics Tests
  ==>
        CG40 Tests
        CG30 Tests
  ==>
DG> tests *
DG> tests
DG> tests
selected tests: 40
                      41
                           42
                                 43
                                           45
                                                     47
                                      44
                                                46
                 48
                      49
                           50
                                 51
                                      52
                                           53
                                                54
                                                     55
                 56
                      57
                           58
                               59
                                      60
                                                62
                                                     63
                                           61
                 64
                      65
                           66
                                 67
                                      68
                                           69
                                                70
                                                     71
```

DG>

SEE ALSO

cd (1), dg (1), run (1), tests (1)

menu - Display listing of available tests

SYNOPSIS

menu [menu_name | all]

DESCRIPTION

menu lists the names of all available tests in the default order of execution. menu displays tests in the default order of execution.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example displays a listing of the top-level test directories:

```
DG> menu
Menu of installed test programs (==> denotes menu):
==> Memory Tests
==> IO Tests
==> Graphics Tests
==> Miscellaneous Tests
DG>
```

The following example displays a complete listing of all directories and the tests that reside in the directories:

```
DG> menu all
Menu of installed test programs (==> denotes menu):
 ==>
      Memory Tests
      1: Cache block virtual alias test (affected by prompt)
      2: Memory Data RAM test (affected by prompt)
      3: Memory ECC RAM test (affected by prompt)
      IO Tests
  ==>
    ==>
        ASIC Tests
        4: I/O ASIC register access test
        5: I/O ASIC reset tests
        6: I/O ASIC FIFO/ECC test
      ==>
          LANCE (7990) Tests
          7: 7990 LANCE initialization test
          8: 7990 LANCE internal loopback test
          9: 7990 LANCE external loopback test (must be prompted)
         10: 7990 LANCE data alignment test
         11: 7990 LANCE cacheable data merge test
        12: 7990 LANCE address/cache data test (affected by prompt)
         13: 7990 LANCE cache block buswatcher test
      ==> SCSI (33C93) Tests
         14: 33C93 SBIC (SCSI) data path (Write Buffer) test
        15: SCSI cacheable block data merge test
        16: SCSI Multi-bit ECC error test
        17: SCSI page overflow bit test
        18: SCSI address counter/cache data test
        19: SCSI cache block buswatcher test
        20: SCSI cache block flush test
```

21: SCSI/LANCE data transfer test ==> VMEbus interface Tests 22: VMEbus address map RAM test 23: VMEbus data path test 24: VMEbus address path test 25: VMEbus data multiplexing test 26: VMEbus Interrupt (IACK) test 27: RF3500 SCSI data path (Write Buffer) test 28: VMEbus cacheable data merging test 29: VMEbus block mode data merging test 30: VMEbus cacheable data buffer test 31: VMEbus ping-pong data buffers test 32: VMEbus block mode address counter test 33: VMEbus panic interrupt test BW20 Tests ==> 34: Frame Buffer RAM Test (affected by prompt) 35: Frame Buffer interrupt and Interrupt Registers Tests ==> Serial Ports Tests 36: Serial ports reset test 37: Serial ports internal loopback test 38: System Timer test 39: Profile Timer test 40: RTC-58321 real time clock test ==> Graphics Tests ==> CG40 Tests ==> Bt458 Tests 41: Registers Test 42: Address Register (Autoincrement) Test 43: Main CLUT (Colormap) Test 44: Overlay CLUT (Colormap) Test 45: Frame Buffer RAM Test (affected by prompt) 46: Frame Buffer Interrupt and Interrupt Registers Tests ==> CG30 Tests ==> Registers Tests 47: IO location/control status registers test 48: IO location (space bits) register test 49: 8-bit registers test 50: 16-bit registers test 51: Dummy registers access test ==> Bt458 Tests 52: Registers test 53: Address register (autoincrement) test 54: Main CLUT (Colormap) test 55: Overlay CLUT (Colormap) test 56: Colormap pattern test (must be prompted) 57: Frame buffer to colormap test 58: Colorbar test pattern (must be prompted) ==> Bt431 Tests 59: Address register (autoincrement) test 60: Single cursor control registers test 61: Both cursors to single cursor control registers test 62: Single cursor pattern RAM test 63: Both cursors to single cursor pattern RAM test

Solbourne Computer, Inc.

10 May 1989
MENU(1)

```
Shadow RAM Tests
   ==>
      64: Shadow RAM test
      65: Shadow RAM to Bt458 colormap update test
   ==> Frame Buffer RAM Tests
      66: Plane major mode test (affected by prompt)
      67: Pixel major mode test (affected by prompt)
      68: Pixel major to plane major test
   ==> ROP Tests
     ==> Registers Tests
        69: Single plane registers test
        70: All planes to single plane registers test
        71: BTLA mode single plane registers test
        72: BTLA mode all planes to single plane registers test
     ==> Mode Tests
        73: Mode-0 read access test
        74: Mode-1 read access test
        75: Mode-4 read access test
        76: Mode-5 read access test
        77: Mode-6 read access test
        78: Mode-0 write access test
        79: Mode-1 write access test
        80: Mode-2 write access test
        81: Mode-3 write access test
        82: Mode-4 write access test
        83: Mode-5 write access test
        84: Mode-6 write access test
        85: Mode-7 write access test
        Retrace Interrupts Tests
   ==>
      86: Vertical retrace interrupts test
      87: Framecount register (counter) test
      88: Retrace/SCC interrupt combination test
   ==> Serial Ports Tests
      89: Serial ports reset test
      90: Serial ports loopback test
    Miscellaneous Tests
==>
   91: Random ECC generation test (must be prompted)
   92: Cache data bus test
   93: RS232 connector loopback test (must be prompted)
   94: IDPROM checksum tests (affected by prompt)
   95: Ethernet tftp read test
   96: Disk write/read test (affected by prompt)
   97: Tape write/read test (must be prompted)
   98: NMI and Test switch test (must be prompted)
```

DG>

EXAMPLE

The following example shows the listing received when the user requests the menu of the memory tests:

DG> menu Memory menu of: /Memory tests

1: Cache block virtual alias test (affected by prompt) 2: Memory Data RAM test (affected by prompt) 3: Memory ECC RAM test (affected by prompt)

DG>

SEE ALSO dg (1), tests (1)

names - Enable or disable printing of test names during test execution

SYNOPSIS

names [on | off]

DESCRIPTION

names enables or disables the printing of test names during test execution.

OPTIONS

- on Enables the printing of the test names during test execution. This is the default setting.
- off Disables the printing of the test names during test execution.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example causes the state of the name flag to be displayed.

DG> **names** names = on DG>

The following example illustrates how the names flag is changed and redisplayed.

```
DG> names off
DG> names
names = off
DG>
```

SEE ALSO

dg (1), status (1)

next - Execute next selected test

SYNOPSIS

next

DESCRIPTION

next causes the test sequence to be continued, starting with the next selected test. It is used when a test halts on an error and the user wishes to continue test execution with the next test in the sequence.

EXAMPLE

User input in the example is shown in **boldface** type.

In the following example run was entered to begin test execution. The current test selection was executed until an error was encountered in test 3. next was entered to continue the test sequence starting with the next test in the sequence.

```
DG> run

Starting Test 1: (testname)

Starting Test 2: (testname)

Starting Test 3: (testname)

Test 3 error: (error message)

DG> next

Starting Test 4: (testname)

Starting Test 5: (testname)

Starting Test 6: (testname)

.

.

Starting Test n: (testname)

Tests completed: Passes = 1 Errors = 1 Wed May 10 15:41:29 1989

DG>
```

SEE ALSO

between (1), errlim (1), dg (1), passlim (1), restart (1), run (1)

passes - Display pass count

SYNOPSIS

passes

DESCRIPTION

passes displays the number of complete test passes that have made since the last run command.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example illustrates how to use the **passes** command.

```
DG> passes
Total passes = 0
DG>
```

SEE ALSO

dg(1), **p**asslim(1)

passlim - Set or display pass limit

SYNOPSIS

passlim [limit]

DESCRIPTION

passlim sets or displays the current setting of the test pass *limit*. **passlim** specifies the number of test passes that can occur before test execution is halted.

This command should be used when it is desired to execute numerous passes of the test sequence.

OPTION

limit

Sets the number of test passes that will be run. By default, *limit* is set to one. *Limit* must be entered in unsigned decimal format in the range 0-to-2,147,483,647, inclusive. A *limit* of 0 specifies that tests execute continuously until a Control-C is entered.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example illustrates how to display the current pass limit.

```
DG> passlim
Pass limit = 1
DG>
```

The following example illustrates how to change and re-display the pass limit.

```
DG> passlim 0
DG> passlim
Pass limit = 0
DG>
```

SEE ALSO

dg(1), passes(1)

prompt - Set or display prompt flags

SYNOPSIS

prompt[all | off | test test ... | test :test ...]

DESCRIPTION

prompt sets or displays the prompt flag for each test program. The command allows the user to selectively alter the default behavior of the test programs by turning the flag for the specified tests on or off.

Only a few of the **dg** tests use the **prompt** flag. The behavior of the test depends on what the test is attempting to accomplish. In some case, if a test isn't prompted it does not execute. In others, it modifies the test algorithm.

Single tests or a range of tests may be prompted by specifying the test numbers or range of tests number.

The menu (1) command indicates which tests examine their prompt flags.

OPTIONS

- all Set prompt flags for all tests. all can be specified at any time to prompt all tests.
- off Turns prompt flags for all tests off. off can be specified at any time to turn off prompts for all tests.
- test Prompt specified test. If test is not specified, the prompt command displays the current status of the prompt flags.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example illustrates how to display the prompt flags.

DG> prompt no prompted tests DG> prompt 1:26 DG> prompt							
prompted tests:	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		
DG> prompt off DG> prompt 8 9 10 DG> prompt							
prompted tests:	8	9	10				

SEE ALSO

dg(1), **menu**(1), **tests**(1)

quiet - Set or display error message enable flag

SYNOPSIS

quiet[on | off]

DESCRIPTION

quiet sets or displays the error message enable flag. If no parameters are specified, quiet displays the current setting of the flag.

The error message enable flag prevents error messages from being displayed on test failures. This feature should be used to create the tightest possible loop when the loop flag is on. A Control-C must be entered to stop the loop and return to the DG> prompt.

OPTIONS

on Turns on the quiet flag.

off Turns off the quiet flag.

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example causes the current error message enable flag to be displayed.

DG> **quiet** quiet = off DG>

The following example illustrates how the quiet flag is changed and redisplayed.

```
DG> quiet on
DG> quiet
quiet = on
DG>
```

SEE ALSO

dg(1), status(1)

quit - Exit from dg program

SYNOPSIS

quit

DESCRIPTION

quit exits from the dg program and returns the user to the ROM> prompt.

SEE ALSO

dg(1)

restart - Restart execution of selected tests

SYNOPSIS

restart

DESCRIPTION

restart causes the current **test**(1) selection to be executed beginning with the first test current test selection. The major difference between **restart** and **run**(1) is that **restart** goes back to the first test in the sequence, while **run** continues execution with the next selected test.

The number of times the test selection is executed depends on the value of the **passlim** (1) *limit*.

EXAMPLE

User input in the example is shown in **boldface** type.

In the following example **run** was entered to begin test execution. The current test selection were executed until an error was encountered in test 3. **restart** was entered to start the test sequence again from the beginning.

```
DG> run

Starting Test 1: (testname)

Starting Test 2: (testname)

Starting Test 3: (testname)

Test 3 error: (error message)

DG> restart

Starting Test 1: (testname)

Starting Test 2: (testname)

Starting Test 3: (testname)

.

Starting Test n: (testname)

Tests completed: Passes = 1 Errors = 0 \

Wed May 10 16:30:33 1989

DG>
```

SEE ALSO

dg(1), next(1), passlim(1), run(1)

run - Start execution of selected tests

SYNOPSIS

run

DESCRIPTION

run causes the current test (1) selection to be executed. The number of times the test selection is executed depends on the value of the passlim (1) *limit*.

EXAMPLE

User input in the example is shown in **boldface** type.

In the following example **run** was entered to begin test execution. The current test selection was executed once (passlim = 1) followed by a tests completed message. If **passlim**'s *limit* is set to a value other than one, the complete test sequence would be repeatedly executed until *limit* is reached, at which time the program would return to the DG> prompt. The test completed message is displayed after each pass.

```
DG> run

Starting Test 1: (testname)

Starting Test 2: (testname)

Starting Test 3: (testname)

.

.

Starting Test n: (testname)

Tests completed: Passes = 1/ Errors = 0 \

Wed May 10 16:30:33 1989

DG>
```

SEE ALSO

dg (1), next (1), passlim (1), restart (1)

screenload - Loads a raster image file into the specified frame buffer

SYNOPSIS

screenload [slot] < rasterfile</pre>

DESCRIPTION

screenload is used to load raster image file into the specified frame buffer.

If the optional slot number is ommitted on the command line, the first graphics board identified in the frame buffer configuration table is used to load the rasterfile.

If the rasterfile does not exist or is the wrong file type, a message indicating the error is printed.

EXAMPLE

User input in the example is shown in **boldface** type.

DG> screenload 1 <rasterfile

The above command will load the specified rasterfile into the frame buffer in slot 1. If there is no frame buffer identified in the frame buffer configuration table (see (**fbconfig**(1)) for slot 1, an error message indicating that no graphics board is configured in slot 1 is printed and the command is ignored.

SEE ALSO

dg(1), fbconfig(1)

status - Display or reset state of modes, flags, and counts

SYNOPSIS

status [reset] [flags]

DESCRIPTION

status displays the current state of all modes, program flags, and counters. **flags** resets all the flags, which includes names, continue, loop, quiet, and xbuf.

OPTION

reset Resets the status of flags, counts, and limits to the default setting. reset also resets the test selection back to default values.

flags Resets the status of flags to the default settings.

EXAMPLE

User input in the example is shown in **boldface** type.

```
DG> status
Wed May 10 12:45:20 1989
Names = on
Continue = off
Loop = off
Quiet = off
Time = off
XBUF echo = off
Pass count = 0
Pass limit = 1
Between count = 1
Error count = 0
Error limit = 0
DG>
```

SEE ALSO

between (1), continue (1), dg (1), ecc (1), errlim (1), errors (1), loop (1), names (1), passes (1), passlim (1), quiet (1), time (1), xbuf (1)

tests - Select or display tests to be executed

SYNOPSIS

tests [all | [test ...] [test :test ...] [menu_path] [*]]

DESCRIPTION

tests select the tests for execution by the run(1) command. By default, all tests are selected for execution when the program is initialized.

Single tests or a range of tests may be selected by specifying the test numbers or range of tests number.

OPTIONS

all Execute all the tests. all can be specified at any time to reselect all tests.

menu_path

Executes the specified group (menu) of tests. *menu_path* can be specified from the top of the menu tree or from the current working directory by entering an asterisk (*).

test Select specified *test*. If *test* is not specified, the **tests** command displays the current test selection for execution.

test :test Select specified test range for execution

EXAMPLES

User input in the examples is shown in **boldface** type.

The following example illustrates how to display the test selection.

DG> tests								
selected tests:	1	2	3	4	5	6	7	8
	9	10	11	12				

DG>

In the following example, every test in the dg test suite is selected for execution using the all option.

DG> tests all								
DG> tests								
selected tests:	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	50	51	52	53	54	55	56
	57	58	59	60	61	62	63	64
	65	66	67	68	69	70	71	72
	73	74	75	76	77	78	79	80
	81	82	83	84	85	86	87	88
	89	90	91	92	93	94	95	96
	97	98						

Solbourne Computer, Inc.

TESTS(1)

In the following sequence of commands, the user is first uses the menu (1) command to display the test directories in the current working directory. The user then uses the cd (1) command to move to the Graphics Tests directory. Using the ls (1) command, the user displays the test directories located in the Graphics Test directory, then selects all the tests in the CG40 directory by simply giving the menu path.

```
DG> menu
Menu of installed test programs (==> denotes menu):
  ==>
        Memory Tests
        IO Tests
  ==>
  ==>
        Graphics Tests
  ==>
        Miscellaneous Tests
DG> cd graphics
DG> 1s
Menu of : /Graphics Tests
  ==>
        CG40 Tests
  ==>
        CG30 Tests
DG> tests CG40
DG> tests
selected tests: 40
                      41
                           42
                                43
                                     44
                                           45
DG>
```

The following example illustrates how the asterik (*) argument to the tests command is used to select tests. First the user lists the contents of the current directory using the ls command. The current directory is displayed as the Graphics Tests and the subdirectories of CG40 Tests and CG30 Tests are shown. Using the * as an argument to the tests command, all the tests in both directories are selected for execution.

```
DG> 1s
Menu of : /Graphics Tests
        CG40 Tests
  ==>
  ==>
        CG30 Tests
DG> tests *
DG> tests
DG> tests
selected tests: 40
                            42
                                  43
                                            45
                                                  46
                                                       47
                     41
                                       44
                  48
                       49
                                                       55
                            50
                                  51
                                       52
                                            53
                                                  54
                  56
                       57
                            58
                                  59
                                       60
                                            61
                                                  62
                                                       63
                  64
                       65
                            66
                                  67
                                       68
                                            69
                                                 70
                                                       71
DG>
```

In the following example, tests 8 through 1 are selected and displayed. Note that tests

may be selected to run in any order.

```
DG> tests 8:1
DG> tests
selected tests: 8 7 6 5 4 3 2 1
DG>
```

In the following example, all installed tests are selected and displayed.

```
DG> tests all
DG> tests
selected tests: 1 2 3 4 5 6 7 8
9 10 11 12
DG>
```

** • • • •

SEE ALSO cd (1), dg (1), ls (1), next (1), restart (1), run (1)

time - Set or display print time flag

SYNOPSIS

time [on | off]

DESCRIPTION

time sets or displays the print-time flag. If no parameters are specified, time displays the current setting of the print-time flag and the current time and data. The print-time flag controls whether the current time and date is printed when test names are displayed during test execution. The default state of the print-time flag is off (no time printed). If both the names flag and print-time flag are on, the time and date is printed on the line following the test name during test execution.

OPTIONS

- on Turns on the print-time flag.
- off Turns off the print-time flag.

EXAMPLES

The following example causes the current print-time flag to be displayed:

```
DG> time
time = off
Wed May 10 14:20:00 1989
DG>
```

The following example illustrates how the print-time flag is changed and redisplayed.

```
DG> time on
DG> time
time = on
Wed May 10 14:20:00 1989
DG>
```

SEE ALSO

names (1), **dg** (1), **status** (1)

vmeconf - Configure VMEbus devices

SYNOPSIS

vmeconf

DESCRIPTION

vmeconf generates or displays the VMEbus configuration table.

When dg is invoked, it does not asks the user to generate a VMEbus configuration table. Therefore, if the user wishes to perform tests of the VMEbus chassis, they must first execute this command.

vmeconf prompts for all user input. It accepts no options or arguments at the command line.

Currently, vmeconf supports the Ciprico Rimfire, Excelan Ethernet, and Plessy RAM boards.

EXAMPLE

User input in the example is shown in **boldface** type.

The following example shows how **vmeconf** is used to remove an Excelan Ethernet VMEbus board from the configuration, then how the program would be used to put the board back into the configuration table.

DG> vmeconf

VMEbus Configuration consists of four boards
 (0) Ciprico Rimfire 3500 VMEbus-to-SCSI
 Am = 0x2d Addr = 0x5000 Physaddr = 0x85ff5000
 (1) Excelan Ethernet
 Am = 0x3d Addr = 0xd00000 Physaddr = 0x87d00000
 (2) Plessey RAM (512K)
 Am = 0x3d Addr = 0x100000 Physaddr = 0x87100000
Do you wish to change this configuration? (y/n) y
Do you want the default configuration? (y/n) n
Do you want to delete any entries? (y/n) y
Entry number to delete (q to quit)? 1
Entry number to add any entries? (y/n) n

(0) Ciprico Rimfire 3500 VMEbus-to-SCSI Am = 0x2d Addr = 0x5000 Physaddr = 0x85ff5000 (2) Plessey RAM (512K) Am = 0x3d Addr = 0x100000 Physaddr = 0x87100000

Do you wish to change this configuration? (y/n) **y** Do you want the default configuration? (y/n) **n** Do you want to delete any entries? (y/n) **n** Do you want to add any entries? (y/n) **y** How many vme boards are to be added? (0-5) **1**

Enter information for board 1:

Valid vme board types are: 0: none 1: Ciprico Rimfire 3500 VMEbus-to-SCSI 2: Excelan Ethernet 3: Plessey RAM (512K) Type of board? 2 Valid address modifiers are: 9: extended user data access d: extended supervisor data access 39: standard user data access 3d: standard supervisor data access 29: short user data access 2d: short supervisor data access Address modifier? 3d Address? d00000 VMEbus Configuration consists of 3 boards (0) Ciprico Rimfire 3500 VMEbus-to-SCSI Am = 0x2dAddr = 0x5000Physaddr = 0x85ff5000(1) Excelan Ethernet Am = 0x3d $Addr = 0xd00000 \quad Physaddr = 0x87d00000$ (2) Plessey RAM (512K) Am = 0x3d Addr = 0x100000 Physaddr = 0x87100000 Do you wish to change this configuration? (y/n) n DG>

SEE ALSO dg (1)

what - Display information about Kbus boards installed in system

SYNOPSIS

what

DESCRIPTION

what goes through the ID PROMs of every board installed in the Kbus and displays information about the slot number, board type, revision level, and other information about the board.

EXAMPLE

The following is an example of what output.

DG> what

SLOT	TYPE	REV	INFO
7	S 0	AF	Kbus System Board hostid: 0x0 serial number: 11 ethernet address: 0:0:0:0:0:0 manufacturing date: 880826
7	GO	AA	BW20 Monochrome Frame Buffer
6	MO	AA	Kbus Memory Board 16 Megabytes Base Address 0x0
5	G1	AA	Kbus CG40 Color Graphics Board
4	PO	AD	Kbus Series4 Processor Board
3	MO	AA	Kbus Memory Board 16 Megabytes Base Address 0x0
1 DG>	G2	AA	Kbus CG30 Color Graphics Board

SEE ALSO dg (1)

-

xbuf - Display or execute the contents of the command buffer

SYNOPSIS

xbuf[-list | -echo | -noecho | < filename [count] | count >

DESCRIPTION

xbuf executes commands in internal command buffer.

OPTIONS

-list	Displays	the contents	of the co	mmand buf	fer.
-------	----------	--------------	-----------	-----------	------

-echo Echoes commands to the screen as they are processed.

-noecho Disable commands from being echoed to the screen as they are processed.

filename read commands from specified file.

count Specifies the number of times the command buffer is to be executed. A value of zero causes continuous execution (a Control-C aborts execution).

SEE ALSO

fbuf(1), **dg**(1)

Appendix A: MARCH Algorithm

A.1 Introduction

This appendix explains the way the MARCH pattern algorithm works. The steps follow:

- 1. Every location of the target RAM is written from low address to the high address with the background pattern. This is the first pattern of the initial complement pair (00).
- 2. The low address in the RAM is read and verified to contain the background pattern. If the background pattern is correct, the complement pattern is written to the same address. The location is read again and verified to now contain the complement pattern.
- 3. Step 2 is repeated for all addresses, from low to high. At this point, every location in RAM contains the complement pattern. Once this has completed, the test goes in reverse order from the high address back to the low address. The first read is now verified to contain the complement pattern followed by a write and read of the background pattern.
- 4. This test advances to the next complement pattern pair and repeats the same steps.

-

Appendix B: Moving Inversions Test Algorithm

B.1 Introduction

This appendix contains a general description of the Moving Inversions Algorithm (MOVI) commonly used to verify the addressing and data integrity of RAM devices. The strengths of this algorithm are its relatively short execution time, functional testing of memory bits, and dynamic tests of best and worst case access times.

In principle, MOVI inverts the data of each address sequentially, thus creating an access time by the jump from one address to another which contains different information. To measure access times, the data at each address is read before and after inversion. This requires three operations on each address: a read, a write, and another read.

The read/write/read operations are performed with both forward and backward (reverse) address sequences, and also with n orders of the address-bit significance (where n is the number of address bits).

A general stepwise description of the MOVI algorithm is presented below:

- 1. The target memory is filled with a background data pattern.
- 2. In the forward direction (from low to high addresses) a target location is read and check for the correct background pattern (this is the 1st read of the forward pass).
- 3. The complement pattern (1's complement of the background pattern) is written to the target location.
- 4. The target location is read and checked for the correct complement pattern (this is the 2nd read of the forward pass).
- 5. Steps 2 through 4 are repeated for all addresses until the high address of the memory has been reached. At this point the memory should be filled with the complement pattern (assuming no errors were encountered).
- 6. In the reverse direction (from high to low address) a target location is read and checked for the correct complement pattern (this is the 1st read of the reverse pass).
- 7. The background pattern (1's complement of the complement pattern) is written to the target location.
- 8. The target location is read and checked for the correct background pattern (this is the 2nd read of the reverse pass).
- 9. Steps 6 through 8 are repeated for all addresses until the low address of memory has been reached. At this point the memory should be filled with the background pattern again (assuming no errors were encountered during the reverse pass).

The steps above implement the first iteration of the MOVI test, where the basic address increment value is 1. Successive iterations use higher address-bit significance up to 2**n where n is the number of involved address bits. This is the same as using a different bit of the address each time as the least significant bit for incrementing through all possible addresses. This has the effect of incrementing through all the addresses by 2's, 4's, 8's, and so on; every address overflow generates and end-around carry, so that all addresses are tested once in each sequence.

B-1

The table illustrates the binary address sequences generated by MOVI for and eight location memory:

Forward sequences:

Iteration	0	1	2
	lsb	lsb	lsb
	v	v	v
	000 0	000 0	0000
	001 1	010 2	100 4
	010 2	100 4	
	0113	1106	0011
	100 4		101 5
	101 5	001 1	
	1106	0113	010 2
	1117	101 5	1106
		1117	
			011 3
			1117

Reverse sequences:

Iteration	0	1	2
	lsb	lsb	lsb
	v	v	v
	1117	1117	1117
	1106	101 5	011 3
	101 5	011 3	
	100 4	001 1	1106
	011 3		010 2
	010 2	1106	
	001 1	100 4	101 5
	0000	010 2	001 1
		0000	
			100 4
			0000

and an an an owner of the second

Appendix C: Frame Buffer Test Modes

C.1 Introduction

All of the frame buffer tests may be executed in two modes depending on the state of the prompt flag and the number of frame buffers in the frame buffer configuration table.

If the prompt flag is set and there are multiple frame buffers, all the tests will present a menu of available frame buffers. The user is asked to select the board(s) that are to be tested. The menu will look as follows if there were two boards (slots 5 and 3):

```
Board to test:

1) Board in slot 5

2) Board in slot 3

Which board ?
```

If the prompt flag is not set all boards will be tested.

С

Comamnds: Menu, 2-5 Commands: Entering, 2-2 Listing with 1s, 3-2 Moving about with cd, 3-2 Names and functions, 2-3 Reference pages, 8-1 Rules for use, 2-2 Using, 2-3

D

dg: Commands, 8-1 Description, 1-1 Entering commands, 2-2 Exiting, 2-10 Getting started with, 2-1 Invoking the program, 2-1 Related documentation, 1-1

Ε

Exiting dg, 2-10

F

Frame Buffer Test Modes, C-1

Ι

Invoking the program, 2-1

Μ

MARCH Algorithm, A-1

Menu structure: Overview, 3-1 Menu: all option, 2-5 Command, 2-5 Moving Inversions Algorithm, B-1

Т

Tables initialized, 2-1 Test 1 - Cache Block Virtual Alias Test (affected by prompt), 4-1 Test 10 - 7990 LANCE Data Alignment Test, 5-6 Test 11 - 7990 LANCE Cacheable Data Merge Test, 5-6 Test 12 - 7990 LANCE Address/Cache Data Test (affected by prompt), 5-7 Test 13 - 7990 LANCE Cache Block Buswatcher Test, 5-7 Test 14 - 33C93 SBIC (SCSI) Data Path (Write Buffer) Test, 5-8 Test 15 - SCSI Cacheable Block Data Merge Test. 5-9 Test 16 - SCSI Multi-bit ECC Error Test, 5-10 Test 17 - SCSI Page Overflow Bit Test, 5-11 Test 18 - SCSI Address Counter/Cache Data Test, 5-13 Test 19 - SCSI Cache Block Buswatcher Test, 5-13 Test 2 - Memory Data RAM Test (affected by prompt), 4-1 Test 20 - SCSI Cache Block Flush Test, 5-14 Test 21 - SCSI/LANCE Data Transfer Test, 5-14 Test 22 - VMEbus Address Map RAM Test, 5-15 Test 23 - VMEbus Data Path Test, 5-16 Test 24 - VMEbus Address Path Test, 5-16 Test 25 - VMEbus Data Multiplexing Test, 5-17 Test 26 - VMEbus Interrupt (IACK) Test, 5-18 Test 27 - RF3500 SCSI Data Path (Write Buffer) Test, 5-20

Test 28 - VMEbus Cacheable Data Merging Test, 5-22 Test 29 - VMEbus Block Mode Data Merging Test, 5-22 Test 3 - Memory ECC RAM Test (affected by prompt), 4-2 Test 30 - VMEbus Cacheable Data Buffer Test, 5-23 Test 31 - VMEbus Ping-Pong Data Buffers Test, 5-23 Test 32 - VMEbus Block Mode Address Counter Test, 5-24 Test 33 - VMEbus Panic Interrupt Test, 5-24 Test 34 - Frame Buffer RAM Test (affected by prompt), 5-26 Test 35 - Frame Buffer Interrupt and Interrupt **Registers Test**, 5-27 Test 36 - Serial Ports Reset Test, 5-28 Test 37 - Serial Ports Internal Loopback Test, 5-29 Test 38 - System Timer Test, 5-30 Test 39 - Profile Timer Test, 5-31 Test 4 - I/O ASIC Register Access Test, 5-1 Test 40 - RTC-58321 Real Time Clock Test, 5-31 Test 41 - Bt458 Registers Test, 6-1 42 - Bt458 Address Test Register (Autoincrement) Test, 6-2 Test 43 - Bt458 Main CLUT (Colormap) Test, 6-3 Test 44 - Bt458 Overlay CLUT (Colormap) Test, 6-4 Test 45 - Frame Buffer RAM Test (affected by prompt), 6-5 Test 46 - Frame Buffer Interrupt and Interrupt **Registers Test**, 6-7 Test 47 - I/O Location/Control Status **Registers Test**, 6-8 Test 48 - I/O Location (Space bits) Register Test, 6-9 Test 49 - 8-bit Registers Test, 6-10 Test 5 - I/O ASIC Reset Tests, 5-2 Test 50 - 16-bit Registers Test, 6-10 Test 51 - Dummy Registers Access Test, 6-11 Test 52 - Bt458 Registers Test, 6-12 Test 53 -Bt458 Address Register (Autoincrement) Test, 6-13 Test 54 - Bt458 Main CLUT (Colormap) Test, 6-14 Test 55 - Bt458 Overlay CLUT (Colormap) Test, 6-15 Test 56 - Colormap Pattern Test", 6-16"

Test 57 - Frame Buffer to Colormap Test, 6-17 Test 58 - Colorbar Test Pattern, 6-18 Test 59 - Address Register (Autoincrement) Test, 6-18 Test 6 - I/O ASIC FIFO/ECC Test, 5-3 Test 60 - Single Cursor Control Registers Test, 6-19 Test 61 - Both Cursors to Single Cursor Control Registers Test, 6-21 Test 62 - Single Cursor Pattern RAM Test, 6-22 Test 63 - Both Cursors to Single Cursor Pattern RAM Test, 6-23 Test 64 - Shadow RAM Test, 6-25 Test 65 - Shadow RAM to Bt458 Colormap Update Test, 6-26 Test 66 - Plane Major Mode Test (affected by prompt), 6-28 Test 67 - Pixel Major Mode Test (affected by prompt), 6-30 Test 68 - Pixel Major to Plane Major Test, 6-31 Test 69 - Single Plane Registers Test, 6-33 Test 7 - 7990 LANCE Initialization Test, 5-4 Test 70 - All Planes to Single Plane Registers Test, 6-34 Test 71 - BTLA Mode Single Plane Registers Test, 6-35 Test 72 - BTLA Mode All Planes to Single Plane Registers Test, 6-37 Test 73 - Mode-0 Read Access Test, 6-39 Test 74 - Mode-1 Read Access Test, 6-40 Test 75 - Mode-4 Read Access Test, 6-42 Test 76 - Mode-5 Read Access Test, 6-44 Test 77 - Mode-6 Read Access Test, 6-45 Test 78 - Mode-0 Write Access Test, 6-47 Test 79 - Mode-1 Write Access Test, 6-49 Test 8 - 7990 LANCE Internal Loopback Test, 5-4 Test 80 - Mode-2 Write Access Test, 6-51 Test 81 - Mode-3 Write Access Test, 6-53 Test 82 - Mode-4 Write Access Test, 6-55 Test 83 - Mode-5 Write Access Test, 6-57 Test 84 - Mode-6 Write Access Test, 6-59 Test 85 - Mode-7 Write Access Test, 6-61 Test 86 - Vertical Retrace Interrupts Test, 6-63 Test 87 - Framecount Register (Counter) Test, 6-64 Test 88 - Retrace/SCC Interrupt Combination Test, 6-65 Test 89 - Serial Ports Reset Test, 6-67 Test 9 - 7990 LANCE External Loopback Test (must be prompted), 5-5 Test 90 - Serial Ports Loopback Test, 6-68

Test 91 - Random ECC Generation Test (must be prompted), 7-1

Test 92 - Cache Data Bus Test (must be prompted), 7-2

Test 93 - RS-232-C Connector Loopback Test (must be prompted), 7-3

Test 94 - IDPROM checksum tests (affected by prompt), 7-11

Test 95 - Ethernet TFTP Read Test, 7-13

Test 96 - Disk Write/Read Test (affected by prompt), 7-14

Test 97 - Tape Write/Read Test (must be prompted), 7-19

Test 98 - NMI and Test Switch Test (must be prompted), 7-23

Test:

Handling failures, 2-9 Variations, 2-8

Tests:

Execution, 2-4 Numerical listing, 3-4 Overview, 3-1 Selection, 2-5

U

Using the -i option, 2-1