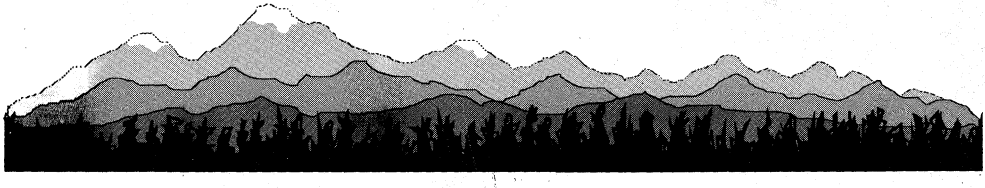
Solbourne Computer

SE Quick Reference Guide

Solbourne Confidential - Do Not Reproduce



Solbourne Computer, Inc. 303-772-3400
1900 Pike Road
Longmont, Colorado 80501

For Solbourne support call: 1-800-447-2861

Preface

This reference is intended as an abbreviated lookup resource for Solbourne field technical personnel. The information it contains is intended to cover a wide spectrum of reference material for the new as well as seasoned veteran. As you use it, please make notes in the blank Field Notes pages at the end of the book.

Please email all comments to refguide@solbourne.com.

Comments will be forwarded to everyone on the "ses" alias as well as archived for use in creating future editions to this guide. Especially note the sections in the guide that you use the least as well as those used the most. Apathy in this regard is a loud vote to make this the last edition.

The manual is divided into 15 sections, as follows:

Section 1 - System Configurations
: This section provides system configuration information for the Solbourne product line.

Section 2 - Hardware Overview
: The hardware characteristics of the Solbourne product family are discussed in this section.

Section 3 - Peripherals: Disk and Tape Drives and Boards
: Information on all peripherals shipped by Solbourne is given in this section.

Section 4 - Environmental Data
: This section gives all the environmental information associated with the Solbourne product family.

Section 5 - Boot Environment
: A quick reference on the booting procedures is given in this section.

Section 6 - Man Pages on Key System Administration Files
: Some of the most frequently used system administration man pages are given in this section.

Section 7 - Man Pages on Network Status Tools
: This section offers frequently used networking man pages.

Section 8 - YP Services
: This section gives information on setting and administrating YP.

Section 9 - Miscellaneous and 'How To ...' Information
: Considerable miscellaneous information is given in this section.

Section 10 - General Diagnostics Information
: This section introduces diagnostics and tells how to use the LEDs on the System and CPU boards.

Section 11 - System Power-On Self-Tests
: The system power-on self-tests for the Series4 and Series5 are given in this section.

Section 12 - dg Diagnostics
This section gives an overview of the standalone test controller (**dg**) diagnostics.

Section 13 - rdg Diagnostics
An overview of the ROM Resident Diagnostics (**rdg**) is given in this section.

Section 14 - mdg Diagnostics
The multiprocessing diagnostics (**mdg**) is covered in this section.

Section 15 - Field Notes
This section offers space for making notes.

Table of Contents

List of Figures

List of Tables

Section 1: System Configurations

1.1 Introduction

This section covers the PCB and peripherals configurations available in Solbourne Series4 and Series5/600, Series4 and Series5/500, Model 820, and Model 810,

1.2 Series4 and Series5/600

There are 14 bus slots (7 Kbus, 7 VMEbus) and five peripherals bays.

1.2.1 PCB Loading

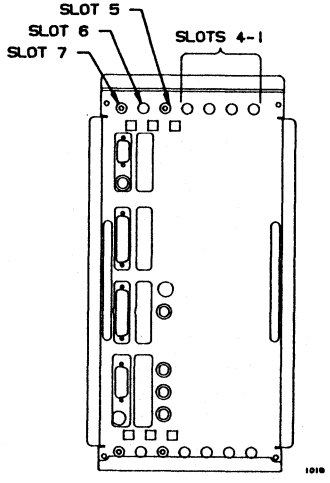


Figure 1-1. Kbus Slots in a Series4 or Series5/600

The following rules apply to Kbus PCBs in the Series4 and Series5/600:

- One required System I/O Board which includes Monochrome Graphics
- One or two CG30 Enhanced Color Frame Buffers in single-headed (e.g., single keyboard) configuration only (single X-server till next OS/MP release);
- Same rules for older CG40 Color Frame Buffer
- Maximum of five CPU Series4 or Series5 boards (software limit on CPUs is eight in 4.0C)
- Maximum of five Memory boards (slot limitation; Series5 board limited to 256 MBytes main memory addressing)
- VMEbus slots are loaded from the rightmost slot (1), out. See Section 3 for VMEbus boards supported.

1.2.2 Peripherals Loading

The peripherals loading rules in the Series4 and Series5/600 are as follows:

- Four full height (5 ¼-inch) bays for SCSI devices
- Three bays are reserved for full height SCSI hard disks
- Fourth bay may be configured with a full height hard disk, with one full height tape (e.g., Exabyte), or two half-height SCSI tape drives

1.3 Model 820

Companion cabinet to Series4 and Series5/600 and cosmetically identical to it, the Model 820 communicates with the Series4 and Series5/600 through its SMD/VMEbus cables and a SCSI cable. Has its own power supply(s) (one power supply for every two SMD drives). Four full SMD bays allowing a maximum of four SMD drives or three SMD plus one Exabyte 8mm SCSI. Ratio of drives:controllers can be 4:1, 3:1, 2:1, 1:1.

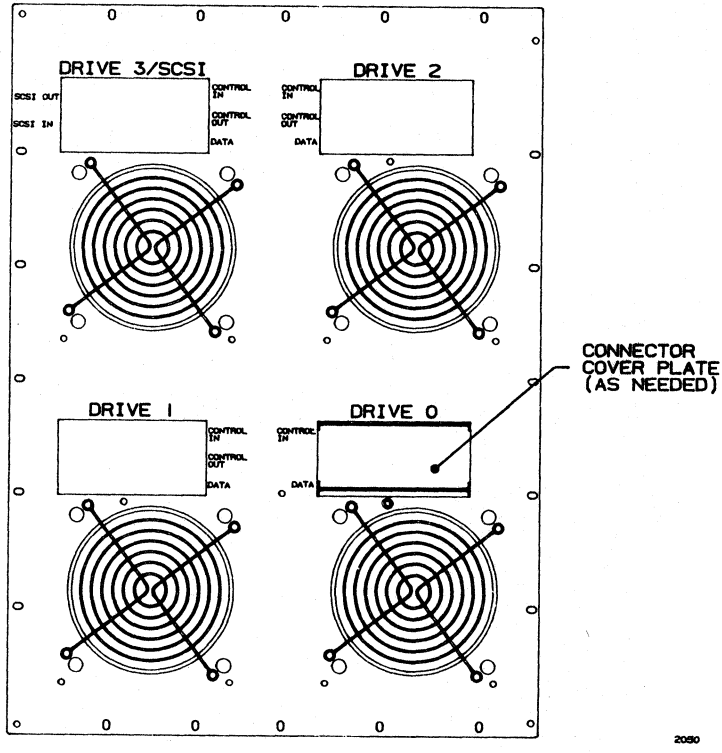


Figure 1-1. Model 820 Device Bays--Rear View

1.4 Series 4 and Series5/500

Board loading is as follows:

Table 1-1. Series4 PC Board Rules

Slot Number	CPU Used		Comment
	Series4	Series5	
Slot 5 (top)	System Board	System Board	Always in this slot
Slot 4	Color Graphics or expansion memory	Color Graphics expansion Memory, or third CPU	Has I/O cable slots
Slot 3 (middle)	First Memory	Second CPU, expansion memory	
Slot 2	Expansion Memory or expansion CPU	First CPU if multiprocessor	Leave empty if Series5 CPU in Slot 1
Slot 1 (bottom)	First CPU	CPU if uniprocessor	

1.4.1 Peripherals Rules

The following rules apply to Series4 and Series5/500 peripherals.

- One internal bay available for SCSI 200 Mbyte hard disk.
- Connectivity to one or two Model 810s (see Section 1-5) by daisy-chained single-ended SCSI cable.
- Up to five hard disks and up to four tape drives can be on-line to Series4 or Series5/500.
- Seven add-on SCSI devices is protocol limit per system.

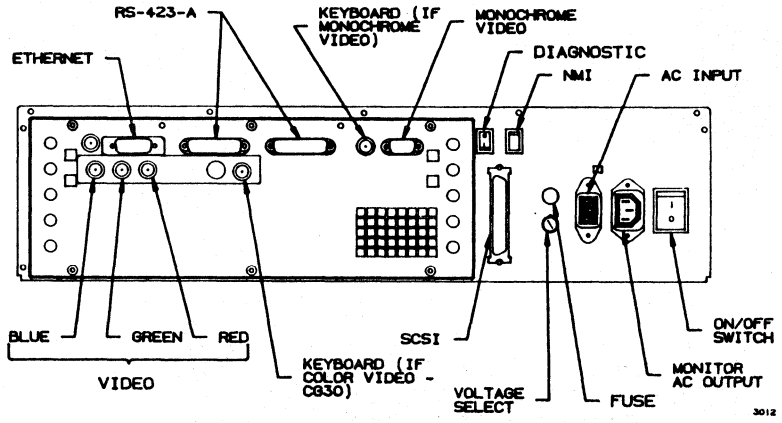


Figure 1-2. Connections for Series4 and Series5/500

1.5 Model 810

The Model 810 is the external SCSI peripherals package for the Series4 and Series5/500. Features of the Model 810 are:

- No boards on the Kbus
- Talks to the host through SCSI cable
- Has its own power supply
- Two half-height SCSI tape drives maximum (or 1 Exabyte 8mm drive)
- Two full height (5¼-inch) shock-mounted SCSI bays (two disk drives maximum)

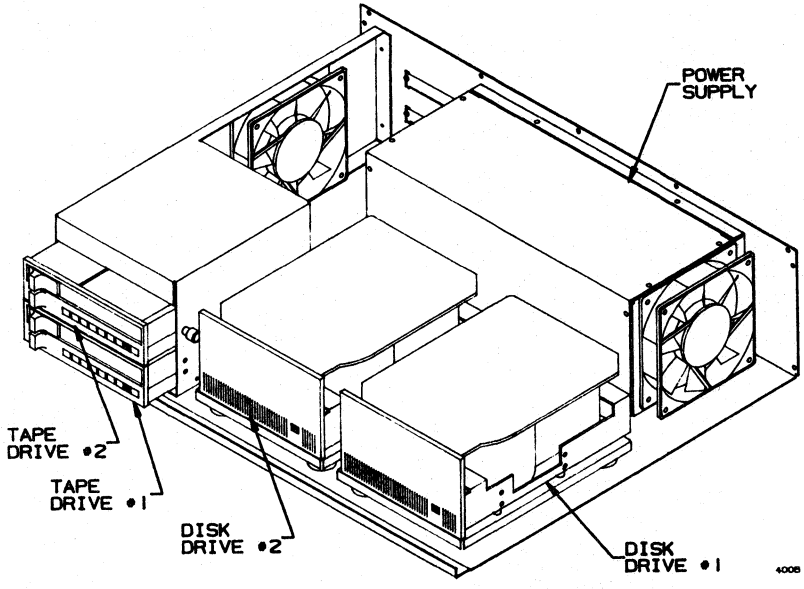


Figure 1-3. Fully Configured Model 810

Section 2: Hardware

2.1 Introduction

This section covers the proprietary backplanes and PCBs used in Solbourne Systems.

2.2 Kbus Backplane

General Kbus facts are as follows:

- 64 bit data bus/ 32 bit address bus
- 128 Mbytes/second transfer rate
- Seven slots for Series4 and Series5/600
- Five slots for Series4 and Series5/500
- Temperature sensor above Slot 4 in Series4 and Series5/600
- Slots numbered bottom-up in Series4 and Series5/500
- Slots numbered right-to-left in Series4 and Series5/600
- In Series4 and Series5/600, air flow restrictors required on empty slots
- In general, populate the Series4 and Series5/600 bus left-to-right

2.3 System Board

Features of the System I/O Board are as follows:

- System EAROM (IDPROM) resident on this board, see illustrations for location. See "Section 5: Boot Environment" for listing of environment variables.
- Monochrome Frame Buffer
  - 256 Kbyte Memory
  - Supports:1152 by 900 at 69 Hz using 126 Kbyte RAM (not Sun std-mono compatible)
  - 1600 by 1280 at 66 Hz using 250 Kbyte RAM (Sun high-res compatible)
- I/O ASIC
  - Synchronous SCSI, up to 5.0 Mbytes/second transfer rate
  - Ethernet in accordance to IEEE 802.3, 10 Mbits/second transfer rate
- Serial Ports
  - RS-423-A ports ttya and ttyb, superset of RS-232-C
  - RS-232-C compatible
  - 57.6 Kbaud asynchronous, 92.1 Kbaud synchronous Data Rates
  - Note change in serial port data/stop bit definitions with bootrom versions S4-3.2c and S5-3.3; see Section 9.4: setting up a vt100 on a tty port
- Keyboard and Mouse
  - Type 3: 126-key, Engineering-style, Sun4 compatible Keyboard (Cherry)
  - Type 4: 107-key, PC-style, Sun4 compatible Keyboard
  - 3-button optical mouse (Mouse Systems) with 4 foot cable
- SCSI implementation meets ANSI X3.131-1986, Small Computer System Interface and ANSI X3T9.2/85-53 Rev 4.B Common Command Set (CCS)
- VMEbus
  - Three Ribbon Connectors to System Board
  - Supports VMEbus Block Mode Transfer

☆ ☆ ☆ **NOTE** ☆ ☆ ☆

Maximum length of keyboard and monitor extension cable is 50 feet.

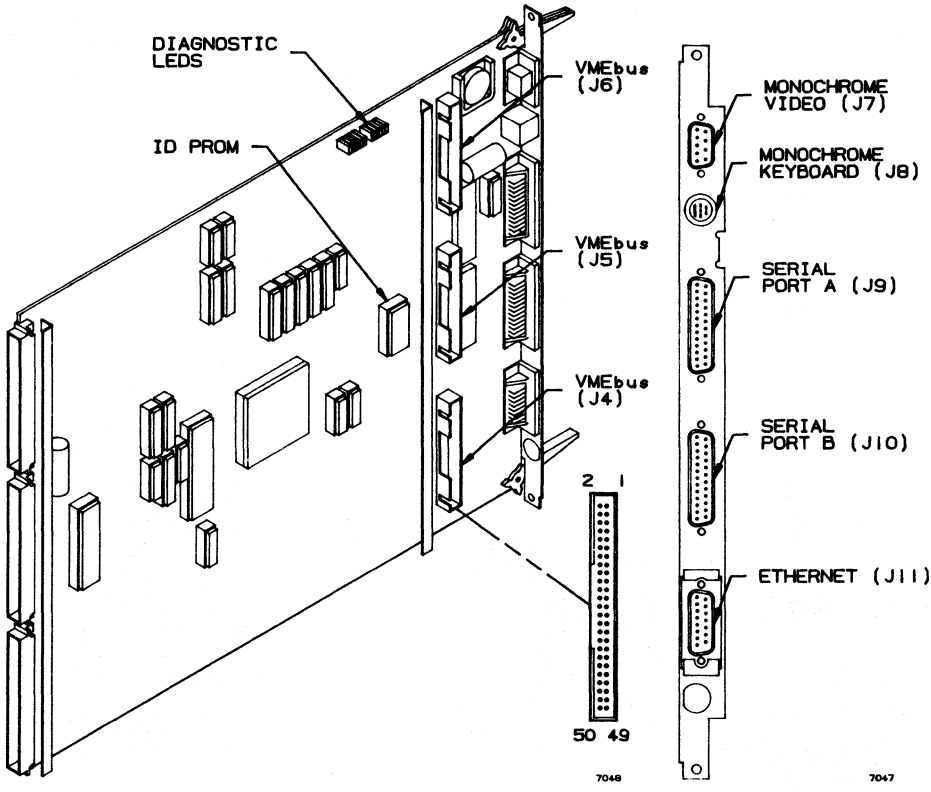


Figure 2-1. CA System Board Component Side and Cover Plate

2.3.1 System Board Revision Levels

Version DA/EA of the System Board has a configurable jumper setting which can be used to disable or enable each of the VMEbus interrupt levels. The board is set at the factory with all levels enabled. There are six customer-replaceable fuses (Solbourne part number 102888), shown in Figure 2-3. See System Board release notes for detailed version history.

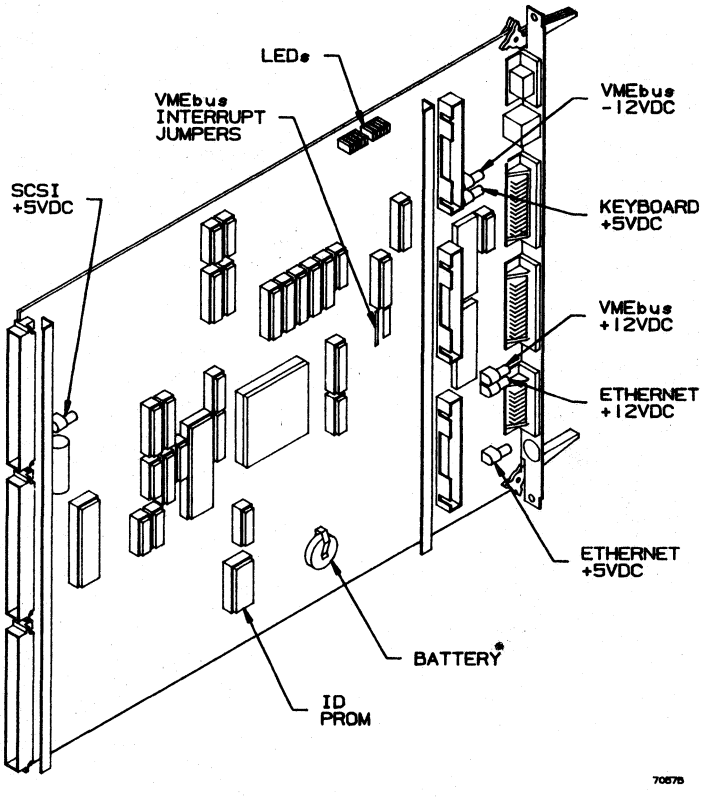


Figure 2-2. DA/EA Revision of System Board

Pin No.	Signal
Shell	Chassis Ground
1	Video +
2	Logic Ground
3	Horizontal sync
4	Vertical sync
5	Logic ground
6	Video -
7	Logic Ground
8	Logic Ground
9	Logic Ground

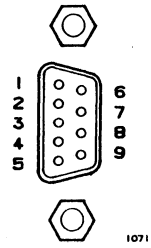


Figure 2-4. Monochrome Video Pinout

Pin Signal	Signal	Type
1	Clock	Input/output
2	Data	Input/output
3, 5	V_{cc}	Power
4, 8	Gnd	Power
6	not connected	
7	Mouse	Output
Shell	S.G.	Power

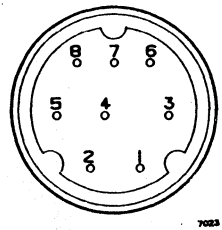


Figure 2-5. Keyboard Connector

Pin No.	Signal	Polarity
1	GND	N/A
2	CLSN	(+)
3	TRMT	(+)
4	GND	N/A
5	RECV	(+)
6	GND	N/A
7	Unused	N/A
8	GND	N/A
9	CLSN	(-)
10	TRMT	(-)
11	GND	N/A
12	RECV	(-)
13	+12 VDC	(+)
14	GND	N/A
15	Unused	N/A

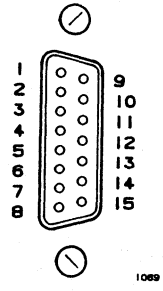


Figure 2-6. Ethernet Connector Pinout

Pin No.	Signal	Description
1	GND (Chassis Ground)	GND is physical ground to AC connector and beyond.
2	TXD	TXD is data transmitted to the DCE from the workstation.
3	RXD	RXD is data received from the DCE.
4	RTS	Normally RTS is a handshake signal to the DCE; on CA+ and earlier rev boards, it is connected to DTR, effectively cancelling both signals.
5	CTS (RTxC)	CTS is clear to send; an incoming signal from DCE indicating it's ready to accept data.
6	DSR	Data set (i.e., a modem) ready. Similar to CTS, but used on different systems.
7	Signal Ground (Common)	Reference voltage.
8	DCD	Data carrier detect; modem has received a phone call.
9-19	Unused	
20	DTR	Data terminal ready; a received handshake. On CA+ and earlier rev boards, it is connected to RTS, effectively cancelling both signals.
21-23	Unused	
24	TRxC	External transmit clock.
25	VERR	-5 VDC reference signal; used by some modems.

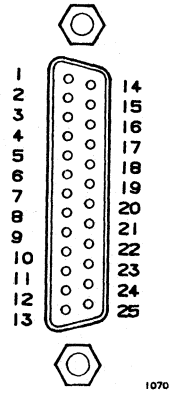


Figure 2-6. RS-423-A Serial Port Pinout (Two per System Board)

2.4 Series 4 CPU Board

Features of the Series4 CPU Board are as follows:

- 64K direct mapped virtual cache
- 16.67 MHz Fujitsu SPARC (RISC) MPU with Fujitsu floating point controller
- Weitek 1164/1165 FP chip set: 32 bit single-precision, 64 bit double-precision
- Hardware assisted MMU
- Board ID PROM resident, identifies what type of board this is to the system
- 64-bit data bus with ECC
- Four 512 by 8 Boot PROMS (located at U3400, U3401, U3402, U3403)
- Contains diagnostic LEDs, 7-segment displays (see Section 5)

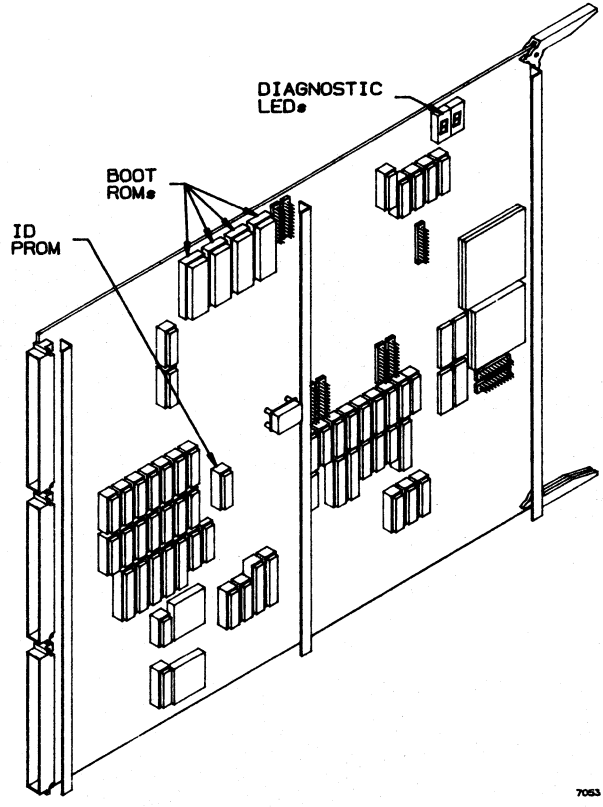


Figure 2-7. Series4 CPU Board

2.5 Series 5 CPU Board

Features of the Series5 CPU Board are as follows:

- 128K direct mapped physical cache
- 33.33 MHz Cypress SPARC (RISC) MPU with Weitek 3171 floating point unit
- Two level MMU, Fast and Global TLB
- Fast RIO Cycle
- Board-resident ID PROM identifies what type of board this is to the system
- 64-bit data bus with ECC
- Supports up to 256 Mbyte of RAM
- Four 512 by 8 Boot PROMS (located at U3400, U3401, U3402, U3403)
- Contains diagnostic LEDs, seven-segment displays (see Section 5)

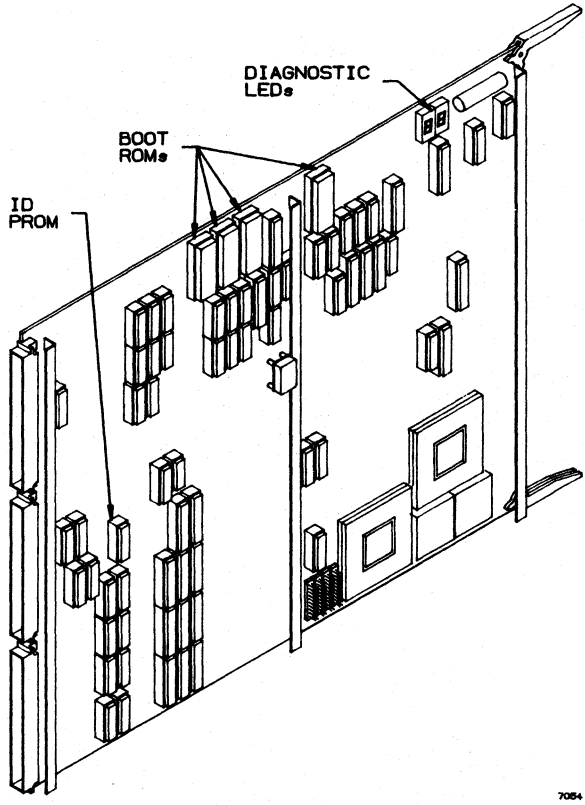
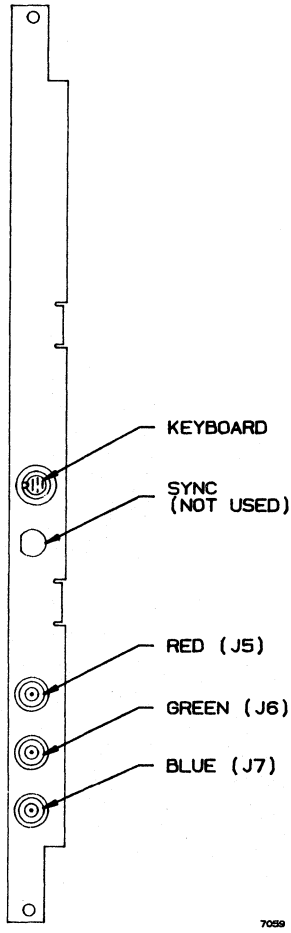


Figure 2-8. Series5 CPU Board

2.6 CG 40 and CG30 Color Frame Buffer Boards

Features of the Color Frame Buffer Boards are as follows:



Features of the CG 40 Color Frame Buffer Board are as follows:

- Discontinued, but supported
- Simultaneous display of 256 colors from a palette of more than 16.7 million
- BNC outputs: sync, red, green, blue Sync is imbedded in the green line, not a separate line
- Eight bit color storage, two bit overlay storage

Features of the CG 30 Color Frame Buffer are:

- Simultaneous display of 256 colors from a palette of more that 16.7 million
- BNC outputs: sync, red, green, blue
- Sync is imbedded in the green line, not a separate line
- Hardware support for cursor operations in X
- Hardware assist for Bit Blt operations

Figure 2-9. CG 30 Color Board Cover Plate

2.7 Memory Boards

Features of the Memory Board product line are as follows:

- ECC memory sold in 16 Mbyte increments
- ECC memory available in 16 Mbyte, 32 Mbyte, and 128 Mbyte boards
- 72 bit (64 data and 8 check bits) Kbus data interface
- 32 byte cache block memory transactions
- Software settable base address
- Software settable enables for memory reads and writes
- Board-resident ID PROM identifies what type of board this is to the system

2.8 VMEbus backplane

Features of the Solbourne VMEbus implementation are as follows:

- Seven slots, numbered from center of machine
- Bandwidth of 25 MByte
- Slot priority, with slot 1 having highest priority
- Card cage houses 6u "eurocard" format boards
- See the *Series4 and Series5/600 Service Manual*, Section 5, for more VMEbus information on backplane layout, connector specifications, signal termination, and bus arbitration

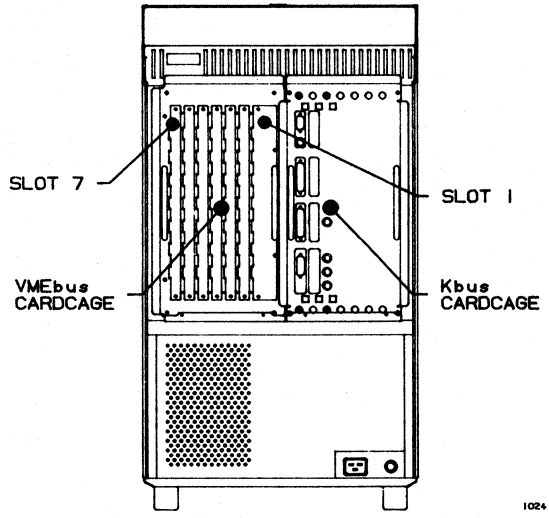


Figure 2-9. VMEbus Slots in Series4 and Series5/600

2.9 SCSI

Features of the Solbourne SCSI implementation are as follows:

- SCSI bus is terminated on both ends, at System Board and chassis SCSI port
- No termination is required or used on any installed disk drive
- Maximum cable length of 6 meters, including internal and external length

Table 2-1. SCSI Connector Pin Assignments

Pin No.	Signal	Pin No.	Signal
1	Ground	2	-Data Bus 0[1]
3	Ground	4	-Data Bus 1
5	Ground	6	-Data Bus 2
7	Ground	8	-Data Bus 3
9	Ground	10	-Data Bus 4
11	Ground	12	-Data Bus 5
13	Ground	14	-Data Bus 6
15	Ground	16	-Data Bus 7
17	Ground	18	-Data Bus P
19	Ground	20	Ground
21	Ground	22	Ground
23	Ground	24	Ground
25	Open	26	Termination Power
27	Ground	28	Ground
29	Ground	30	Ground
31	Ground	32	-Attention
33	Ground	34	Ground
35	Ground	36	-Busy
37	Ground	38	-Acknowledge
39	Ground	40	-Reset
41	Ground	42	-Message
43	Ground	44	-Select
45	Ground	46	-Control/Data
47	Ground	48	-Request
49	Ground	50	-Input/Output

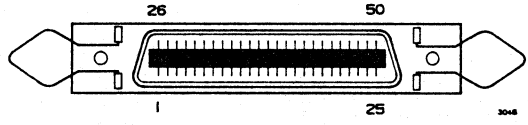


Figure 2-10. SCSI External Port

[1] A dash means active low.

2.10 Field Notes

Section 3: Peripherals: Disk Drives, Tape Drives, and Controllers

3.1 Introduction

This section gives configuration information on Solbourne's current peripherals offerings. The following peripheral devices are detailed in this section:

Table 3-1. Solbourne Peripherals

Type	Mfr	Model	Interface	Form Factor	Formatted Capacity
Disk	Maxtor	LXT-200	SCSI	3½-inch	200 Mbytes
Disk	Maxtor	XT-4380S	SCSI	5½-inch Full	327 Mbytes
Disk	Maxtor	XT-8760S	SCSI	5½-inch Full	661 Mbytes
Disk	Hitachi	DK514C-38	SCSI	5½-inch Full	327 Mbytes
Disk	Hitachi	DK515C-78C	SCSI	5½-inch Full	661 Mbytes
Disk	Fujitsu	M2383K	VMEbus/SMD	8 inch	830 Mbytes
Disk	Seagate	Sabre 9720-1230	VMEbus/SMD	8 inch	1040 Mbytes
Tape	Archive	2060S	SCSI	5¼ inch Half	60 Mbytes/cart.
Tape	Archive	2150S	SCSI	5¼ inch Half	150 Mbytes/cart.
Tape	H-P	88780B	SCSI	½-inch tape	140 Mbytes/reel
Tape	Exabyte	EXB-8200	SCSI	8 mm tape	2 Gbytes/cart.
Controller	Xylogics	753	VMEbus	N/A	N/A
Controller	Interphase	4207 Eagle	VMEbus	N/A	N/A
Controller	Solbourne/ Xylogics	VME/16 Line MUX	VMEbus	N/A	N/A

3.2 Maxtor LXT-200 200 Mbyte Hard Disk

This section describes the Maxtor 200 Mbyte disk.

Features of the LXT-200 are as follows:

- 200 Mbytes formatted capacity
- Fits in /500 chassis
- 15 milliseconds average seek time
- 9.2-14.8 Mbit/sec data transfer rate from disk
- 4 Mbyte/sec data transfer rate (synchronous) on SCSI bus
- 32 Kbyte buffer

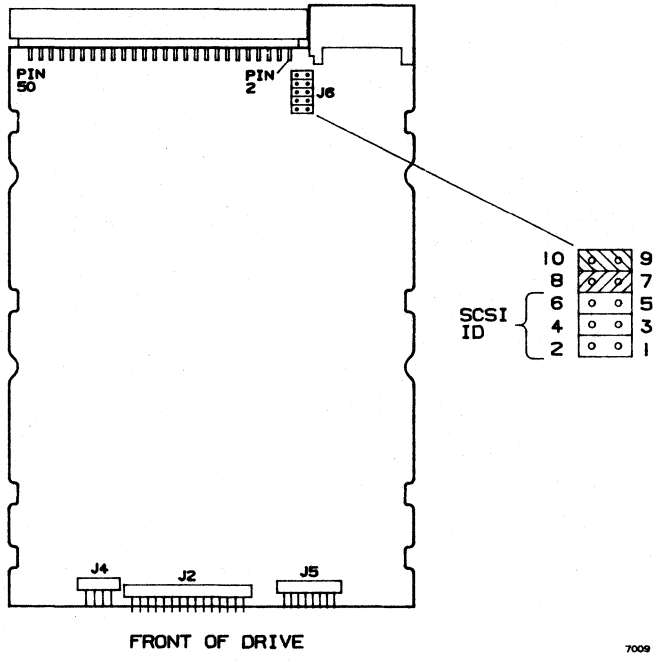


Figure 3-1. LXT-200 PCB

Table 3-2. LXT-200S Address Jumpers

SCSI Address	Pin Pair 5/6	Pin Pair 3/4	Pin Pair 1/2	Comments
0	out	out	out	Lowest priority; default setting
1	out	out	in	
2	out	in	out	
3	out	in	in	
4	in	out	out	Reserved for tape drive
5	in	out	in	Reserved for tape drive
6	in	in	out	Reserved
7	in	in	in	Reserved for controller

3.2.1 LXT-200 format.dat

The following information is in format.dat for the LXT-200.

```
disk_type = "Maxtor LXT-200"
                : ctlr = IOASIC
                : ncyl = 1300 : acyl = 2 : pcyl = 1314 : nhead = 7 : nsect = 43
                : rpm = 3600 : bpt = 22528
                : cache = 0x00 : nzone = 3 : atrks = 0
partition = "Maxtor LXT-200"
                : disk = "Maxtor LXT-200" : ctlr = IOASIC
                : a = 0, 17157 : b = 57, 66220 : c = 0, 391300 : d = 277, 18963
                : g = 340, 288960
```

3.3 Maxtor XT-4380S SCSI 5 ¼-inch Full Height 327 Mbyte Disk

This section describes the XT-4380S. Features of the XT-4380S are:

- 327 Mbytes formatted capacity
- Fits in Model 810 and /600 chassis
- 18 milliseconds average seek time
- 10 Mbit/sec data transfer rate from disk
- 4 Mbyte/sec data transfer rate (synchronous) on SCSI bus
- 64 Kbyte buffer

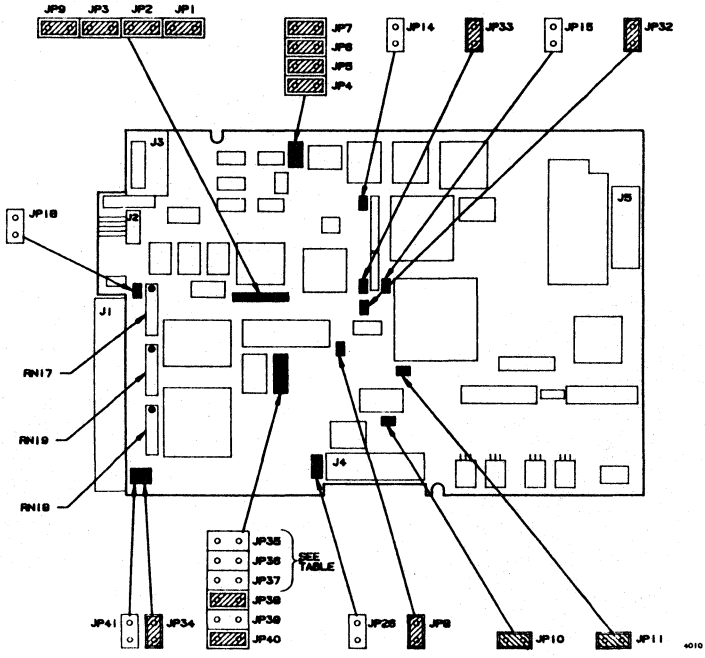


Figure 3-2. XT-4380S TLA 1094448 PCB

Table 3-3. XT-4380S Address Jumper Settings (All TLAs)

SCSI Address	JP37	JP36	JP35
0	out	out	out
1	out	out	in
2	out	in	out
3	out	in	in
4	in	out	out
5	in	out	in
6	in	in	out
7	in	in	in

3.3.1 XT-4380S format.dat

The following information is in format.dat for the XT-4380S.

```
disk_type = "Maxtor XT-4380S"
                    : ctlr = IOASIC : fmt_time = 3
                    : ncyl = 1218 : acyl = 2 : pcyl = 1224 : nhead = 15 : nsect = 35
                    : rpm = 3600 : bpt = 20833
                    : cache = 0x11
partition = "Maxtor XT-4380S"
                    : disk = "Maxtor XT-4380S" : ctlr = IOASIC
                    : a = 0, 16800 : b = 32, 66150 : c = 0, 639450 : d = 158, 19425
                    : g = 195, 537075
```

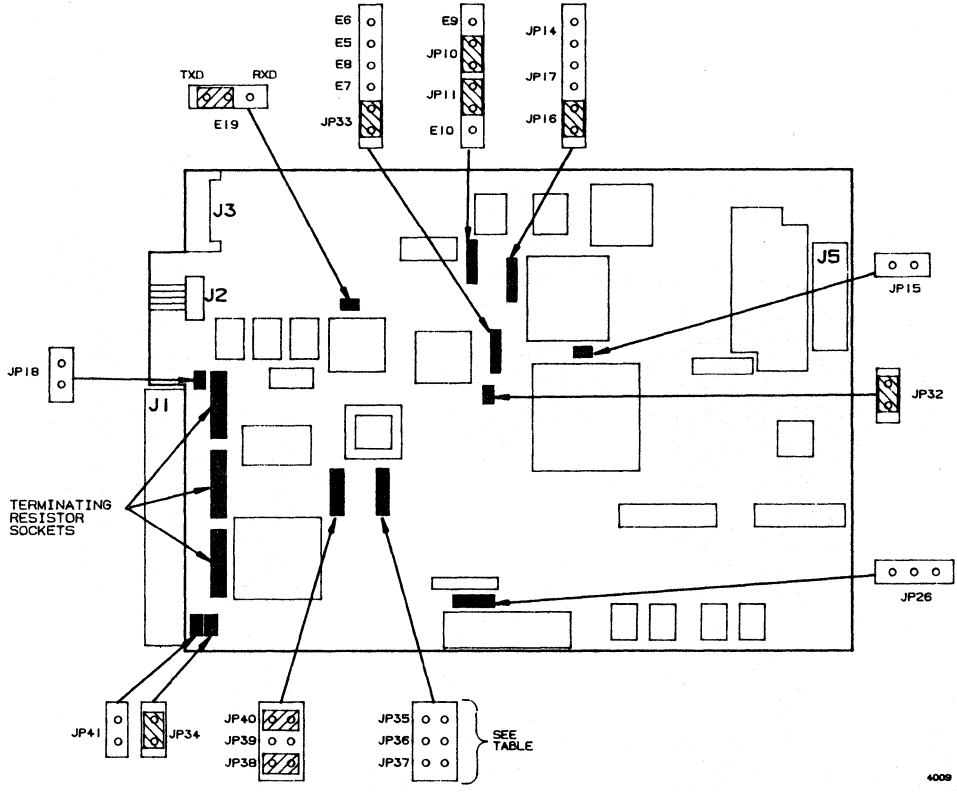


Figure 3-3. XT-4380S TLAs 1094708 and 1094868 PCB

3.4 Maxtor XT-8760S SCSI 5 ¼-inch Full Height 661 Mbyte Disk

This section describes the XT-8760S.. Features of the XT-8760S. are:

- 661 Mbytes formatted capacity
- Fits in Model 810 and /600 chassis
- 18 milliseconds average seek time
- 15 Mbit/sec data transfer rate from disk
- 4 Mbyte/sec data transfer rate (synchronous) on SCSI bus
- 64 Kbyte buffer

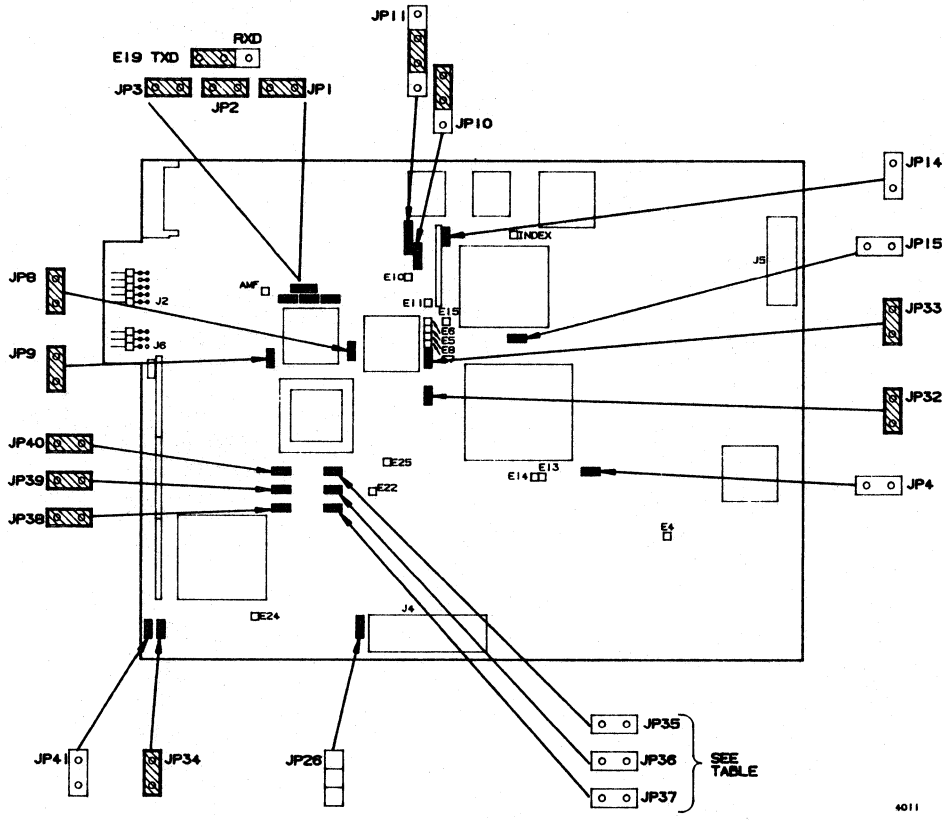


Figure 3-4. XT-8760S PCB

Table 3-4. SCSI Device Identifier Jumpers on the XT-8760S

SCSI Address	JP37	JP36	JP35
0	out	out	out
1	out	out	in
2	out	in	out
3	out	in	in
4	in	out	out
5	in	out	in
6	in	in	out
7	in	in	in

3.4.1 XT-8760S format.dat

The following information is in format.dat for the XT-8760S.

```
disk_type = "Maxtor XT-8760S"
            : ctlr = IOASIC : fmt_time = 3
            : ncyl = 1626 : acyl = 2 : pcyl = 1632 : nhead = 15 : nsect = 53
            : rpm = 3600 : bpt = 31410
            : cache = 0x11
partition = "Maxtor XT-8760S"
            : disk = "Maxtor XT-8760S" : ctlr = IOASIC
            : a = 0, 16695 : b = 21, 66780 : c = 0, 1292670 : d = 105, 19080
            : g = 129, 1190115
```

3.5 Hitachi DK514C-38 SCSI 5 ¼-Inch Full-height 327 Mbyte Disk

This section describes the DK514C-38. Features of the DK514C-38 are:

- 327 Mbytes formatted capacity
- Only works on OS/MP 4.0C and up
- Fits in the Model 810 and /600 chassis
- 16 milliseconds average seek time
- 4 Mbyte/sec data transfer rate (synchronous) on SCSI bus

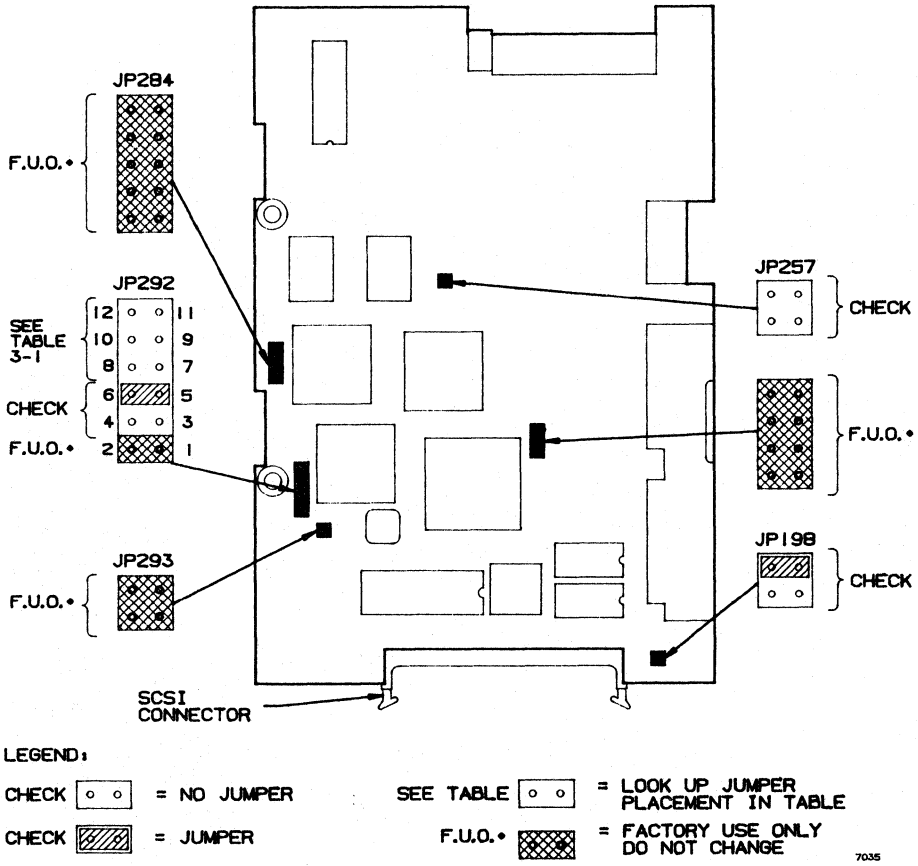


Figure 3-5. DK514C-38 PCB

Table 3-5. SCSI Address on Jumper JP292

SCSI ID	Pins 7-8	Pins 9-10	Pins 11-12	Comment
0	in	in	in	Default
1	in	in	out	
2	in	out	in	
3	in	out	out	
4	out	in	in	
5	out	in	out	
6	out	out	in	
7	out	out	out	Reserved

3.5.1 DK514C-38 format.dat

The following information is in format.dat for the DK514C-38.

```
disk_type = "Hitachi DK514C-38"
            : ctlr = IOASIC : fmt_time = 3
            : ncyl = 896 : acyl = 2 : pcyl = 898 : nhead = 14 : nsect = 51
            : rpm = 3600 : bpt = 226112
            : cache = 0x11
partition = "Hitachi DK514C-38"
            : disk = "Hitachi DK514C-38 " : ctlr = IOASIC
            : a = 0, 16422 : b = 23, 66402 : c = 0, 639744 : d = 116, 19278
            : g = 143, 536928
```

3.6 Hitachi DK515C-78C SCSI 5 ¼-Inch Full-height 661 Mbyte Disk

This section describes the DK515C-78C. Features of the DK515C-78C are:

- 661 Mbytes formatted capacity
- Only works on OS/MP 4.0C and up
- Fits in the Model 810 and /600 chassis
- 16 milliseconds average seek time
- 4 Mbyte/sec data transfer rate (synchronous) on SCSI bus

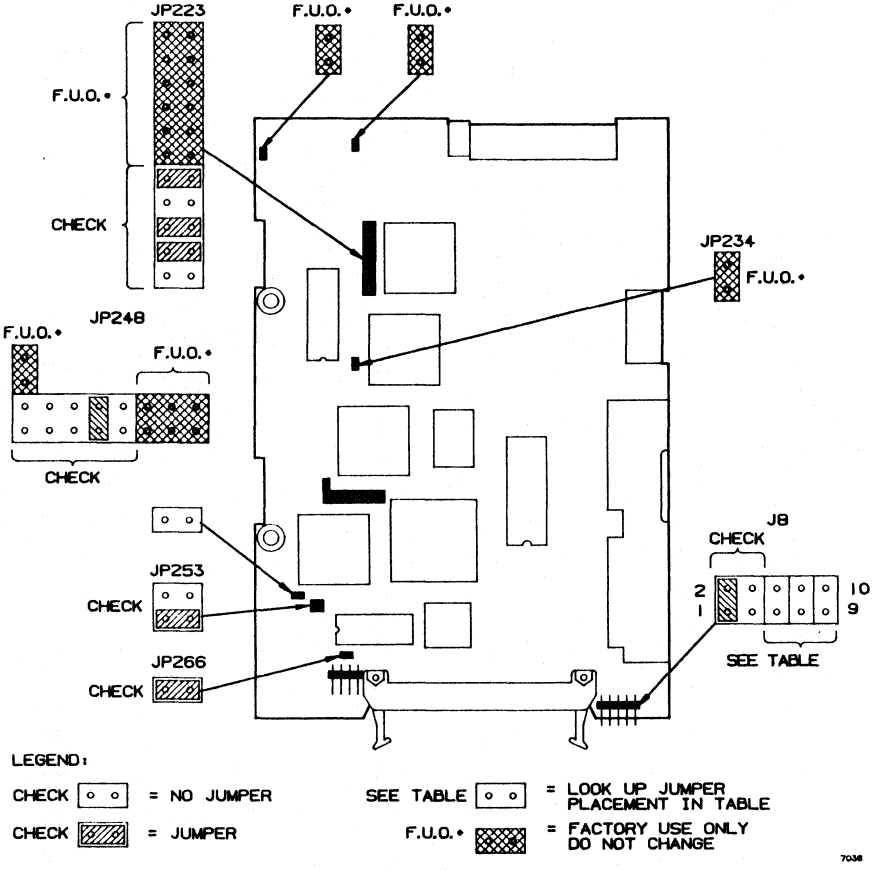


Figure 3-6. DK515C-78C PCB

Table 3-6. Jumper J8, SCSI Address

SCSI ID	Pins 5-6	Pins 7-8	Pins 9-10	Comment
0	out	out	out	Default
1	out	out	in	
2	out	in	out	
3	out	in	in	
4	in	out	out	
5	in	out	in	
6	in	in	out	
7	in	in	in	Reserved

3.6.1 DK515C-78C format.dat

The following information is in format.dat for the DK515C-78C.

```
disk_type = "Hitachi DK515C-78"
                : ctlr = IOASIC : fmt_time = 3
                : ncyl = 1339 : acyl = 2 : pcyl = 1356 : nhead = 14 : nsect = 69
                : rpm = 3600 : bpt = 35328
                : cache = 0x11
partition = "Hitachi DK515C-78"
                : disk = "Hitachi DK515C-78" : ctlr = IOASIC
                : a = 0, 17388 : b = 18, 66654 : c = 0, 1293474 : d = 87, 19320
                : g = 107, 1190112
```

3.7 Fujitsu M2382K SMD 8-inch 830 Mbyte Disk

This section describes the M2382K. Features of the M2382K are:

- 830 Mbytes formatted capacity
- Fits in Model 820 chassis (up to 4 per chassis)
- 3 Mbyte/sec data transfer rate from disk
- 16 milliseconds average seek time

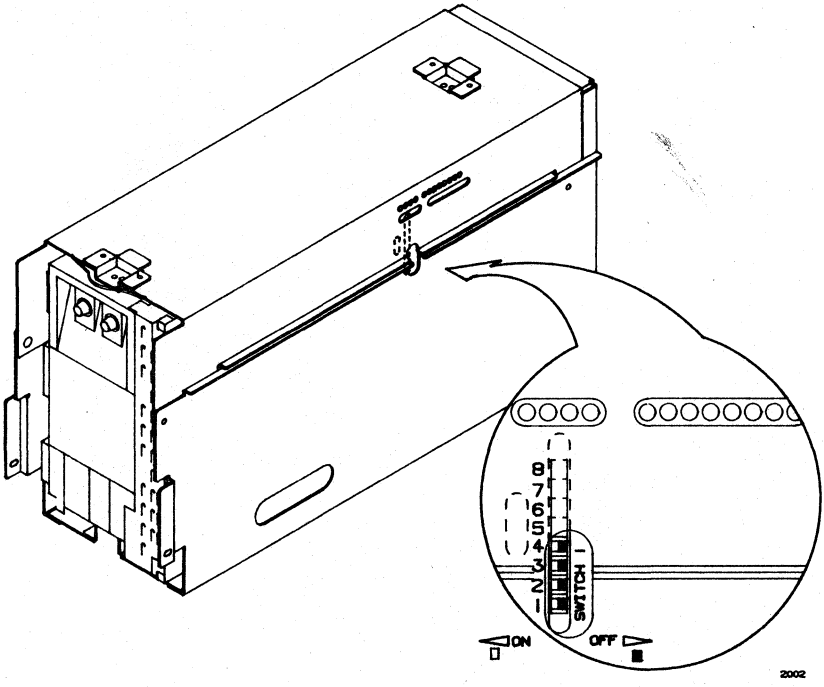


Figure 3-7. M2382K DIPs

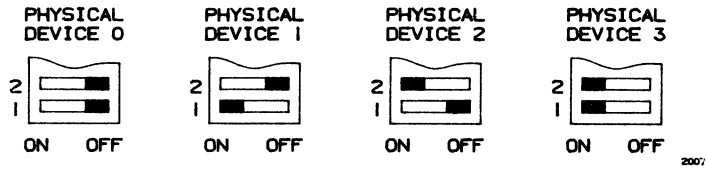


Figure 3-8. M2382K Address Settings

3.7.1 M2382K format.dat

The following information is in format.dat for the M2382K.

```
disk_type = "Fujitsu M2382K"
                : ctlr = XD753
                : ncyl = 743 : acyl = 2 : pcyl = 745 : nhead = 27 : nsect = 81
                : rpm = 3600 : bpt = 49728 : bps = 604
partition = "Fujitsu M2382K"
                : disk = "Fujitsu M2382K" : ctlr = XD753
                : a = 0, 17496 : b = 8, 65610 : c = 0, 1624941 : d = 38, 19683
                : g = 47, 1522152
```

3.8 Seagate Sabre 9720-1230 SMD 8-inch One Gbyte Disk

This section describes the Sabre 9720-1230. Features of the Sabre 9720-1230 are:

- 1040 Mbytes formatted capacity
- Fits in Model 820 chassis (up to 4 per chassis, combinations with Fujitsu allowed)
- 3 Mbytes/sec data transfer rate from disk
- 16 milliseconds average seek time

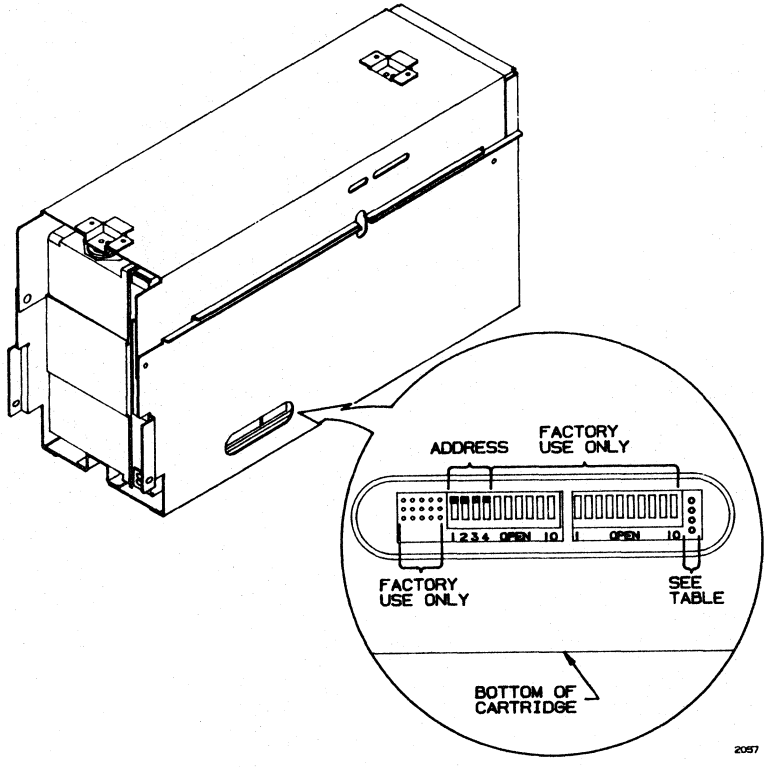


Figure 3-9. Sabre 9720-1230 PCB

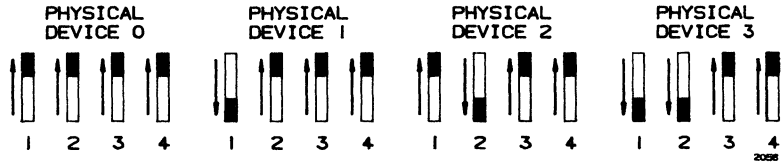


Figure 3-10. Sabre 9720-1230 Address Settings

3.8.1 Sabre 9720-1230 format.dat

The following information is in format.dat for the Sabre 9720-1230.

```
disk_type = "Sabre 9720-1230-1GB"
                : ctlr = XD753
                : ncyl = 1633  : acyl = 2 : pcyl = 1635 : nhead = 15 : nsect = 8
                : rpm = 3600 : bpt = 50400 : bps = 599 : bfi_skew = 8
partition = "Sabre 9720-1230-1GB-599"
                : disk = "Sabre 9720-1230-1GB-599" : ctlr = XD753
                : a = 0, 17430 : b = 14, 65985 : c = 0, 2033085 : d = 67, 19920
                : g = 83, 1929750
```

3.9 Archive 2060S QIC-24 and 2150S QIC-150 Half-height, ¼-Inch Tape Drive

This section describes the Archive 2060S. Features of the Archive 2060S are:

- Fits in Model 810 and /600 chassis
- 1.25 Mbyte/sec data transfer rate (asynchronous) on SCSI bus
- 2060S has 60 Mbyte capacity using DC-600A or 600 XTD cartridges
- 2150S has 150 Mbyte capacity using DC-600A or 600 XTD cartridges

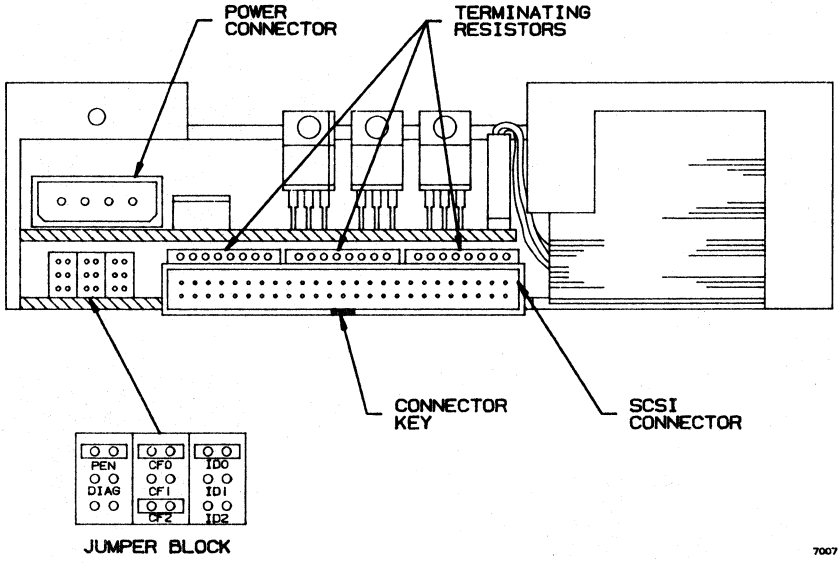


Figure 3-11. Archive 2060S and 2150S Jumpers

Table 3-7. SCSI Address Jumpers on Archive Drives

SCSI Address	ID2	ID1	ID0
0	out	out	out
1	out	out	in
2	out	in	out
3	out	in	in
4	in	out	out
5	in	out	in
6	in	in	out
7	in	in	in

3.10 Exabyte EXB-8200 SCSI 5 ¼-Inch full height 8 mm Cartridge Tape

This section describes the Exabyte EXB-8200. Features of the Exabyte EXB-8200 are:

- Fits in Model 810, /600, and Model 820 chassis.
- 1.5 Mbyte/sec data transfer rate (asynchronous) on SCSI bus
- 2 Gigabyte capacity per standard 8mm video cartridge

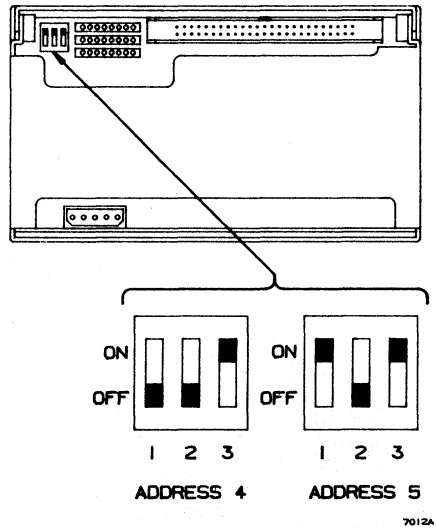


Figure 3-12. Exabyte EXB-8200 Jumpers

3.11 H-P 88780B SCSI ½-Inch Reel Tape Drive

This section describes the H-P 88780B. Features of the H-P 88780B are:

- 198 Kbytes/sec data transfer rate @ 1600 bpi PE (Phase Encoded)
- 747 Kbytes/sec data transfer rate @ 6250 bpi GCR (Group-Coded Recording)
- 93 Kbytes/sec data transfer rate @ 800 bpi NRZI (Non-Return to Zero Inverted)
- 512 Kbyte cache buffer
- 125 ips nominal tape speed
- SCSI cable is connected to either SCSI connector on the back of drive

3.11.1 Changing the 88780B's SCSI Address

To change to another SCSI address, follow these steps:

1. Take the drive offline.
2. Press OPTION to enter the Option Mode. TEST * appears in the display.
3. Press NEXT until ADDR * or ID * appears in the display. ADDR * appears if you have a Pertec-compatible interface, ID * appears if you have a SCSI interface.
4. Press ENTER.
5. Using NEXT or PREV, bring the ADDRESS/ID number desired into the display.
6. Press ENTER. The ADDRESS/ID you selected appears as SET <#>.
7. Leave the Option Mode by pressing OPTION or RESET.

Table 3-8. /dev Entries for the H-P 88780B

Device Name	SCSI Address	Density
rst0/rmt0	0x4	1600 (PE)
rst1/rmt1	0x5	1600 (PE)
rst8/rmt8	0x4	6250 (GCR)
rst9/rmt9	0x5	6250 (GCR)
rst16/rmt16	0x4	800 (NRZI)
rst17/rmt17	0x5	800 (NRZI)

3.12 Xylogics 753 Controller Board

This section details the Xylogics 753.

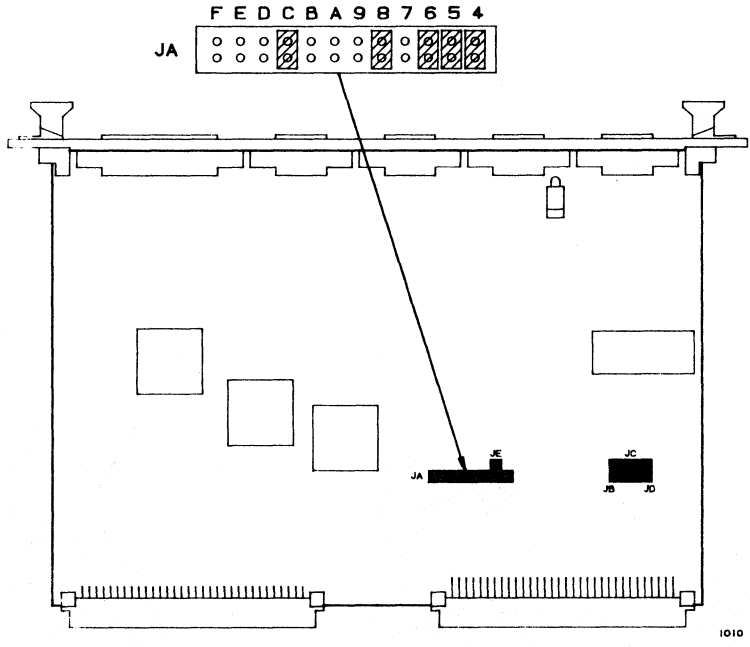


Figure 3-13. Jumper Settings on the Xylogics 753

Table 3-9. Xylogics Base Address Selection

Address	F	E	D	C	B	A	9	8	7	6	5	4
0xEE80	0	0	0	I	0	0	0	I	0	I	I	I
0xEE90	0	0	0	I	0	0	0	I	0	I	I	0
0xEEa0	0	0	0	I	0	0	0	I	0	I	0	I
0xEEb0	0	0	0	I	0	0	0	I	0	I	0	0

3.13 VMEbus/16 Line Multiplexer

This section describes the VMEbus/16 Line Multiplexer. Features of the VMEbus/16 Line Multiplexer are:

- Adds 16 asynchronous channels per board, housed in one 6U-sized VME slot
- Up to 64 channels, four boards, in one /600 or /800 system
- Supports devices such as terminals, line or laser printers, and modems
- Each channel can transfer data at rates ranging from 50 baud to 38.4 Kbaud
- Devices appear as ttyXY where X is the controller # and Y is the port on the controller; e.g. tty12 is the 3rd port (2) on the 2nd Mux card (1).

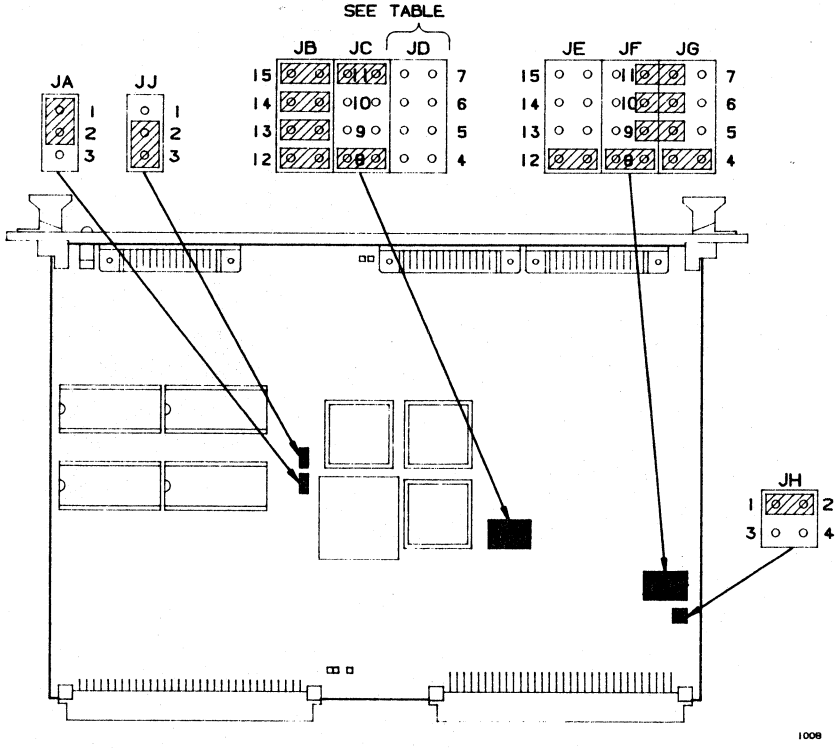


Figure 3-14. Solbourne Multiplexer Board Default Jumper Settings

Table 3-10. MUX Jumper Meanings

Jumper Name	Purpose	Set by Whom
JA	Crystal speed selection	User
JB	VMEbus address	Solbourne factory
JC	VMEbus address	Solbourne factory
JD	VMEbus address	User
JE	Bus Request/grant	Solbourne factory
JF	Bus request/grant	Solbourne factory
JG	Bus request/grant	Solbourne factory
JH	Crystal Speed/Diagnostics	User
JJ	ROM size	Solbourne factory

Table 3-7 shows the settings on jumpers JA and JH that are required to change the board's crystal speed.

Table 3-11. Line Rate Per Jumper Positions

Line Rate	Jumper Positions
0	JA: jumper pin 2 to 3 JH: no jumper between 1 and 2
1	JA: jumper pin 1 to 2 JH: jumper pin 1 to 2

Table 3-12. Setting Jumper JD for VMEbus Addresses

Board Number	Address	Pin 4	Pin 5	Pin 6	Pin 7
0	Ox0620	in	out	in	in
1	Ox0640	in	in	out	in
2	Ox0660	in	out	out	in
3	Ox0680	in	in	in	out

3.14 Interphase Eagle 4207 Ethernet Board

This section details the Eagle 4207.

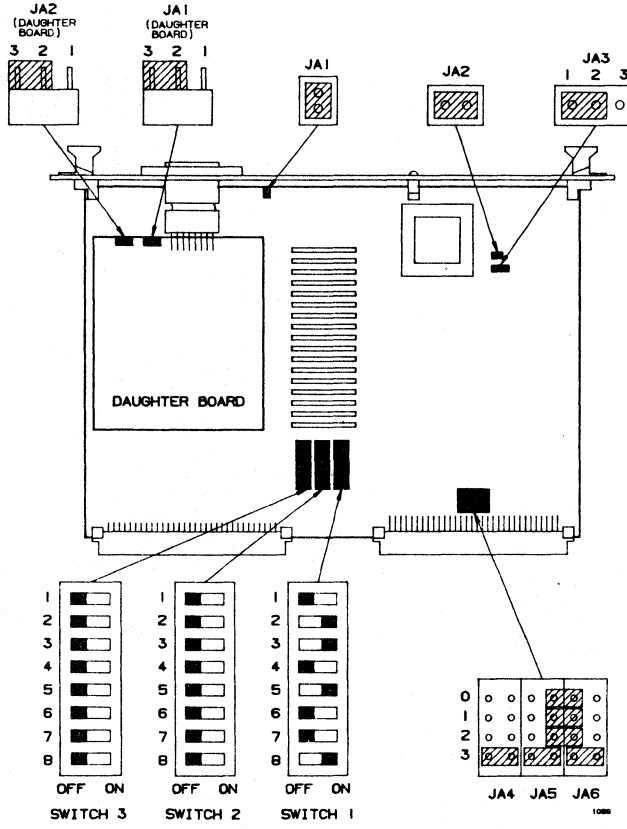


Figure 3-15. Jumper Settings on the Eagle 4207

Table 3-13. Eagle Jumper Block and Switch Functions

Jumper Block/Switch Number	Function
JA 2 — daughter board	Full/half AUI power
JA1 — daughter board	Transceiver power
JA1	Factory test
JA2	Factory test
JA3	EPROM size
JA4,5,6	VMEbus request level
Switches 1, 2, 3	Configuration switches

Section 4: Environmental Data

4.1 Power Ratings, BTU Ratings, and Amperage Requirements

The following table gives the power ratings, BTU ratings, and Amperage requirements for Solbourne products.

Solbourne Products
Power Ratings, BTU Ratings, Amperage Requirements

Solbourne Model	Typ Amps	Rated Volts	Typical BTU/hour	Slow-Blow Fuse Rating	Req Amp Service	Max BTU @ Fuse Rate
/600	7.87	110/120	3088	12A	15A	4708
/600	4.25	220/240	3335	6A		4708
820	6.18	110/120	2425	12A	12A	4708
820	3.91	220/240	3068	6A		4708
/500	5.56	110/120	2182	10A	12A	3924
/500	2.78	220/240	2182	6A		4708
810	1.07	110/120	420	6A	10A	2354
810	0.65	220/240	510	3A		2354
19" Color(140w)		100/120	478	4A	6A	1569
19" Color(140w)		220/240	478	3.15A	5A	2472
16" Color(140w)		100/120	478	4A	6A	1569
16" Color(140w)		220/240	478	3.15A	5A	2472
New Monochrome(100w)		110/120	341	1.5A	3A	589
New Monochrome(100w)		220/240	341	0.8A	2A	628
H-P 88780B ½" Tape		100/120	850	n/a	3A	1280
H-P 88780B ½" Tape		200/240	850	n/a	2A	1280

Notes:

Typ Amps = Measured amperage during UL tests, fully card-populated.
Typ BTU/hour = Calculated @3.412 BTU/hr/watt using Typ Amp figures.
Slow-Blow Fuse Rating = Maximum draw; Nominal Amp Rating.
Max BTU @ Fuse Rate = Calculated BTU/hr based on Fuse Rating.
Figures for individual components/cards not available at this time.

4.2 Operating Temperature

The numbers in this section were taken from the product installation manuals.

- For /500, /600
  Power on: +10° to +40° C (+50° to +104° F)
  Power off: -20° to +75° C (-4° to +167° F)
- For /810
  Power on: +10° to +35° C (+50° to +95° F)
  Power off: -20° to +75° C (-4° to +167° F)
- For /820
  Power on: +10° to +30° C (+50° to +86° F)
  Power off: -40° to +60° C (-40° to +140° F)
- Maximum Temperature Gradient(w/o tape): 15° C (59° F) per hour
  Maximum Temperature Gradient(w tape): 5° C (41° F) per hour

4.3 All Disk Drives: Special Handling for Temperature Changes

When bringing the drive package in from outside, prevent condensation on the drive by allowing the drive to warm up to room temperature before opening the plastic covering. Allow one hour for each 18° F of temperature difference. For example, if it is 20° F outside and 70° F inside (a 50° F difference) wait about 3 hours (50/18 = 2.77 hours) to warm up the drive before opening.

4.4 Operating Humidity

Power on: 20% to 80%, non-condensing at 40° C
Power off: Up to 95%F, non-condensing at 40° C

4.5 Regulation Certification

UL 478, CSA 220, TUV (qualified "GS" mark)
FCC-A
CISPR-22A (VCCI-A)
VDE-A("qualified pass" with Series 5)
X-ray Emit - DHHS Rule 21 (subchapter J), PTB

CSA on Series*N* /600 and Model 820 expected May 1990.

4.6 Field Notes

Section 5: Boot Environment

5.1 EAROM Environment Variables

EAROM environment variables visible at ROM prompt via 'printenv':

```
HOSTID          Read-only variable set in manufacturing, specifies host ID
SERIAL          Read-only variable set in manufacturing, specifies serial #
ENETADDR        Read-only variable set in manufacturing, specifies Ethernet addr
MODEL           Read-only variable set in manufacturing, specifies unit model
PORT_A_BAUD     Specified the baud rate for ttya (defaults to 9600)
PORT_B_BAUD     Specified the baud rate for ttyb (defaults to 9600)
BOOTMODE        Cold and warm reset action - auto (default) or manual
DISPLAYRES      Resolution of the console display, defaults to 1152x900
MASTER          Defines master CPU slot #, defaults to lowest slot #
DIAGBOOT        Where to boot diagnostics from, defaults to sd.si(,,6)stand/dg
DEFAULTBOOT     Where to boot UNIX from, defaults to sd.si()/vmunix
CONSOLE         Type of console: zs() or fb()
DEFAULTSWAP     Specifies swap partition, defaults to sd.si(,,1)
DEFAULTROOT     Specifies root partition, defaults to sd.si()
```

To print all the environment variables' values, at the ROM prompt, type:

```
ROM> pr    (for printenv)
```

envedit
The first four variables can only be entered once; after that they are only alterable with the "envedit" program on a tape cartridge.

At the ROM prompt, insert "envedit" tape and type:

```
ROM> b st.si(,4,)          (for tape drives with an address of 4)
```

Enter the name of the variable to change, then Return
Enter the new value, then Return Return to exit "envedit."

5.2 BOOT ROM Command List

The BOOT ROM accepts various commands to boot and start programs, display and change contents of memory, & display and change environment variables.

The following UNIX commands are available at the ROM prompt: For more information on these commands and their options see **bootrom** (8)

b	boots the program specified by DEFAULTROOT or DIAGBOOT
go	starts the program from the entry point

examine	displays the contents of ranges of memory
deposit	changes the contents of memory at a specified address
printenv	displays the value of a named environment variable
setenv	changes the value of a named environment variable
unsetenv	deletes the value of a named environment variable
cp	copies the contents of a source file to a destination file
mode	changes the access mode used by examine and deposit commands
reset	calls the rom reset routine, i.e. cold, warm, halt, autoboot
help	prints the syntax of the command specified
?	same as 'help'
rdg	invokes the Extended ROM Diagnostic Program
date	without an argument, displays the current date and time otherwise, current date is set as specified by argument
ls	prints a list of files in a specified directory

5.3 Boot Command Options

-s	Single user
-w	Write
-a	Interactive
-b	Skip rc.boot
-m *n*	Limit available memory to 'n' MBytes
-M	Master CPU only
-d	Boots to the program specified by the DIAGBOOT environment variable

5.4 Booting from Specific Devices

The boot command has the syntax:

```
ROM> b device(parameters)pathname args
```

Booting from a disk other than the specified default:

```
ROM> b tape(controller, unit, filenumber)pathname args
```

An example of booting from tape:

```
ROM> b st.si(,4,) -a
```

The above example example boots from the first file (0), drive address 4, controller 0 in interactive mode.

Booting from a disk other than the specified default:

```
ROM> b controller(address, drive, partition)pathname args
```

An example of booting from an alternative disk:

```
ROM> b sd.si()/vmunix.test
```

The above example boots from the first file (0), drive 0, controller 0, to the file named vmunix.test

Booting to a 4.0C Release Tape, using a local tape drive:

```
ROM> b st.si(,4,2) -swabM          for Series4, use st.si(,4,3)
. . .
rootfilesystem type ( spec 4.2 nfs lo ) : 4.2
root device ( sd%d[a-h] ns%d[a-h] rd%d[a-h] ) : rd0a
initialize ram disk from device ( st%d[a-h] ... ) : st0
file number : 5
. . .
swap filesystem type ( spec 4.2 nfs lo ) : spec
swap device ( sd%d[a-h] ns%d[a-h] ... ) : ns0b
. . .
```

5.5 Device_name/Protocol_name Abbreviations

sd	SCSI disk
st	SCSI tape
si	SCSI I/O ASIC controller
ei	Ethernet I/O ASIC
xd	Xylogics disk (SMD) controller
xs	Zilog 8530 serial controller chip (keyboard and mouse)
zs	Serial communication ports: zs0 -> keyboard and mouse zs1 -> ttya & ttyb
sr	SCSI rimfire (made by Ciprico), no longer a supported hardware device

5.6 BOOT ROM Versions 3.2c (Series4) and 3.3 (Series5)

Data/stop bits change with these versions to be compatible with UNIX.
Is: 7data/2stop bits
Was (in previous versions): 8data/1stop bits

5.7 Init daemon and System Initialization Scripts

This is the last step in the boot process

1. **init** (8) runs **rc.boot** (8)
2. rc.boot sets the machine's name. Then, if the system is to come up multiuser, it invokes **fsck** (8) with the preen option (**-p**).
3. **fsck** checks the disks for inconsistencies.
4. If **fsck** does not report problems, **init** invokes **rc** (8). If **fsck** does detect a serious problem, init brings the system up single user. When you press Control-D to leave single user mode, **rc** (8) is invoked.
5. **rc** mounts file systems on the machine's local disks (4.2 mounts), if any. Then it passes control to **rc.local**.
6. **rc.local** starts daemons on the local machine that handle NFS, YP, and **mail** requests. It mounts file systems that the machine accesses over the network (NFS mounts). Finally, it returns control to **rc**.

7. rc starts up standard daemons, preserves editor files, and initiates other system activities, such as starting the network or system accounting, if applicable to the currently booting machine.
8. When rc finishes running, the system comes up in multiuser.

5.8 Field Notes

Section 6: Man Pages on Key System Administration Files

6.1 Introduction

The following key system administration man pages are given in this section:

NAME

ethers - Ethernet address to hostname database or YP domain

DESCRIPTION

The **ethers** file contains information regarding the known (48 bit) Ethernet addresses of hosts on the Internet. For each host on an Ethernet, a single line should be present with the following information:

Ethernet address
official host name

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is "*x:x:x:x:x:x*" where *x* is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than a SPACE, TAB, NEWLINE, or comment character. It is intended that host names in the **ethers** file correspond to the host names in the **hosts**(5) file.

The **ether_line**( ) routine from the Ethernet address manipulation library, **ethers**(3N) may be used to scan lines of the **ethers** file.

FILES

/etc/ethers

SEE ALSO

ethers(3N), **hosts**(5)

NAME

exports, xtab - directories to export to NFS clients

SYNOPSIS

/etc/exports

/etc/xtab

DESCRIPTION

The **/etc/exports** file contains entries for directories that can be exported to NFS clients. This file is read automatically by the **exportfs**(8) command. If you change this file, you must run **exportfs**(8) for the changes to affect the daemon's operation.

Only when this file is present at boot time does the **rc.local** script execute **exportfs**(8) and start the NFS file-system daemon, **nfsd**(8).

The **/etc/xtab** file contains entries for directories that are *currently* exported. This file should only be accessed by programs using **getexportent** (see **exportent**(3)). (Use the **-u** option of **exportfs** to remove entries from this file).

An entry for a directory consists of a line of the following form:

directory -option[,*option*]...

directory is the pathname of a directory (or file).

option is one of

ro Export the directory read-only. If not specified, the directory is exported read-write.

rw=*hostnames*[:*hostname*]...
Export the directory read-mostly. Read-mostly means read-only to most machines, but read-write to those specified. If not specified, the directory is exported read-write to all.

anon=*uid*
If a request comes from an unknown user, use *uid* as the effective user ID. Note: root users (uid 0) are always considered "unknown" by the NFS server, unless they are included in the "root" option below. The default value for this option is -2. Setting "anon" to -1 disables anonymous access. Note: by default secure NFS will accept insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to -1.

root=*hostnames*[:*hostname*]...
Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client* [:*client*]...
Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the netgroup database, and then the hosts database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

A '#' (pound-sign) anywhere in the file indicates a comment that extends to the end of the line.

EXAMPLE

```
/usr          -access=clients              # export to my clients
/usr/local    # export to the world
/usr2         -access=hermes:zip:tutorial  # export to only these machines
/usr/sun      -root=hermes:zip             # give root access only to these
/usr/new      -anon=0                      # give all machines root access
/usr/bin      -ro                          # export read-only to everyone
/usr/stuff    -access=zip,anon=-3,ro       # several options on one line
```

FILES

/etc/exports
/etc/xtab
/etc/hosts
/etc/netgroup
rc.local

SEE ALSO

exportent(3), **hosts**(5), **netgroup**(5), **exportfs**(8), **nfsd**(8)

WARNINGS

You cannot export either a parent directory or a subdirectory of an exported directory that is *within the same filesystem*. It would be illegal, for instance, to export both **/usr** and **/usr/local** if both directories resided on the same disk partition.

NAME

fstab, mtab - static filesystem mounting table, mounted filesystems table

SYNOPSIS

/etc/fstab

/etc/mtab

DESCRIPTION

The **/etc/fstab** file contains entries for filesystems and disk partitions to mount using the **mount**(8) command, which is normally invoked by the **rc.boot** script at boot time. This file is used by various utilities that mount, unmount, check the consistency of, dump, and restore file systems. It is also used by the system itself when locating the swap partition.

The **/etc/mtab** file contains entries for filesystems *currently* mounted, and is read by programs using the routines described in **getmntent**(3). **umount** (see **mount**(8)) removes entries from this file.

Each entry consists of a line of the form:

filesystem directory type options freq pass

filesystem is the pathname of a block-special device, or the name of a remote filesystem in *host*:*pathname* form.

directory is the pathname of the directory on which to mount the filesystem.

type is the filesystem type, which can be one of:

4.2	to mount a block-special device
nfs	to mount an exported NFS filesystem
swap	to indicate a swap partition
ignore	to have the **mount** command ignore the current entry (good for noting disk partitions that are not being used)

options contains a comma-separated list (no spaces) of mounting options, some of which can be applied to all types of filesystems, and others which only apply to specific types.

4.2 options:

quota | **noquota**
disk quotas are enforced or not enforced

nfs options:

bg	**fg**	If the first attempt fails, retry in the background, or, in the foreground
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
wsize=*n*	Set the write buffer size to *n* bytes.
timeo=*n*	Set the NFS timeout to *n* tenths of a second.
retrans=*n*	The number of NFS retransmissions.
port=*n*	The server IP port number.
soft	**hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.
secure	Use a more secure protocol for NFS transactions.
acregmin=*n*	Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n*	Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n*	Hold cached attributes for at least *n* seconds after

directory update.

acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update.

actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.

Common options:

ro | **rw** mount either read-only or read-write

suid | **nosuid**
setuid execution allowed or disallowed

grpid Create files with BSD semantics for propagation of the group ID. With this option, files inherit the group ID of the directory in which they are created, regardless of the directory's setgid bit.

noauto Do not mount this file system automatically (using **mount -a**).

freq is the interval (in days) between dumps.

pass is the **fsck**(8) pass in which to check the partition. Filesystems with the same pass number are checked simultaneously. Filesystems with *pass* equal to 0 are not checked.

A pound-sign (#) as the first non-white character indicates a comment line which is ignored by routines that read this file. The order of records in **/etc/fstab** is important because **fsck**, **mount**, and **umount** process the file sequentially; an entry for a file system must appear *after* the entry for any file system it is to be mounted on top of.

EXAMPLES

In this example, the **/home/user** directory is hard mounted read-write over the NFS, along with additional swap space in the form of a mounted swap file (see *Solbourne System and Network Administration* manual for details on adding swap space):

```
/dev/xy0a / 4.2 rw,noquota 1 1
/dev/xy0b /usr 4.2 rw,noquota 1 1
example:/home/user /home/user nfs rw,hard,fg 0 0
/export/swap/myswap swap swap rw 0 0
```

FILES

/etc/fstab
/etc/mtab

SEE ALSO

getmntent(3), **fsck**(8), **mount**(8), **quotacheck**(8), **quotaon**(8),

NAME

group - group file

SYNOPSIS

/etc/group

DESCRIPTION

The **group** file contains a one-line entry for each group recognized by the system, of the form:

groupname:password:gid:user-list

where:

groupname	is the name of the group.
gid	is the group's numerical ID within the system; it must be unique.
user-list	is a comma-separated list of users allowed in the group.

If the password field is empty, no password is demanded. The **group** file is an ASCII file. Because of the encrypted passwords, the **group** file can and does have general read permission, and can be used as a mapping of numerical group IDs to user names.

A group entry beginning with a '+' (plus sign), means to incorporate an entry or entries from the Yellow Pages. A '+' on a line by itself means to insert the entire contents of the Yellow Pages group file at that point in the file. An entry of the form: *'+groupname'* means to insert the entry (if any) for **groupname**. If a '+' entry has a non-empty *password* or *user-list* field, the contents of that field override the corresponding field from the Yellow Pages. The *gid* field cannot be overridden in this way.

An entry of the form: *-groupname* indicates that the group is disallowed. All subsequent entries for the indicated **groupname**, whether originating from the Yellow Pages, or the local **group** file, are ignored.

Malformed entries cause routines that read this file to halt, in which case group assignments specified further along are never made. To prevent this from happening, use **grpck**(8) to check the **/etc/group** database from time to time.

On all Solbourne systems, OS/MP uses group ID 0 as privilege to run **su**(1).

EXAMPLE

Here is a sample group file when the **group.adjunct** file does not exist:

```
primary:q.mJzTnu8icF.:10:fred,mary
+myproject:::bill,steve
+:
```

Here is a sample group file when the **group.adjunct** file does exist:

```
primary:#$primary:10:fred,mary
+myproject:::bill,steve
+:
```

If these entries appear at the end of a group file, then the group *primary* will have members **fred** and **mary**, and a group ID of **10**. The group *myproject* will have members **bill** and **steve**, and the password and group ID of the Yellow Pages entry for the group **myproject**. All groups listed in the Yellow Pages are pulled in and placed after the entry for **myproject**.

FILES

/etc/group

SEE ALSO

passwd(1), **su**(1), **getgroups**(2), **initgroups**(3), **crypt**(3), **group.adjunct**(5), **passwd**(5), **grpck**(8)

BUGS

The **passwd**(1) command will not change group passwords.

NAME

hosts - host name data base

SYNOPSIS

/etc/hosts

DESCRIPTION

The **hosts** file contains information regarding the known hosts on the DARPA Internet. For each host a single line should be present with the following information:

Internet address
official host name
aliases

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official host data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown hosts.

Network addresses are specified in the conventional '.' notation using the **inet_addr ()** routine from the Internet address manipulation library, **inet**(3N). Host names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

EXAMPLE

Here is a typical line from the **/etc/hosts** file:

```
192.9.1.20      gaia            # John Smith
```

FILES

/etc/hosts

SEE ALSO

gethostent(3N), **inet**(3N)

NAME

hosts.equiv, rhosts - trusted hosts by system and by user

DESCRIPTION

The **/etc/hosts.equiv** file contains a list of trusted hosts. When an **rlogin**(1C) or **rsh**(1C) request is received from a host listed in this file, and when the user making the request is listed in the **/etc/passwd** file, then the remote login is allowed with no further checking. In this case, **rlogin** does not prompt for a password, and commands submitted through **rsh** are executed. Thus, a remote user with a local user ID is said to have "equivalent" access from a remote host named in this file.

The format of the **hosts.equiv** file consists of a one-line entry for each host, of the form:

hostname [*username*]

The *hostname* field normally contains the name of a trusted host from which a remote login can be made. However, an entry consisting of a single '+' indicates that all known hosts are to be trusted. A hostname must be the "official" name as listed in the **hosts**(5) database. This is the first name given in the hosts database entry; hostname aliases are not recognized. Remote login access can also be given or denied for all hosts within a specific network group. An entry of the form:

+@group

means that all hosts in the named network group are trusted. An entry of the form:

-@group

means that all hosts in the group are not trusted; remote login access is denied to hosts in that group, except when an entry for a specific host appears ahead of the "minus" group entry.

The *username* field can be used to specify a user who is allowed to log in under any valid user ID. Careful thought about security should be given before providing this privilege to a user. You can also specify a network group in the *username* field with an entry of the form:

+@group1 +@group2

in which case any user in *group2* logging in from a host in *group1* may log in as anyone. Again, security is an important consideration here.

The User's .rhosts File

Whenever a remote login is attempted, the remote login daemon checks for a **.rhosts** file in the home directory of the user attempting to log in. A user's **.rhosts** file has the same format as the **hosts.equiv** file, and is used to give or deny access only for the *specific user* attempting to log in from a given host. While an entry in the **hosts.equiv** file allows remote login access to *any* user from the indicated host, an entry in a user's **.rhosts** file only allows access from a named host to the user in whose home directory the **.rhosts** file appears. (When this file is used, permissions in the user's home directory should allow read and search access by anyone, so it may be located and read.) When a user attempts a remote login, his **.rhosts** file is, in effect, prepended to the **hosts.equiv** file for permission checking. Thus, if a host is specified in the user's **.rhosts** file, login access is allowed, even if it would otherwise be excluded by a minus group entry in **/etc/hosts.equiv**.

The Root .rhosts File

When the user attempting a remote login is **root**, only the **/.rhosts** file is checked, not **/etc/hosts.equiv**.

FILES

/etc/hosts.equiv
/etc/passwd

~/.rhosts
/etc

SEE ALSO

rlogin(1C), **rsh**(1C), **hosts**(5), **netgroup**(5), **passwd**(5)

NAME

inetd.conf - Internet servers database

DESCRIPTION

The **inetd.conf** file contains the list of servers that **inetd**(8C) invokes when it receives an Internet request over a socket. Each server entry is composed of a single line of the form:

service-name socket-type protocol wait-status uid server-program server-arguments

Fields can be separated by either spaces or TAB characters. A '#' (pound-sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search this file.

service-name is the name of a valid service listed in the file **/etc/services**. For RPC services, the value of the *service-name* field consists of the RPC service name, followed by a slash and either a version number or a range of version numbers (for example, **mountd/1**).

socket-type can be one of:

stream	for a stream socket,
dgram	for a datagram socket,
raw	for a raw socket,
rdm	for a "reliably delivered message" socket, or
seqpacket	for a sequenced packet socket.

protocol must be a recognized protocol listed in the file **/etc/protocols**. For RPC services, the field consists of the string "rpc" followed by a slash and the name of the protocol (for example, **rpc/udp** for an RPC service using the UDP protocol as a transport mechanism).

wait-status is **nowait** for all but "single-threaded" datagram servers — servers which do not release the socket until a timeout occurs (such as **comsat**(8C) and **talkd**(8C)). These must have the status **wait**. Although **tftpd**(8C) establishes separate "pseudo-connections", its forking behavior can lead to a race condition unless it is also given the status **wait**.

uid is the user ID under which the server should run. This allows servers to run with access privileges other than those for root.

server-program is either the pathname of a server program to be invoked by **inetd** to perform the requested service, or the value **internal** if **inetd** itself provides the service.

server-arguments If a server must be invoked with command-line arguments, the entire command line (including argument 0) must appear in this field (which consists of all remaining words in the entry). If the server expects **inetd** to pass it the address of its peer (for compatibility with 4.2BSD executable daemons), then the first argument to the command should be specified as '**%A**'.

FILES

/etc/inetd.conf
/etc/services
/etc/protocols

SEE ALSO

services(5), **comsat**(8C), **inetd**(8C), **talkd**(8C), **tftpd**(8C)

NAME

networks - network name data base

DESCRIPTION

The **networks** file contains information regarding the known networks which comprise the DARPA Internet. For each network a single line should be present with the following information:

official network name
network number
aliases

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network number may be specified in the conventional '.' notation using the **inet_network ( )** routine from the Internet address manipulation library, **inet**(3N). Network names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

FILES

/etc/networks

SEE ALSO

getnetent(3N), **inet**(3N)

BUGS

A name server should be used instead of a static file. A binary indexed file format should be available for fast access.

NAME

netgroup - list of network groups

DESCRIPTION

netgroup defines network wide groups, used for permission checking when doing remote mounts, remote logins, and remote shells. For remote mounts, the information in **netgroup** is used to classify machines; for remote logins and remote shells, it is used to classify users. Each line of the **netgroup** file defines a group and has the format

groupname member1 member2 ....

where member*i* is either another group name, or a triple:

(hostname, username, domainname)

Any of these three fields can be empty, in which case it signifies a wild card. Thus

universal (,,)

defines a group to which everyone belongs.

A gateway machine should be listed under all possible hostnames by which it may be recognized:

wan (gateway,,) (gateway-ebb,,)

Field names that begin with something other than a letter, digit or underscore (such as '**-**') work in precisely the opposite fashion. For example, consider the following entries:

justmachines (analytica,-,sun)
justpeople (-,babbage,sun)

The machine **analytica** belongs to the group **justmachines** in the domain **sun**, but no users belong to it. Similarly, the user **babbage** belongs to the group **justpeople** in the domain **sun**, but no machines belong to it.

The *domainname* field refers to the domain *n* which the triple is valid, not the name containing the trusted host.

FILES

/etc/netgroup

SEE ALSO

getnetgrent(3N), **exports**(5), **makedbm**(8), **ypserv**(8)

NAME

passwd - password file

SYNOPSIS

/etc/passwd

DESCRIPTION

The **passwd** file contains basic information about each user's account. This file contains a one-line entry for each authorized user, of the form:

username:*password*:*uid*:*gid*:*gcos-field*:*home-dir*:*login-shell*

where

username	is the user's login name. This field contains no uppercase characters, and must not be more than eight characters in length.
password	is the user's encrypted password, or a string of the form: **##***name* if the encrypted password is in the **/etc/security/passwd.adjunct** file (see **passwd.adjunct**(5)). If this field is empty, **login**(1) does not request a password before logging the user in.
uid	is the user's numerical ID for the system, which must be unique. *uid* is generally a value between 0 and 32767.
gid	is the numerical ID of the group that the user belongs to. *gid* is generally a value between 0 an 32767.
gcos-field	is the user's real name, along with information to pass along in a mail-message heading. It is called the gcos-field for historical reasons. A **&** in this field stands for the login name (in cases where the login name appears in a user's real name).
home-dir	is the pathname to the directory in which the user is initially positioned upon logging in.
login-shell	is the user's initial shell program. If this field is empty, the default shell is **/usr/bin/sh**.

The **passwd** file can also have lines beginning with a '**+**' (plus sign) which means to incorporate entries from the Yellow Pages. There are three styles of + entries in this file: by itself, + means to insert the entire contents of the Yellow Pages password file at that point; +*name* means to insert the entry (if any) for *name* from the Yellow Pages at that point; +**@***netgroup* means to insert the entries for all members of the network group **netgroup** at that point. If a +*name* entry has a non-NULL *password*, *gcos*, *home-dir*, or *login-shell* field, the value of that field overrides what is contained in the Yellow Pages. The *uid* and *gid* fields cannot be overridden.

The **passwd** file can also have lines beginning with a '**-**' (minus sign) which means to disallow entries from the Yellow Pages. There are two styles of '**-**' entries in this file: -*name* means to disallow any subsequent entries (if any) for *name* (in this file or in the Yellow Pages); -**@***netgroup* means to disallow any subsequent entries for all members of the network group *netgroup*.

The password file is an ASCII file that resides in the **/etc** directory. Because the encrypted passwords on a secure system are kept in the **passwd.adjunct** file, **/etc/passwd** has general read permission on all systems, and can be used by routines that map numerical user IDs to names.

Appropriate precautions must be taken to lock the **/etc/passwd** file against simultaneous changes if it is to be edited with a text editor; **vipw**(8) does the necessary locking.

EXAMPLE

Here is a sample **passwd** file when **passwd.adjunct** does not exist:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
fred:6k/7KCFRPNVXg:508:10:% Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
```

Here is a sample **passwd** file when **passwd.adjunct** does exist:

```
root:##root:0:10:God:/:/bin/csh
fred:##fred:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+::::Guest
```

In this example, there are specific entries for users **root** and **fred**, to assure that they can log in even when the system is running standalone. The user **john** will have his password entry in the Yellow Pages incorporated without change; anyone in the netgroup **documentation** will have their password field disabled, and anyone else will be able to log in with their usual password, shell, and home directory, but with a gcos-field of **Guest**.

FILES

/etc/passwd
/etc/security/passwd.adjunct

SEE ALSO

login(1), **mail**(1), **passwd**(1), **crypt**(3), **getpwent**(3), **group**(5), **passwd.adjunct**(5), **adduser**(8), **sendmail**(8), **vipw**(8)

BUGS

mail(1) and **sendmail**(8) use the gcos-field to compose the **From:** line for addressing mail messages, but these programs get confused by nested parentheses when composing replies. This problem can be avoided by using different types of brackets within the gcos-field; for example:

(& Fredricks [Podunk U <EE/CIS>] {818}-555-5555)

NAME

printcap - printer capability data base

SYNOPSIS

/etc/printcap

DESCRIPTION

printcap is a simplified version of the **termcap**(5) data base for describing printers. The spooling system accesses the **printcap** file every time it is used, allowing dynamic addition and deletion of printers. Each entry in the data base describes one printer. This data base may not be substituted for, as is possible for **termcap**, because it may allow accounting to be bypassed.

The default printer is normally **lp**, though the environment variable PRINTER may be used to override this. Each spooling utility supports a **-P***printer* option to explicitly name a destination printer.

Refer to *Solbourne System and Network Administration* manual for a discussion of how to set up the database for a given printer.

Each entry in the **printcap** file describes a printer, and is a line consisting of a number of fields separated by ':' characters. The first entry for each printer gives the names which are known for the printer, separated by '|' characters. The first name is conventionally a number. The second name given is the most common abbreviation for the printer, and the last name given should be a long name fully identifying the printer. The second name should contain no blanks; the last name may well contain blanks for readability. Entries may continue onto multiple lines by giving a '\' as the last character of a line, and empty fields may be included for readability.

Capabilities in **printcap** are all introduced by two-character codes, and are of three types:

Boolean Capabilities that indicate that the printer has some particular feature. Boolean capabilities are simply written between the ':' characters, and are indicated by the word 'bool' in the **type** column of the capabilities table below.

Numeric Capabilities that supply information such as baud-rates, number of lines per page, and so on. Numeric capabilities are indicated by the word **num** in the **type** column of the capabilities table below. Numeric capabilities are given by the two-character capability code followed by the '#' character, followed by the numeric value. For example:

:br#1200:

is a numeric entry stating that this printer should run at 1200 baud.

String Capabilities that give a sequence which can be used to perform particular printer operations such as cursor motion. String valued capabilities are indicated by the word **str** in the **type** column of the capabilities table below. String valued capabilities are given by the two-character capability code followed by an '=' sign and then a string ending at the next following ':'. For example,

:rp=spinwriter:

is a sample entry stating that the remote printer is named **spinwriter**.

CAPABILITIES

Name *Type* *Default* *Description*

af	str	NULL	name of accounting file
br	num	none	if lp is a tty, set the baud rate (ioctl call)
cf	str	NULL	cifplot data filter
df	str	NULL	TeX data filter (DVI format)
du	str	0	User ID of user 'daemon'.
fc	num	0	if lp is a tty, clear flag bits
ff	str	"\f"	string to send for a form feed
fo	bool	false	print a form feed when device is opened
fs	num	0	like 'fc' but set bits
gf	str	NULL	graph data filter (plot(3X) format)
hl	bool	false	print the burst header page last
ic	bool	false	driver supports (non standard) ioctl to indent printout
if	str	NULL	name of text filter which does accounting
lf	str	"/dev/console"	error logging file name
lo	str	"lock"	name of lock file
lp	str	"/dev/lp"	device name to open for output
mc	num	0	maximum number of copies
ms	str	NULL	list of terminal modes to set or clear
mx	num	1000	maximum file size (in BUFSIZ blocks), zero = unlimited
nd	str	NULL	next directory for list of queues (unimplemented)
nf	str	NULL	ditroff data filter (device independent troff)
of	str	NULL	name of output filtering program
pc	num	200	price per foot or page in hundredths of cents
pl	num	66	page length (in lines)
pw	num	132	page width (in characters)
px	num	0	page width in pixels (horizontal)
py	num	0	page length in pixels (vertical)
rf	str	NULL	filter for printing FORTRAN style text files
rg	str	NULL	restricted group. Only members of group allowed access
rm	str	NULL	machine name for remote printer
rp	str	"lp"	remote printer name argument
rs	bool	false	restrict remote users to those with local accounts
rw	bool	false	open printer device read/write instead of write-only
sb	bool	false	short banner (one line only)
sc	bool	false	suppress multiple copies
sd	str	"/var/spool/lpd"	spool directory
sf	bool	false	suppress form feeds
sh	bool	false	suppress printing of burst page header
st	str	"status"	status file name
tc	str	NULL	name of similar printer; must be last
tf	str	NULL	troff data filter (C/A/T phototypesetter)
tr	str	NULL	trailer string to print when queue empties
vf	str	NULL	raster image filter
xc	num	0	if lp is a tty, clear local mode bits
xs	num	0	like 'xc' but set bits

If the local line printer driver supports indentation, the daemon must understand how to invoke it.

Note: the **fs**, **fc**, **xs**, and **xc** fields are flag *masks* rather than flag *values*. Certain default device flags are set when the device is opened by the line printer daemon if the device is connected to a terminal port. The flags indicated in the **fc** field are then cleared; the flags in the **fs** field are then set (or vice-versa, depending on the order of **fc***#nnnn* and **fs***#nnnn* in the **/etc/printcap** file). The bits cleared by the **fc** field and set by the **fs** field are those in the **sg_flags** field of the **sgtty** structure, as set by the **TIOCSETP ioctl** call, and the bits

cleared by the **xc** field and set by the **xs** field are those in the "local flags" word, as set by the **TIOCLSET ioctl** call. See **ttcompat**(4M) for a description of these flags. For example, to set exactly the flags 06300 in the **fs** field, which specifies that the **EVENP**, **ODDP**, and **XTABS** modes are to be set, and all other flags are to be cleared, do:

:fc#0177777:fs#06300:

The same process applies to the **xc** and **xs** fields. Alternatively, the **ms** field can be used to specify modes to be set and cleared. These modes are specified as **stty**(1V) modes; any mode supported by **stty** may be specified, except for the baud rate which must be specified with the **br** field. This permits modes not supported by the older terminal interface described in **ttcompat**(4M) to be set or cleared. Thus, to set the terminal port to which the printer is attached to even parity, tab expansion, no newline to carriage-return/line-feed translation, and RTS/CTS flow control enabled, do:

:ms=evenp,-tabs,nl,crtscts:

FILES

/etc/printcap

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **stty**(1V), **plot**(3X), **ttcompat**(4M), **termcap**(5), **lpc**(8), **lpd**(8), **pac**(8)

Solbourne System and Network Administration manual

NAME

rpc - rpc program number data base

SYNOPSIS

/etc/rpc

DESCRIPTION

The *rpc* file contains user readable names that can be used in place of rpc program numbers. Each line has the following information:

name of server for the rpc program
rpc program number
aliases

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Here is an example of the **/etc/rpc** file from the OS/MP System.

```
#
#              rpc  1.10  87/04/10
#
portmapper     100000        portmap sunrpc
rstatd                       100001        rstat rup perfmeter
rusersd                      100002        rusers
nfs                          100003        nfsprog
ypserv                       100004        ypprog
mountd                       100005        mount showmount
ypbind                       100007
walld                        100008        rwall shutdown
yppasswdd      100009        yppasswd
etherstatd     100010        etherstat
rquotad                      100011        rquotaprog quota rquota
sprayd                       100012        spray
3270_mapper    100013
rje_mapper     100014
selection_svc  100015        selnsvc
database_svc   100016
rexd                         100017        rex
alis                         100018
sched                        100019
llockmgr       100020
nlockmgr       100021
x25.inr                      100022
statmon                      100023
status                       100024
bootparam      100026
ypupdated      100028        ypupdate
keyserv                      100029        keyserver
```

FILES

/etc/rpc

SEE ALSO

getrpcent(3N)

NAME

services - Internet services and aliases

DESCRIPTION

The **services** file contains an entry for each service available through the DARPA Internet. Each entry consists of a line of the form:

service-name *port*/protocol aliases

service-name	This is the official Internet service name.
port/*protocol*	This field is composed of the port number and protocol through which the service is provided (for instance, **512/tcp**).
aliases	This is a list of alternate names by which the service might be requested.

Fields can be separated by any number of spaces or TAB's. A '#' (pound-sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

FILES

/etc/services

SEE ALSO

getservent(3N), **inetd.conf**(5)

BUGS

A name server should be used instead of a static file.

NAME

termcap - terminal capability data base

DESCRIPTION

termcap is a data base describing the capabilities of terminals. Terminals are described in **termcap** source descriptions by giving a set of capabilities which they have, by describing how operations are performed, by describing padding requirements, and by specifying initialization sequences. This database is used by applications programs such as **vi**(1), and libraries such as **curses**(3X), so they can work with a variety of terminals without changes to the programs.

Each **termcap** entry consist of a number of colon-separated (:) fields. The first field for each terminal lists the various names by which it is known, separated by bar (|) characters. The first name is always two characters long, and is used by older (version 6) systems (which store the terminal type in a 16-bit word in a system-wide database). The second name given is the most common abbreviation for the terminal (this is the one to which the environment variable **TERM** would normally be set). The last name should fully identify the terminal's make and model. All other names are taken as synonyms for the initial terminal name. All names but the first and last should be in lower case and contain no blanks; the last name may well contain upper case and blanks for added readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions:

- The particular piece of hardware making up the terminal should have a root name chosen; for example, for the Hewlett-Packard 2621, **hp2621**. This name should not contain hyphens.
- Modes that the hardware can be in or user preferences should be indicated by appending a hyphen and an indicator of the mode. Thus, a **vt100** in 132-column mode would be given as: **vt100-w**. The following suffixes should be used where possible:

Suffix	*Meaning*	*Example*
-w	wide mode (more than 80 columns)	**vt100-w**
-am	with automatic margins (usually default)	**vt100-am**
-nam	without automatic margins	**vt100-nam**
-n	number of lines on the screen	**aaa-60**
-na	no arrow keys (leave them in local)	**concept100-na**
*-n*p	number of pages of memory	**concept100-4p**
-rv	reverse video	**concept100-rv**

Terminal entries may continue onto multiple lines by giving a \ as the last character of a line, and empty fields may be included for readability (here between the last field on a line and the first field on the next). Comments may be included on lines beginning with #.

ENVIRONMENT

If the environment variable **TERMCAP** contains an absolute pathname, programs look to that file for terminal descriptions, rather than **/usr/share/lib/termcap**. If the value of this varible is in the form of a **termcap** entry, programs use that value for the terminal description.

FILES

/usr/share/lib/termcap
file containing terminal descriptions

SEE ALSO

ex(1), **more**(1), **tset**(1), **ul**(1), **vi**(1), **curses**(3X), **printf**(3S), **termcap**(3X), **term**(5V), **terminfo**(5V)

Solbourne System and Network Admininration

CAVEATS AND BUGS

UNIX System V uses **terminfo**(5V) rather than **termcap**. OS/MP supports either **termcap** or **terminfo**(5V) terminal databases, depending on whether you link with the **termcap**(3X) or **curses**(3V) libraries. Transitions between the two should be relatively painless if capabilities flagged as "obsolete" are avoided.

vi allows only 256 characters for string capabilities, and the routines in **termcap**(3X) do not check for overflow of this buffer. The total length of a single entry (excluding only escaped NEWLINE characters) may not exceed 1024.

Not all programs support all entries.

NAME

ttytab, ttys - terminal initialization data

DESCRIPTION

The **/etc/ttytab** file contains information that is used by various routines to initialize and control the use of terminal special files. This information is read with the **getttyent**(3) library routines. There is one line in **/etc/ttytab** file per special file.

The **/etc/ttys** file should not be edited; it is derived from **/etc/ttytab** by **init(8)** at boot time, and is only included for backward compatibility with programs that may still require it.

Fields are separated by TAB and/or SPACE characters. Some fields may contain more than one word and should be enclosed in double quotes. Blank lines and comments can appear anywhere in the file; comments are delimited by '#' and NEWLINE. Unspecified fields default to NULL. The first field is the terminal's entry in the device directory, **/dev**. The second field of the file is the command to execute for the line, typically **getty**(8), which performs such tasks as baud-rate recognition, reading the login name, and calling **login**(1). It can be, however, any desired command, for example the start up for a window system terminal emulator or some other daemon process, and can contain multiple words if quoted. The third field is the type of terminal normally connected to that tty line, as found in the **termcap**(5) data base file. The remaining fields set flags in the **ty_status** entry (see **getttyent**(3)) or specify a window system process that **init**(8) will maintain for the terminal line.

As flag values, the strings **on** and **off** specify whether **init** should execute the command given in the second field, while **secure** in addition to **on** allows "root" to login on this line. If the console is not marked "secure," the system prompts for the root password before coming up in single-user mode. These flag fields should not be quoted. The string **window=** is followed by a quoted command string which **init** will execute before starting **getty**. If the line ends in a comment, the comment is included in the **ty_comment** field of the **ttyent** structure.

EXAMPLE

```
console "/usr/etc/getty std.1200"  vt100      on secure
ttyd0   "/usr/etc/getty d1200"     dialup     on        # 555-1234
ttyh0   "/usr/etc/getty std.9600"  hp2621-nl  on        # 254MC
ttyh1   "/usr/etc/getty std.9600"  plugboard  on        # John's office
ttyp0   none                       network
ttyp1   none                       network    off
ttyv0   "/usr/new/xterm -L :0"     vs100      on window="/usr/new/Xvs100 0"
```

The first line permits "root" login on the console at 1200 baud, and indicates that the console is secure for single-user operation. The second example allows dialup at 1200 baud without "root" login, and the third and fourth examples allow login at 9600 baud with terminal types of **hp2621-nl** and **plugboard**, respectively. The fifth and sixth lines are examples of network pseudo-ttys, for which **getty** should not be enabled. The last line shows a terminal emulator and window-system startup entry.

FILES

/dev
/etc/ttytab

SEE ALSO

login(1), **getttyent**(3), **gettytab**(5), **termcap**(5), **getty**(8), **init**(8)

Section 7: Man Pages on Network Status Tools

7.1 Introduction

This section offers the following man pages on system administration commands and network status tools:

NAME

config - build system configuration files

SYNOPSIS

/etc/config [**-fgnp**] [**-o** *obj_dir*] *config_file*

DESCRIPTION

config does the preparation necessary for building a new system kernel with **make**(1). The *config_file* named on the command line describes the kernel to be made in terms of options you want in your system, size of tables, and device drivers to be included. When you run **config**, it uses several input files located in the current directory (typically the **conf** subdirectory of the system source including your *config_file*). The format of this file is described below.

If the directory named *../config_file* does not exist, **config** will create one. One of **config**'s output files is a makefile which you use with **make**(1) to build your system.

You use **config** as follows. Run **config** from the **conf** subdirectory of the system source (in a typical Solbourne environment, from **/usr/sys/kap0/conf**):

```
example# /usr/etc/config config_file
Doing a "make depend"
example# cd ../config_file
example# make
...lots of output...
```

While **config** is running watch for any errors. Never use a kernel which **config** has complained about; the results are unpredictable. If **config** completes successfully, you can change directory to the *../config_file* directory, where it has placed the new makefile, and use **make** to build a kernel. The output files placed in this directory include **ioconf.c**, which contains a description of I/O devices attached to the system; a makefile, which is used by **make** to build the system; a set of header files (*device_name*.**h**) which contain the number of various devices that may be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for the root file system, swapping, and system dumps.

Now you can install your new kernel and try it out.

OPTIONS

-f Set up the makefile for fast builds. This is done by building a **vmunix.o** file which includes all the **.o** files which have no source. This reduces the number of files which have to be **stat**ed during a system build. This is done by prelinking all the files for which no source exists into another file which is then linked in place of all these files when the kernel is made. This makefile is faster because it does not **stat** the object files during the build.

-g Get the current version of a missing source file from its SCCS history, if possible.

-n Do not do the **'make depend'**. Normally **config** will do the **'make depend'** automatically. If this option is used **config** will print **'Don't forget to do a "make depend"'** before completing as a reminder.

-p Configure the system for profiling (see **kgmon**(8) and **gprof**(1)).

-o *obj_dir*

Use *../obj_dir* instead of **../OBJ** as the directory to find the object files when the corresponding source file is not present in order to generate the files necessary to compile and link your kernel.

NAME

df - report free disk space on file systems

SYNOPSIS

df [**-a**] [**-i**] [**-t** *type*] [*filesystem*...] [*filename*...]

DESCRIPTION

df displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. Used without arguments, **df** reports on all mounted file systems, producing something like:

```
tutorial% df
Filesystem  kbytes  used   avail  capacity  Mounted on
/dev/ip0a    7445   4714   1986   70%       /
/dev/ip0g   42277  35291   2758   93%       /usr
```

Note that used+avail is less than the amount of space in the file system (kbytes); this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using **tunefs**(8). When all the space on a file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When a file system is overallocated in this way, **df** may report that the file system is more than 100% utilized.

If arguments to **df** are disk partitions (for example, **/dev/ip0a**s or path names, **df** produces a report on the file system containing the named file. Thus **df .** shows the amount of space on the file system containing the current directory.

OPTIONS

-a Reports on all filesystems including the uninteresting ones which have zero total blocks. (e.g. *automounter*)

-i Report the number of used and free inodes.

-t *type* Report on filesystems of a given *type* (for example, **nfs** or **4.2**).

FILES

/etc/mtab List of filesystems currently mounted.

SEE ALSO

du(1V), **mtab**(5), **quot**(8), **tunefs**(8)

NAME

dkinfo - report information about a disk's geometry and partitioning

SYNOPSIS

/usr/etc/dkinfo *disk* [*partition*]

DESCRIPTION

dkinfo gives the total number of cylinders, heads, and sectors or tracks on the specified *disk*, and gives this information along with the starting cylinder for the specified *partition*. If no *partition* is specified on the command line, **dkinfo** reports on all partitions.

The *disk* specification here is a disk name of the form *xxn*, where *xx* is the controller device abbreviation (ip, xy, etc.) and *n* is the disk number. The *partition* specification is simply the letter used to identify that partition in the standard UNIX system nomenclature. For example, '**/usr/etc/dkinfo xy0**' reports on the first disk in a system controlled by a Xylogics controller; '**/usr/etc/dkinfo xy0g**' reports on the seventh partition of such a disk.

EXAMPLE

A request for information on my local disk, an 84 MByte disk controlled by a Xylogics 450 controller, might look like this:

```
#/usr/etc/dkinfo xy0
xy0: Xylogics 450 controller at addr ee40, unit # 0
586  cylinders 7 heads 32 sectors/track
a: 15884 sectors (70 cyls, 6 tracks, 12 sectors)
starting cylinder 0
b: 33440 sectors (149 cyls, 2 tracks)
starting cylinder 71
c: 131264 sectors (586 cyls)
starting cylinder 0
d: No such device or address
e: No such device or address
f: No such device or address
g: 81760 sectors (365 cyls)
starting cylinder 221
h: No such device or address
#
```

FILES

/dev/*rxxnp*

SEE ALSO

dkio(4S), **format**(8)

NAME

du - display the number of disk blocks used per directory or file

SYNOPSIS

du [**-s**] [**-a**] [*filename* ...]

SYSTEM V SYNOPSIS

du [**-s**] [**-a**] [**-r**] [*filename* ...]

DESCRIPTION

du gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file *filename*. If *filename* is missing, '.' (the current directory) is used.

A file which has multiple links to it is only counted once.

SYSTEM V DESCRIPTION

The System V version of **du** gives the number of 512-byte blocks rather than the number of kilobytes.

OPTIONS

-s Only display the grand total for each of the specified *filenames*.

-a Generate an entry for each file.

Entries are generated only for each directory in the absence of options.

SYSTEM V OPTIONS

-r The System V version of **du** is normally silent about directories that cannot be read, files that cannot be opened, etc. The **-r** option will cause **du** to generate messages in such instances.

EXAMPLE

Here is an example of using **du** in a directory. We used the **pwd**(1) command to identify the directory, then used **du** to show the usage of all the subdirectories in that directory. The grand total for the directory is the last entry in the display:

```
% pwd
/usr/ralph/misc
% du
5       ./jokes
33      ./squash
44      ./tech.papers/lpr.document
217     ./tech.papers/new.manager
401     ./tech.papers
144     ./memos
80      ./letters
388     ./window
93      ./messages
15      ./useful.news
1211    .
%
```

SEE ALSO

df(1), **pwd**(1), **quot**(8)

BUGS

Filename arguments that are not directory names are ignored, unless you use **-a** .
If there are too many distinct linked files, **du** will count the excess files more than once.

NAME

format - disk partitioning and maintenance utility

SYNOPSIS

format [**-f** *command-file*] [**-l** *log-file*] [**-x** *data-file*] [**-d** *disk-name*] [**-t** *disk_type*] [**-p** *partition-name*] [**-s**] *diskname*...

DESCRIPTION

format enables you to format, label, repair and analyze disks on your Solbourne computer. Unlike previous disk maintenance programs, **format** runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, **format** is also supported within the memory-resident system environment. For most applications, however, running **format** under SunOS is the more convenient approach.

If no *disk-list* is present, **format** uses the disk list defined in the data file specified with the **-x** option. If that option is omitted, the data file defaults to **format.dat** in the current directory, or else **/etc/format.dat**.

OPTIONS

-f *command-file*
: Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, **format** does not issue **continue?** prompts.

-l *log-file*
: Log a transcript of the **format** session to the indicated *log-file*, including the standard input, the standard output and the standard error.

-x *data-file*
: Use the disk list contained in *data-file*.

-d *disk_name*
: Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, - xy0). This can also be accomplished by specifying a single disk in the disk list.

-t *disk-type*
: Specify the type of disk which is current upon entry into the program, A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.

-p *partition-name*
: Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being made current, and its type is either specified or available from the disk label.

-s Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the **-f** option.

FILES

/etc/format.dat default data file

NAME

fsck - file system consistency check and interactive repair

SYNOPSIS

/usr/etc/fsck -p [*filesystem* ...]

/usr/etc/fsck [**-b** block#] [**-w**] [**-y**] [**-n**] [*filesystem*] ...

DESCRIPTION

The first form of *fsck* preens a standard set of file systems or the specified file systems. It is normally used in the **/etc/rc** script during automatic reboot. In this case, *fsck* reads the table **/etc/fstab** to determine the file systems to check. It inspects disks in parallel, taking maximum advantage of I/O overlap to check the file systems as quickly as possible.

Normally, the root file system is checked in pass 1; other root-partition file systems are checked in pass 2. Small file systems on separate partitions are checked in pass 3, while larger ones are checked in passes 4 and 5.

Only partitions marked in **/etc/fstab** with a file system type of "4.2" and a non-zero pass number are checked.

fsck corrects innocuous inconsistencies such as: unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block, automatically. It displays a message for each inconsistency corrected that identifies the nature of, and file system on which, the correction is to take place. After successfully correcting a file system, *fsck* prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

If *fsck* encounters other inconsistencies that it cannot fix automatically, it exits with an abnormal return status (and the reboot fails).

If sent a QUIT signal, *fsck* will finish the file system checks, then exit with an abnormal return status that causes the automatic reboot to fail. This is useful when you wish to finish the file system checks, but do not want the machine to come up multiuser.

Without the **-p** option, *fsck* audits and interactively repairs inconsistent conditions on file systems. In this case, it asks for confirmation before attempting any corrections. Inconsistencies other than those mentioned above can often result in some loss of data. The amount and severity of data lost can be determined from the diagnostic output.

The default action for each correction is to wait for the operator to respond either **yes** or **no**. If the operator does not have write permission on the file system, *fsck* will default to a **-n** (no corrections) action.

If no file systems are given to *fsck* then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Incorrect directory sizes.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks, file pointing to unallocated inode, inode number out of range.
8. Super Block checks: more blocks for inodes than there are in the file system.

9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the **lost+found** directory does not exist, it is created. If there is insufficient space its size is increased.

A file system may be specified by giving the name of the cooked or raw device on which it resides, or by giving the name of its mount point. If the latter is given, *fsck* finds the name of the device on which the file system resides by looking in **/etc/fstab**.

Checking the raw device is almost always faster.

OPTIONS

-b Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.

-w Check writable file systems only.

-y Assume a **yes** response to all questions asked by *fsck*; this should be used with extreme caution, as it is a free license to continue, even after severe problems are encountered.

-n Assume a **no** response to all questions asked by *fsck*; do not open the file system for writing.

FILES

/etc/fstab contains default list of file systems to check

DIAGNOSTICS

The diagnostics produced by *fsck* are fully enumerated and explained in *System and Network Administration* .

EXIT STATUS

0 Either no errors detected or all errors were corrected.

4 Root file system errors were corrected. The system must be rebooted.

8 Some uncorrected errors exist on one or more of the file systems checked, there was a syntax error, or some other operational error occurred.

12 A signal was caught during processing.

SEE ALSO

fstab(5), fs(5), newfs(8), mkfs(8), crash(8S), reboot(8)

System and Network Administration

BUGS

There should be some way to start a **fsck -p** at pass *n*.

NAME

mount, umount - mount and dismount filesystems

SYNOPSIS

/usr/etc/mount [**-p**]
/usr/etc/mount -a[**fnv**] [**-t** *type*]
/usr/etc/mount [**-fnrv**] [**-t** *type*] [**-o** *options*] *filesystem directory*
/usr/etc/mount [**-vfn**] [**-o** *options*] *filesystem* | *directory*

/usr/etc/umount [**-t** *type*] [**-h** *host*]
/usr/etc/umount -a[**v**]
/usr/etc/umount [**-v**] *filesystem* | *directory* ...

DESCRIPTION

mount attaches a named *filesystem* to the filesystem hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host*:*pathname*, it is assumed to be an NFS filesystem (type **nfs**).

umount unmounts a currently mounted filesystem, which can be specified either as a *directory* or a *filesystem*.

mount and **umount** maintain a table of mounted filesystems in **/etc/mtab**, described in **fstab**(5). If invoked without an argument, **mount** displays the contents of this table. If invoked with either a *filesystem* or *directory* only, mount searches the file **/etc/fstab** for a matching entry, and mounts the filesystem indicated in that entry on the indicated directory.

MOUNT OPTIONS

-p Print the list of mounted filesystems in a format suitable for use in **/etc/fstab**.

-a All. Attempt to mount all the filesystems described in **/etc/fstab**. If a *type* argument is specified with **-t**, mount all filesystems of that type. Using **-a**, **mount** builds a dependency tree of mount points in **/etc/fstab**. **mount** will correctly mount these filesystems regardless of their order in **/etc/fstab** (except loopback mounts; see WARNINGS below).

-f Fake an **/etc/mtab** entry, but do not actually mount any filesystems.

-n Mount the filesystem without making an entry in **/etc/mtab**.

-v Verbose. Display a message indicating each filesystem being mounted.

-t *type* Specify a filesystem type. The accepted types are **4.2**, **nfs**, and **lo**. see **fstab**(5) for a description of **4.2**, and **nfs**; see **lofs**(4S) for a description of **lo**.

-r Mount the specified filesystem read-only, even if the entry in **/etc/fstab** specifies that it is to be mounted read-write.

Physically write-protected and magnetic-tape filesystems must be mounted read-only. Otherwise errors occur when the system attempts to update access times, even if no write operation is attempted.

-o *options*

Specify filesystem *options* —list of comma-separated words from the list below. Some options are valid for all filesystem types, while others apply to a specific type only.

options valid on *all* filesystems:

rw | **ro**	Read/write or read-only.
suid | **nosuid**	Setuid execution allowed or disallowed.
grpid	Create files with BSD semantics for the propagation of

the group ID. Under this option, files inherit the GID of the directory in which they are created, regardless of the directory's set-GID bit.

noauto Do not mount this filesystem that is currently mounted read-only. If the filesystem is not currently mounted, an error results.

remount If the file system is currently mounted, and if the entry in **/etc/fstab** specifies that it is to be mounted read-write or **rw** was specified along with **remount**, remount the file system making it read-write. If the entry in **/etc/fstab** specifies that it is to be mounted read-only and **rw** was not specified, the file system is not remounted. If the file system is currently mounted read-write, specifying **ro** along with **remount** results in an error. If the file system is not currently mounted, an error results.

The default is '**rw,suid**'.

options specific to **4.2** filesystems:

quota | noquota
Usage limits are enforced, or are not enforced. The default is **noquota.**

options specific to **nfs** (NFS)filesystems:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.

retry=*n* The number of times to retry the mount operation.

rsize=*n* Set the read buffer size to *n* bytes.

wsize=*n* Set the write buffer size to *n* bytes.

timeo=*n* Set the NFS timeout to *n* tenths of a second.

retrans=*n* The number of NFS retransmissions.

port=*n* The server IP port number.

soft | hard Return an error if the server does not respond, or continue the retry request until the server responds.

intr Allow keyboard interrupts on hard mounts.

secure Use a more secure protocol for NFS transactions.

acregmin=*n* Hold cached attributes for at least *n* seconds after file modification.

acregmax=*n* Hold cached attributes for no more than *n* seconds after file modification.

acdirmin=*n* Hold cached attributes for at least *n* seconds after directory update.

acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update.

actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.

noac Suppress attribute caching.

Regular defaults are:

```
fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,\
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
```

actimeo has no default; it sets **acregmin, acregmax, acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

UMOUNT OPTIONS

-h *host* Unmount all filesystems listed in **/etc/mtab** that are remote-mounted from *host*.

-t *type* Unmount all filesystems listed in **/etc/mtab** that are of a given *type*.

-a Unmount all filesystems currently mounted (as listed in **/etc/mtab**).

-v Verbose. Display a message indicating each filesystem being unmounted.

NFS FILESYSTEMS

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (**mountd**(8c)) does not respond. **mount** retries the request up to the count specified in the **retry=***n* option. Once the filesystem is mounted, each NFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a filesystem mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Read-Write vs. Read-Only

Filesystems that are mounted **rw** (read-write) should use the **hard** option.

Interrupting Processes With Pending NFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted filesystem.

Secure Filesystems

The **secure** option must be given if the server requires secure mounting for the filesystem.

File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* extends flush time by *n* seconds for both regular files and directories.

SYSTEM V COMPATIBILITY

System V File-Creation Semantics

Ordinarily, when a file is created its GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis, by setting the set-GID bit of the parent directory; in this case, the GID is set to the GID of the parent directory (see **open**(2V) and **mkdir**(2)). Files created on filesystems that are mounted with the **grpid** option will obey BSD semantics; that is, the GID is unconditionally inherited from that of the parent directory.

EXAMPLES

To mount a local disk:	**mount /dev/xy0g /usr**
To fake an entry for **nd** root:	**mount -ft 4.2 /dev/nd0 /**
To mount all 4.2 filesystems:	**mount -at 4.2**
To mount a remote filesystem:	**mount -t nfs serv:/usr/src /usr/src**
To mount a remote filesystem:	**mount serv:/usr/src /usr/src**
To hard mount a remote filesystem:	**mount -o hard serv:/usr/src /usr/src**
To save current mount state:	**mount -p > /etc/fstab**

FILES

/etc/mtab	table of mounted filesystems
/etc/fstab	table of filesystems mounted at boot

WARNINGS

mount does not understand the mount order dependencies involved in loopback mounting. Loopback mounts may be dependent on two mounts having been previously performed, while **nfs** and **4.2** mounts are dependent only on a single previous mount. As a rule of thumb, place loopback mounts at the end of **/etc/fstab**file. See **lofs**(4S) for a complete description.

SEE ALSO

mkdir(2), **mount**(2), **unmount**(2), **open**(2V), **fstab**(5), **mtab**(5), **mountd**(8C), **nfsd**(8)

BUGS

Mounting filesystems full of garbage crashes the system.

If the directory on which a filesystem is to be mounted is a symbolic link, the filesystem is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself.

NAME

ncheck - generate names from i-numbers

SYNOPSIS

/usr/etc/ncheck [-i *numbers* **] [-as] [** *filesystem* **]**

DESCRIPTION

Note: For most normal file system maintenance, the function of **ncheck** is subsumed by **fsck**(8).

ncheck with no argument generates a pathname versus i-number list of all files on a set of default file systems. Names of directory files are followed by '/'.

A file system may be specified by the optional *filesystem* argument.

The report is in no useful order, and probably should be sorted.

OPTIONS

-i *numbers*
Report only those files whose i-*numbers* follow.

-a Print the names '.' and '..', which are ordinarily suppressed.

-s Report only special files and files with set-user-ID mode. This is intended to discover concealed violations of security policy.

SEE ALSO

sort(1V), **dcheck**(8), **fsck**(8), **icheck**(8)

DIAGNOSTICS

When the filesystem structure is improper, '??' denotes the "parent" of a parentless file and a pathname beginning with '...' denotes a loop.

NAME

newfs - construct a new file system

SYNOPSIS

/usr/etc/newfs [-nNv] [*mkfs-options* **]** *block-special-file*

DESCRIPTION

newfs is a "friendly" front-end to the **mkfs**(8) program. On Solbourne systems, the disk type is determined by reading the disk label for the specified *block-special-file*.

block-special-file is the name of a block special device residing in **/dev**. If you want to make a file system on **sd0**, you can specify **sd0 rsd0** or **/dev/rsd0**; if you only specify **sd0**, **newfs** will find the proper device.

newfs then calculates the appropriate parameters to use in calling **mkfs**, builds the file system by forking **mkfs** and, if the file system is a root partition, installs the necessary bootstrap programs in its initial 16 sectors.

OPTIONS

-n Do not install the bootstrap programs.

-N Print out the file system parameters without actually creating the file system.

-v Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

mkfs-options

Options that override the default parameters passed to **mkfs**(8) are:

-b *block-size*

The block size of the file system in bytes.

-c *#cylinders/group*

The number of cylinders per cylinder group in a file system. The default value used is 16.

-d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

-f *frag-size*

The fragment size of the file system in bytes.

-i *bytes/inode*

This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

-m *free-space%*

The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

-o *optimization*

(**space** or **time**). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the **-m** option) is less than 10%, the default is to optimize for space; if the minimum free space threshold is greater than or equal to 10%, the default is to optimize for time.

-r *revolutions/minute*

The speed of the disk in revolutions per minute (normally 3600).

-s *size* The size of the file system in sectors.

-t *#tracks/cylinder*
The number of tracks per cylinders on the disk.

FILES

/usr/etc/mkfs	to actually build the file system
/usr/mdec	for boot strapping programs **/dev**

SEE ALSO

fs(5), **fsck**(8), **mkfs**(8), **tunefs**(8)

System and Network Administration

NAME

ps - display the status of current processes

SYNOPSIS

ps [**-acCegklnrStuvwxU**] [*num*] [*kernel_name*] [*c_dump_file*] [*swap_file*]

DESCRIPTION

ps displays information about processes. Normally, only those processes that are running with your effective user ID and are attached to a controlling terminal (see **termio**(4)) are shown. Additional categories of processes can be added to the display using various options. In particular, the **-a** option allows you to include processes that are not owned by you (that do not have your user ID), and the **-x** option allows you to include processes without control terminals. When you specify both **-a** and **-x**, you get processes owned by anyone, with or without a control terminal. The **-r** option restricts the list of processes printed to running processes: runnable processes, those in page wait, or those in disk or other short-term waits.

ps displays the process ID, under PID; the control terminal (if any), under TT; the cpu time used by the process so far, including both user and system time), under CPU; the state of the process, under STAT; and finally, an indication of the COMMAND that is running.

The state is given by a sequence of four letters, for example, '**RWNA**'.

First letter indicates the runnability of the process:

- **R** Runnable processes,
- **T** Stopped processes,
- **P** Processes in page wait,
- **D** Processes in disk (or other short term) waits,
- **S** Processes sleeping for less than about 20 seconds,
- **I** Processes that are idle (sleeping longer than about 20 seconds),
- **Z** Processes that have terminated and that are waiting for their parent process to do a **wait**(2) (zombie processes).

Second letter indicates whether a process is swapped out;

- *blank* (that is, a SPACE) in this position indicates that the process is loaded (in memory).
- **W** Process is swapped out.
- **>** Process has specified a soft limit on memory requirements and has exceeded that limit; such a process is (necessarily) not swapped.

Third letter indicates whether a process is running with altered CPU scheduling priority (nice):

- *blank* (that is, a SPACE) in this position indicates that the process is running without special treatment.
- **N** The process priority is reduced,
- **<** The process priority has been raised artificially.

Fourth letter indicates any special treatment of the process for virtual memory replacement. The letters correspond to options to the **vadvise**(2) system call. Currently the possibilities are:

- *blank* (that is, a SPACE) in this position stands for **VA_NORM**.
- **A** Stands for **VA_ANOM**. An **A** typically represents a program which is doing garbage collection.
- **S** Stands for **VA_SEQL**. An **S** is typical of large image processing programs that are using virtual memory to sequentially address voluminous data.

kernel_name specifies the location of the system namelist. If the **-k** option is given, *c_dump_file* tells **ps** where to look for the core dump. Otherwise, the core dump is located in the file **/vmcore** and this argument is ignored. *swap_file* gives the location of a swap file other than the default, **/dev/drum**.

OPTIONS

-a Include information about processes owned by others.

-c Display the command name, as stored internally in the system for purposes of accounting, rather than the command arguments, which are kept in the process' address space. This is more reliable, if less informative, since the process is free to destroy the latter information.

-C Display raw CPU time in the %CPU field instead of the decaying average.

-e Display the environment as well as the arguments to the command.

-g Display all processes. Without this option, **ps** only prints interesting processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.

-k Normally, *kernel_name* defaults to **/vmunix**, *c_dump_file* is ignored, and *swap_file* defaults to **/dev/drum**. With the **-k** option in effect, these arguments default to **/vmunix**, **/vmcore**, and **/dev/drum**, respectively.

-l Display a long listing, with fields **PPID**, **CP**, **PRI**, **NI**, **SZ**, RSS and WCHAN as described below.

-n Produce numerical output for some fields. In a long listing, the WCHAN field is printed numerically rather than symbolically, or, in a user listing, the USER field is replaced by a UID field.

-r Restrict output to running processes.

-S Display accumulated CPU time used by this process and all of its reaped children.

t*x* Restrict output to processes whose controlling terminal is *x* (which should be specified as printed by **ps**, for example, **t3** for **/dev/tty3**, **tco** for **/dev/console**, **td0** for **/dev/ttyd0**, **t?** for processes with no terminal, etc). This option must be the last one given.

-u Display user-oriented output. This includes fields **USER**, **%CPU**, **%MEM**, **SZ**, RSS and START as described below.

-v Display a version of the output containing virtual memory. This includes fields **RE**, **SL**, **PAGEIN**, **SIZE**, **RSS**, **LIM**, %CPU and **%MEM**, described below.

-w Use a wide output format (132 columns rather than 80); if repeated, that is, **-ww**, use arbitrarily wide output. This information is used to decide how much of long commands to print.

-x Include processes with no controlling terminal.

-U Update a private database where **ps** keeps system information. Thus, '**ps -U**' should be included in the **/etc/rc** file.

num A process number may be given, in which case the output is restricted to that process. This option must also be last.

DISPLAY FORMATS

Fields that are not common to all output formats:

USER Name of the owner of the process.

%CPU CPU utilization of the process; this is a decaying average over up to a

minute of previous (real) time. Since the time base over which this is computed varies (since processes may be very young) it is possible for the sum of all %CPU fields to exceed 100%.

NI — Process scheduling increment (see **getpriority**(2) and **nice**(3C).)

SIZE

SZ — The combined size of the data and stack segments (in kilobyte units)

RSS — Real memory (resident set) size of the process (in kilobyte units).

LIM — Soft limit on memory used, specified using a call to **getrlimit**(2); if no limit has been specified then shown as *xx*.

%MEM — Percentage of real memory used by this process.

RE — Residency time of the process (seconds in core).

SL — Sleep time of the process (seconds blocked).

PAGEIN — Number of disk I/O's resulting from references by the process to pages not loaded in core.

UID — Numerical user-ID of process owner.

PPID — Numerical ID of parent of process.

CP — Short-term CPU utilization factor (used in scheduling).

PRI — Process priority (non-positive when in non-interruptible wait).

START — Time the process was created if that was today, or the date it was created if that was before today.

WCHAN — Event on which process is waiting (an address in the system). A symbol is chosen that classifies the address, unless numerical output is requested (see the **n** flag). In this case, the address is printed in hexadecimal.

F — Flags associated with process as in **<sys/proc.h>**:.

SLOAD	00000001	in core
SSYS	00000002	swapper, pager, or idle process
SLOCK	00000004	process being swapped out
SSWAP	00000008	save area flag
STRC	00000010	process is being traced
SWTED	00000020	parent has been told that this process stopped
SULOCK	00000040	user settable lock in core
SPAGE	00000080	process in page wait state
SKEEP	00000100	another flag to prevent swap out
SOMASK	00000200	restore old mask after taking signal
SWEXIT	00000400	working on exiting
SPHYSIO	00000800	doing physical I/O
SVFORK	00001000	process resulted from **vfork**()
SVFDONE	00002000	another vfork flag
SNOVM	00004000	no vm, parent in a **vfork**()
SPAGI	00008000	init data space on demand, from inode
SSEQL	00010000	user warned of sequential vm behavior
SUANOM	00020000	user warned of anomalous vm behavior
STIMO	00040000	timing out during sleep
SPGLDR	00080000	process is session process group leader
STRACNG	00100000	process is tracing another process
SOWEUPC	00200000	owe process an **addupc**() call at next ast
SSEL	00400000	selecting; wakeup/waiting danger

SLOGIN	00800000	a login process (legit child of init)
SMASTER	01000000	process must execute only on master cpu
SFAVORD	02000000	favored treatment in swapout and pageout
SLKDONE	04000000	record-locking has been done
STRCSYS	08000000	tracing system calls
SRUNQ	10000000	process on run queue
SGLOBAL	20000000	process on global run queue
SIDLE	40000000	process is an idle process
SACTIVE	80000000	process is running on a cpu right now

A process that has exited and has a parent, but has not yet been waited for by the parent is marked <defunct>; a process that is blocked trying to exit is marked <exiting>; otherwise, **ps** makes an educated guess as to the file name and arguments given when the process was created by examining memory or the swap area. The method is inherently somewhat unreliable and in any event a process is entitled to destroy this information, so the names cannot be counted on too much.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory
/dev/drum	swap device
/vmcore	core file
/dev	searched to find swap device and terminal names
/etc/psdatabase	system namelist, device, and wait channel information

SEE ALSO

kill(1), **w**(1), **getpriority**(2), **getrlimit**(2), **wait**(2), **vadvise**(2), **nice**(3C), **termio**(4), **pstat**(8)

BUGS

Things can change while **ps** is running; the picture it gives is only a close approximation to the current state.

NAME

pstat - print system facts

SYNOPSIS

/usr/etc/pstat [**-afipSsT**] [**-u** *pid*] [*system* [*corefile*]]

DESCRIPTION

pstat interprets the contents of certain system tables. If *corefile* is given, the tables are sought there, otherwise in **/dev/kmem**. The required namelist is taken from **/vmunix** unless *system* is specified.

OPTIONS

-a Under **-p**, describe all process slots rather than just active ones.

-f Print the open file table with these headings:

- LOC — The memory address of this table entry.
- TYPE — The type of object the file table entry points to.
- FLG — Miscellaneous state variables encoded thus:
  - R open for reading
  - W open for writing
  - A open for appending
  - S shared lock present
  - X exclusive lock present
  - I signal pgrp when data ready
- CNT — Number of processes that know this open file.
- MSG — Number of references from message queue.
- DATA — The location of the vnode table entry or socket for this file.
- OFFSET — The file offset (see **lseek**(2)).

-i Print the inode table including the associated vnode entries with these headings:

- ILOC — The memory address of this table entry.
- IFLAG — Miscellaneous inode state variables encoded thus:
  - A inode access time must be corrected
  - C inode change time must be corrected
  - L inode is locked
  - R inode is being referenced
  - U update time (**fs**(5)) must be corrected
  - W wanted by another process (L flag is on)
- IDEVICE — Major and minor device number of file system in which this inode resides.
- INO — I-number within the device.
- MODE — Mode bits in octal, see **chmod**(2).
- NLK — Number of links to this inode.
- UID — User ID of owner.
- SIZE/DEV — Number of bytes in an ordinary file, or major and minor device of special file.
- VFLAG — Miscellaneous vnode state variables encoded thus:
  - R root of its file system
  - S shared lock applied
  - E exclusive lock applied
  - Z process is waiting for a shared or exclusive lock
- CNT — Number of open file table entries for this vnode.
- SHC — Reference count of shared locks on the vnode.
- EXC — Reference count of exclusive locks on the vnode (this may be '**> 1**' if, for example, a file descriptor is inherited across a

fork).

TYPE Vnode file type, either VNON (no type), VREG (regular), VDIR (directory), VBLK (block device), VCHR (character device), VLNK (symbolic link), VSOCK (socket), VFIFO (named pipe), or VBAD (bad).

-p Print process table for active processes with these headings:

LOC The memory address of this table entry.

S Run state encoded thus:

0 no process
1 awaiting an event
2 (abandoned state)
3 runnable
4 being created
5 being terminated
6 stopped (by signal or under trace)

F Miscellaneous state variables, ORed together (hexadecimal):

0000001	loaded
0000002	a system process (scheduler or page-out daemon)
0000004	locked for swap out
0000008	swapped out during process creation
0000010	process is being traced
0000020	tracing parent has been told that process is stopped
0000040	user settable lock in memory
0000080	in page-wait
0000100	prevented from swapping during **fork**(2)
0000200	will restore old mask after taking signal
0000400	exiting
0000800	doing physical I/O
0001000	process resulted from a **vfork**(2) which is not yet complete
0002000	another flag for **vfork**(2)
0004000	process has no virtual memory, as it is a parent in the context of **vfork**(2)
0008000	process is demand paging pages from its executable image vnode
0010000	process has advised of sequential VM behavior with **vadvise**(2)
0020000	process has advised of random VM behavior with **vadvise**(2)
0080000	process is a session process group leader
0100000	process is tracing another process
0200000	process needs a profiling tick
0400000	process is scanning descriptors during select
4000000	process has done record locks
8000000	process is having its system calls traced

PRI	Scheduling priority, see **getpriority**(2).
SIG	Signals received (signals 1-32 coded in bits 0-31),
UID	Real user ID.
SLP	Amount of time process has been blocked.
TIM	Time resident in seconds; times over 127 coded as 127.
CPU	Weighted integral of CPU time, for scheduler.
NI	Nice level, see **getpriority**(2).
PGRP	Process number of root of process group.
PID	The process ID number.
PPID	The process ID of parent process.
RSS	Resident set size — the number of physical page frames allocated to this process.
SRSS	RSS at last swap (0 if never swapped).
SIZE	The size of the process image. That is, the sum of the data and stack segment sizes, not including the sizes of any shared libraries.
WCHAN	Wait channel number of a waiting process.
LINK	Link pointer in list of runnable processes.

-S Print the streams table with these headings:

LOC	The memory address of this table entry.
WRQ	The address of this stream's write queue.
VNODE	The address of this stream's vnode.
DEVICE	Major and minor device number of device to which this stream refers.
PGRP	This stream's process group number.
FLG	Miscellaneous stream state variables encoded thus:

I	waiting for **ioctl**( ) to finish
R	read/recvmsg is blocked
W	write/putmsg is blocked
P	priority message is at stream head
H	device has been "hung up" (M_HANGUP)
O	waiting for open to finish
M	stream is linked under multiplexor
D	stream is in message-discard mode
N	stream is in message-nondiscard mode
E	fatal error has occurred (M_ERROR)
T	waiting for queue to drain when closing
2	waiting for previous **ioctl**( ) to finish before starting new one
3	waiting for acknowledgment for **ioctl**( )
B	stream is in non-blocking mode
A	stream is in asynchronous mode
o	stream uses old-style no-delay mode
S	stream has had **TOSTOP** set
C	VTIME clock running
V	VTIME timer expired
r	collision on **select**( ) for reading
w	collision on **select**( ) for writing
e	collision on **select**( ) for exceptional condition

The queues on the write and read sides of the stream are listed for each stream. Each queue is printed with these headings:

NAME	The name of the module or driver for this queue.

COUNT The approximate number of bytes on this queue.
FLG Miscellaneous state variables encoded thus:
- E queue is enabled to run
- R someone wants to get from this queue when it becomes non-empty
- W someone wants to put on this queue when it drains
- F queue is full
- N queue should not be enabled automatically by a putq

MINPS The minimum packet size for this queue.
MAXPS The maximum packet size for this queue, or INF if there is no maximum.
HIWAT The high-water mark for this queue.
LOWAT The low-water mark for this queue.

-s Print information about swap space usage:

allocated: The amount of swap space (in bytes) allocated to private pages.

reserved: The number of swap space bytes not currently allocated, but claimed by memory mappings that have not yet created private pages.

used: The total amount of swap space, in bytes, that is either allocated or reserved.

available: The total swap space, in bytes, that is currently available for future reservation and allocation.

-T Print the number of used and free slots in the several system tables. This is useful for checking to see how full system tables have become if the system is under heavy load. Shows both used and cached inodes.

-u *pid* Print information about the process with ID *pid*.

FILES

/vmunix	namelist
/dev/kmem	default source of tables

SEE ALSO

ps(1), **chmod**(2), **fork**(2), **lseek**(2), **getpriority**(2), **stat**(2), **vadvise**(2), **vfork**(2), **fs**(5), **iostat**(8), **vmstat**(8)

BUGS

It would be very useful if the system recorded "maximum occupancy" on the tables reported by -T; even more useful if these tables were dynamically allocated.

NAME

savecore - save a core dump of the operating system

SYNOPSIS

/usr/etc/savecore *dirname* [*system-name*]

DESCRIPTION

savecore saves a core dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is meant to be called near the end of the **/etc/rc.local** file after the system boots. However, it is not normally run by default. You must edit that file to enable it.

savecore checks the core dump to be certain it corresponds with the version of the operating system currently running. If it does, **savecore** saves the core image in the file *dirname*/**vmcore.***n* and the kernel's namelist, in *dirname*/**vmunix.***n* The trailing *.n* in the pathnames is replaced by a number which grows every time **savecore** is run in that directory.

Before **savecore** writes out a core image, it reads a number from the file *dirname*/**minfree**. If there is less free space on the filesystem containing *dirname* than the number obtained from the minfree file, the core dump is not saved. If the **minfree** file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility **LOG_AUTH** (see **syslog**(3)) If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump was from a system other than **/vmunix**, the name of that system must be supplied as *system-name*.

FILES

/vmunix	the kernel
/etc/rc.local	

SEE ALSO

syslog(3), **sa**(8), **crash**(8S)

BUGS

Can be fooled into thinking a core dump is the wrong size.

You must run **savecore** very soon after booting — before the swap space containing the crash dump is overwritten by programs currently running.

NAME

vmstat - report virtual memory statistics

SYNOPSIS

vmstat [**-fisS**] [*interval* [*count*]]

DESCRIPTION

vmstat delves into the system and normally reports certain statistics kept about process, virtual memory, disk, trap and CPU activity.

Without options, **vmstat** displays a one-line summary of the virtual memory activity since the system has been booted. If *interval* is specified, **vmstat** summarizes activity over the last *interval* seconds. If a *count* is given, the statistics are repeated *count* times.

For example, the following command displays a summary of what the system is doing every five seconds. This is a good choice of printing interval since this is how often some of the statistics are sampled in the system.

```
example% vmstat 5
 procs   memory            page    faults
 r b w  avm fre re at pi po fr de sr x0 x1 x2 x3 in  sy cs us sy id
 2 0 0  918 286  0  0  0  0  0  0  0  1  0  0  0  4 12  5  3  5 91
 1 0 0  846 254  0  0  0  0  0  0  0  6  0  1  0 42 153 31 7 40 54
 1 0 0  840 268  0  0  0  0  0  0  0  5  0  0  0 27 103 25 8 26 66
 1 0 0  620 312  0  0  0  0  0  0  0  6  0  0  0 26 76  25 6 27 67
^C
example%
```

The fields of **vmstat**'s display are:

procs Report the number of processes in each of the three following states:

- **r** in run queue
- **b** blocked for resources (i/o, paging, etc.)
- **w** runnable or short sleeper (< 20 secs) but swapped

memory Report on usage of virtual and real memory. Virtual memory is considered active if it belongs to processes which are running or have run in the last 20 seconds.

- **avm** number of active virtual Kbytes
- **fre** size of the free list in Kbytes

page Report information about page faults and paging activity. The information on each of the following activities is averaged each five seconds, and given in units per second.

- **re** page reclaims — but see the **-S** option for how this field is modified.
- **at** number of attaches — but see the **-S** option for how this field is modified.
- **pi** kilobytes per second paged in
- **po** kilobytes per second paged out
- **fr** kilobytes freed per second
- **de** anticipated short term memory shortfall in Kbytes
- **sr** pages scanned by clock algorithm, per-second

disk Report number of disk operations per second (this field is system dependent). For Solbourne systems, four slots are available for up to four drives: "x0" (or "s0" for SCSI disks), "x1", "x2", and "x3".

faults Report trap/interrupt rate averages per second over last 5 seconds.
- **in** (non clock) device interrupts per second
- **sy** system calls per second
- **cs** CPU context switch rate (switches/sec)

cpu Give a breakdown of percentage usage of CPU time.
- **us** user time for normal and low priority processes
- **sy** system time
- **id** CPU idle

OPTIONS

-f Report on the number of **forks** and **vforks** since system startup and the number of pages of virtual memory involved in each kind of fork.

-i Report the number of interrupts per device. Autovectored interrupts (including the clock) are listed first.

-s Display the contents of the **sum** structure, giving the total number of several kinds of paging-related events which have occurred since boot. Some statistics are given on a per-cpu basis.

-S Report on swapping rather than paging activity. This option will change two fields in **vmstat**'s "paging" display: rather than the "re" and "at" fields, **vmstat** will report "si" (swap-ins), and "so" (swap-outs).

FILES

/dev/kmem
/vmunix

BUGS

If more than one autovectored device has the same name, interrupts are counted for all like-named devices regardless of unit number. Such devices are listed with a unit number of '**?**'.

NAME

iostat - report I/O statistics

SYNOPSIS

iostat [*interval* [*count*]] [*drivename*]

DESCRIPTION

iostat iteratively reports the number of characters read and written to terminals, and, for each disk, the number of kilobytes transferred per second, and the milliseconds per average seek. It also gives the percentage of time the system has spent in user mode, in user mode running low priority (niced) processes, in system mode, and idling.

To compute this information, for each disk, seeks and data transfer completions and number of words transferred are counted; for terminals collectively, the number of input and output characters are counted. Also, each fiftieth of a second, the state of each disk is examined and a tally is made if the disk is active. From these numbers and given the transfer rates of the devices approximate average seek times are calculated for each device.

The optional *interval* argument causes **iostat** to report once each *interval* seconds. The first report is for all time since a reboot and each subsequent report is for the last interval only.

The optional *count* argument restricts the number of reports.

The optional *drivename* forces **iostat** to display information for that disk if it is active, then any other active drives that fit.

FILES

/dev/kmem
/vmunix

SEE ALSO

vmstat(8)

NAME

uustat - uucp status inquiry and job control

SYNOPSIS

uustat -a | **-m** | **-p** | | **-k***jobid*] | **-r***jobid*]

uustat [**-s***system*] [**-u***user*]

DESCRIPTION

uustat displays the status of, or cancels, previously specified **uucp**(1C) commands. It also reports the status of **uucp** connections to other systems. When no options are given, **uustat** displays the status of all **uucp** requests issued by the current user.

OPTIONS

Only one of the following options can be specified at a time:

-a Output all jobs in queue.

-m Report the status of accessibility of all machines.

-p Execute a **ps -flp** for all the PIDs listed in the lock files.

-q List the jobs queued for each machine. If a status file exists for the machine, its date, time status information are reported. In addition, if a number appears in parentheses next to the number of C or X files, it is the age in days of the oldest C./X. file for that system. The **Retry** field represents the number of hours until the next possible call. The **Count** is the number of failure attempts. For systems with a moderate number of outstanding jobs, this could take 30 seconds or more to execute. An example of the output from **-q** is:

eagle 3C 04/07-11:07NO DEVICES AVAILABLE
mh3bs3 2C 07/07-10:42SUCCESSFUL

This indicates the number of command files that are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system followed by the status of the interaction.

-k*jobid* Kill the **uucp** request with job identification of *jobid*. You must either own the job to be killed, or be the super-user.

-r*jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the next limit imposed by the daemon.

The following options can be specified separately or together:

-s*sys* Report the status of all **uucp** requests for remote system *sys*.

-u*user* Report the status of all **uucp** requests issued by user.

Output for both the **-s** and **-u** options has the following format:

eaglen0000 4/07-11:01:03(POLL)
eagleN1bd7 4/07-11:07Seagledan522 /usr/dan/A
eagleC1bd8 4/07-11:07Seagledan59 D.3b2a12ce4924
4/07-11:07Seagledanrmail mike

The first field is the job ID. This is followed by the date and time. The next field is either an **S** or **R** depending on whether the job is to send or request a file. This is followed by the user ID of the user who queued the job. The next field contains the size of the file,

or in the case of a remote execution request, the name of the command. When the size appears in this field, the file name is also given. This can either be the name given by the user, or an internal name created for data files associated with remote executions (**rmail** in this example).

FILES

/var/spool/uucp/* **uucp** spool directories

SEE ALSO

uucp(1C)

NAME

etherfind - find packets on Ethernet

SYNOPSIS

etherfind [**-nprtuvx**] [**-c** *count*] [**-i** *interface*] *expression*

DESCRIPTION

etherfind prints out the headers of packets on the ethernet that match the boolean *expression*. When an internet packet is fragmented into more than one ethernet packet, all fragments except the first are marked with an asterisk. You must be root to invoke **etherfind**.

OPTIONS

-n Do not convert host addresses and port numbers to names.

-p Normally, the selected interface is put into promiscuous mode, so that **etherfind** has access to all packets on the ethernet. However, when the **-p** flag is used, the interface will not go promiscuous.

-r RPC mode: treat each packet as an RPC message, printing the program and procedure numbers.

-t Timestamps: precede each packet listing with a time value in seconds and hundredths of seconds since the first packet.

-u Make the output line buffered.

-v Verbose mode: print out some of the fields of TCP and UDP packets.

-x Dump the header in hex, in addition to the line printed for each packet by default.

-c *count*

Exit after receiving *count* packets. This is sometimes useful for dumping a sample of ethernet traffic to a file for later analysis.

-i *interface*

etherfind listens on *interface*. The program **netstat**(8C) when invoked with the **-i** flag lists all the interfaces that a machine has.

expression

The syntax of of *expression* is similar to that used by **find**(1). Here are the allowable primaries.

-dst *destination*

True if the destination field of the packet is *destination*, which may be either an address or a name.

-src *source*

True if the source field of the packet is *source*, which may be either an address or a name.

-between *host1 host2*

True if either the source of the packet is *host1* and the destination *host2*, or the source is *host2* and the destination *host1*.

-dstnet *destination*

True if the destination field of the packet has a network part of *destination*, which may be either an address or a name.

-srcnet *source*

True if the source field of the packet has a network part of *source*, which may be either an address or a name.

-srcport *port*
True if the packet has a source port value of *port*. It must be either upd or tcp (see **tcp**(4P)), **udp**(4P)). The *port* can be a number or a name used in **/etc/services**.

-dstport *port*
True if the packet has a destination port value of *port*. The *port* can be a number or a name.

-less *length*
True if the packet has a length less than or equal to *length*.

-greater *length*
True if the packet has a length greater than or equal to *length*.

-proto *protocol*
True if the packet is an ip packet (see **ip**(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **udp**, **nd**, or **tcp**.

-byte *byte op value*
True if byte number *byte* of the packet is in relation *op* to *value*. Legal values for *op* are **+**, **<**, **>**, **&**, and **|**. Thus **4=6** is true if the fourth byte of the packet has the value 6, and **20&0xf** is true if byte twenty has one of its four low order bits nonzero.

-broadcast
True if the packet is a broadcast packet.

-arp True if the packet is a arp packet (see **arp**(4P)).

-rarp True if the packet is a rarp packet.

-ip True if the packet is an ip packet.

The primaries may be combined using the following operators (in order of decreasing precedence):

A parenthesized group of primaries and operators (parentheses are special to the Shell and must be escaped).

The negation of a primary ('**!**' is the unary *not* operator).

Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries).

Alternation of primaries ('**-o**' is the *or* operator).

EXAMPLE

To find all packets arriving at or departing from sundown

```
example% etherfind -src sundown -o -dst sundown
example%
```

SEE ALSO

find(1), **traffic**(1C), **arp**(4P), **ip**(4P), **nit**(4P) **tcp**(4P), **udp**(4P), **netstat**(8C)

BUGS

The syntax is painful.

NAME

nfsstat - Network File System statistics

SYNOPSIS

nfsstat [-csnrz]

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is **nfsstat -csnr** That is, display everything, but reinitialize nothing.

OPTIONS

-c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **-n** and **-r** options to print client NFS or client RPC information only.

-s Display server information.

-n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **-c** and **-s** options to print client or server NFS information only.

-r Display RPC information.

-z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the fields:

calls	total number of RPC calls received
badcalls	total number of calls rejected
nullrecv	number of times no RPC packet was available when trying to receive
badlen	number of packets that were too short
xdrcall	number of packets that had a malformed header

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls	total number of RPC calls sent
badcalls	total of calls rejected by a server
retrans	number of times a call had to be retransmitted
badxid	number of times a reply did not match the call
timeout	number of times a call timed out
wait	number of times a call had to wait on a busy **CLIENT** handle
newcred	number of times authentication information had to be refreshed

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory

NAME

showmount - show all remote mounts

SYNOPSIS

/usr/etc/showmount [**-ade**] [host]

DESCRIPTION

showmount lists all the clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd**(8C) server on *host*, and is saved across crashes in the file **/etc/rmtab**. The default value for *host* is the value returned by **hostname**(1).

OPTIONS

-a Print all remote mounts in the format

hostname:directory

where **hostname** is the name of the client, and **directory** is the root of the file system that has been mounted.

-d List directories that have been remotely mounted by clients.

-e Print the list of exported file systems.

FILES

/etc/rmtab

SEE ALSO

hostname(1), **exports**(5), **exports**(5), **mountd**(8C)

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes '**umount -a**'.

NAME

trpt - transliterate protocol trace

SYNOPSIS

/usr/etc/trpt [**-afjst**] [**-p***hex-address*] [*system* [*core*]]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **getsockopt**(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

-a In addition to the normal output, print the values of the source and destination addresses for each packet recorded.

-f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

-j Just give a list of the protocol control block addresses for which there are trace records.

-s In addition to the normal output, print a detailed description of the packet sequencing information.

-t In addition to the normal output, print the values for all timers at each point in the trace.

-p hex-address

Show only trace records associated with the protocol control block, the address of which follows.

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the **socket**(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **-A** option to **netstat**(8C). Then run **trpt** with the **-p** option, supplying the associated protocol control block addresses. The **-f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **-j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

FILES

/vmunix
/dev/kmem

SEE ALSO

getsockopt(2), **netstat**(8C)

DIAGNOSTICS

no namelist

When the system image does not contain the proper symbols to find the trace buffer; others which should be self explanatory.

BUGS

Should also print the data for each input or output, but this is not saved in the trace record.

The output format is inscrutable and should be described here.

NAME

netstat - show network status

SYNOPSIS

netstat [-aAn] [-f *address_family* **] [system] [core]**

netstat [-n] [-s] [-m | -i | -h | -r] [-f *address_family* **] [system] [core]**

netstat [-n] [-I *interface* **]** *interval* **[system] [core]**

DESCRIPTION

netstat displays the contents of various network-related data structures in various formats, depending on the options you select.

The first form of the command displays a list of active sockets for each protocol. The second form selects one from among various other network data structures. The third form displays running statistics of packet traffic on configured network interfaces; the *interval* argument indicates the number of seconds in which to gather statistics between displays.

The default value for the **system** argument is **/vmunix**; for *core*, the default is **/dev/kmem**.

OPTIONS

-a Show the state of all sockets; normally sockets used by server processes are not shown.

-A Show the address of any protocol control blocks associated with sockets; used for debugging.

-f *address_family*
Limit statistics or address control block reports to those of the specified *address_family*, which can be one of:

inet	For the AF_INET address family, or
unix	For the AF_UNIX family.

-h Show the state of the IMP host table. (This does not work in an environment where the IMP host tables do not exist.)

-i Show the state of interfaces that have been auto-configured. Interfaces that are statically configured into a system, but not located at boot time, are not shown.

-I *interface*
Highlight information about the indicated *interface* in a separate column; the default (for the third form of the command) is the interface with the most traffic since the system was last rebooted. *interface* can be any valid interface listed in the system configuration file, such as **ie0** or **le0**.

-m Show the statistics recorded by management routines for the network's private buffer pool.

-n Show network addresses as numbers. **netstat** normally displays addresses as symbols. This option may be used with any of the display formats.

-r Show the routing tables. (When **-s** is also present, show routing statistics instead.)

-s Show per-protocol statistics. When used with the **-r** option, show routing statistics.

-t Replace queue length information with timer information.

DISPLAYS

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the protocol, and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either:

hostname.port

when the name of the host is specified, or:

network.port

if a socket address specifies a network but no specific host. Each **hostname** and *network* is shown according to its entry in the **/etc/hosts** or the **/etc/networks** file, as appropriate.

If the network or hostname for an address is not known (or if the **-n** option is specified), the numerical network address is shown. Unspecified, or "wildcard", addresses and ports appear as "*". (For more information regarding the Internet naming conventions, refer to **inet**(3N)).

TCP Sockets

The possible state values for TCP sockets are as follows:

CLOSED	Closed: the socket is not being used.
LISTEN	Listening for incoming connections.
SYN_SENT	Actively trying to establish connection.
SYN_RECEIVED	Initial synchronization of the connection under way.
ESTABLISHED	Connection has been established.
CLOSE_WAIT	Remote shut down: waiting for the socket to close.
FIN_WAIT_1	Socket closed, shutting down connection.
CLOSING	Closed, then remote shutdown: awaiting acknowledgement.
LAST_ACK	Remote shut down, then closed: awaiting acknowledgement.
FIN_WAIT_2	Socket closed, waiting for shutdown from remote.
TIME_WAIT	Wait after close for remote shutdown retransmission.

Network Data Structures (Second Form)

The form of the display depends upon which of the **-m**, **-i**, **-h** or **-r**, options you select. (If you specify more than one of these options, **netstat** selects one in the order listed here.)

Routing Table Display

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route (**U** if "up"), whether the route is to a gateway (**G**), and whether the route was created dynamically by a redirect (**D**).

Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The **refcnt** column gives the current number of active uses per route. (Connection-oriented protocols normally hold on to a single route for the duration of a connection, whereas connectionless protocols obtain a route while sending to the same destination.)

The **use** column displays the number of packets sent per route.

The *interface* entry indicates the network interface utilized for the route.

Cumulative Traffic Statistics (Third Form)

When the *interval* argument is given, **netstat** displays a table of cumulative statistics regarding packets transferred, errors and collisions, the network addresses for the interface, and the maximum transmission unit ("mtu"). The first line of data displayed, and

every 24th line thereafter, contains cumulative statistics from the time the system was last rebooted. Each subsequent line shows incremental statistics for the *interval* (specified on the command line) since the previous display.

SEE ALSO

hosts(5), **networks**(5), **protocols**(5), **services**(5), **iostat**(8), **trpt**(8C), **vmstat**(8)

BUGS

The notion of errors is ill-defined. Collisions mean something else for the IMP.

The kernel's tables can change while **netstat** is examining them, creating incorrect or partial displays.

Section 8: YP Services

8.1 Commands Used for Maintaining YP

ypserv (8) — Describes the processes that comprise the YP service. These are **ypserv** (8), the YP map server daemon and **ypbind**, the YP binder daemon. **ypserv** must run on each YP server. **ypbind** must run on all clients.

ypfiles (5) — Describes the file structure of the YP service.

ypinit (8) — Automatically constructs maps from files located in /etc, such as /etc/hosts, /etc/passwd, and others. **ypinit** also constructs initial versions of required maps that are not built from files in /etc, for example, **ypservers**. Use **ypinit** to set up the master YP server and the slave YP servers for the first time. You typically do not use it as an administrative tool for running systems.

ypmake (8) — Describes the use of /var/yp/Makefile, which builds several commonly-changed components of YP maps. These are the maps built from the files in /etc on the master YP server: **passwd** (5), **hosts** (5), **group** (5), **netgroup** (5), **networks** (5), **protocols** (5), and **services** (5).

makedbm (8) — Takes an input file and converts it into a pair of **dbm** files, which then become valid YP maps. For example, "ypmaps.dir" and "ypmaps.pag" are both **dbm** files. You can use **makedbm** to build or rebuild maps not built from /var/yp/Makefile. You can also use **makedbm** to "disassemble" a map, so that you can see the key-value pairs that comprise it. You can also edit the disassembled form using editors such as **vi** (1), **emacs** (1), and **ex** (1), or text processing tools like **awk** (1), **grep** (1), and **cat** (1). The disasssembled form is in the format required for input back into **makedbm**.

ypxfr (8) — Moves a YP map from one YP server to another, using YP itself as the transport medium. You can run **ypxfr** interactively, or periodically from a **crontab** (1) file.

yppush (8) — Requests each of the **ypserv** processes within a domain to transfer a particular map, waits for a summary response from the transfer agent, and prints out the results for each server. You run it on the master YP server.

ypset (8) — Tells a **ypbind** process (the local one, by default) to get YP services for a domain from a named YP server. This is not for casual use.

yppoll (8) — Asks any **ypserv** for the information it holds internally about a single map.

ypcat (1) — Displays the contents of a YP map. Use it when you do not care which server's version you are seeing. If you need to see a particular server's map, **rlogin** to that server (or use **rsh**) and use **makedbm**.

ypmatch (1)	Prints the value for one or more specified keys in a YP map. Again, you have no control over which YP server's version of the map you are seeing.
ypwhich (1)	Use this command to see which YP server a host is using at the moment for YP services, or which YP server is master of a particular map.
ypupdated (8c)	Daemon used for changing YP information. This daemon is normally started up by **inetd**. **ypupdated** consults the file "updaters" in the /var/yp directory to determine which maps should be updated and how to change them. Note that **ypupdated** only works if the network is running secure RPC.

8.2 How Administrative Files Are Consulted on a YP Network

OS/MP programs do not consult the same system administrative files on a network with YP that they would on a network without YP. They consult YP maps instead.

/etc/passwd	Always consulted. If there are + or - entries, the YP password map is consulted, otherwise YP is not used. See **passwd** (5).
/etc/group	Always consulted. If there are + or - entries, the YP group map is consulted, otherwise YP is not used. See **group** (5).
/etc/services	Never consulted. The data that was formerly read from this file now comes from the YP services map.
/etc/protocols	Never consulted. The data that was formerly read from this file now comes from the YP protocols map.
/etc/networks	Never consulted. Data is taken from this file to create the YP networks map.
/etc/netgroup	Never consulted. The data that was formerly read from this file now comes from the YP netgroup map.
/etc/bootparams	Never consulted. The data that was formerly read from this file now comes from the YP bootparams map.
/etc/ethers	Never consulted. The data read from this file comes from the YP netgroup map.
/etc/hosts	Consulted only when booting (by the **ifconfig** command in the /etc/rc.boot file). After that the YP map is used instead.
/etc/hosts.equiv	(And similarly for **.rhosts**) Always consulted, though neither of these files is in the YP domain. (See the section below How Security Is Changed with YP, for a fuller explanation of these two files.) If there are + or - entries, whose arguments are netgroups, the YP netgroup map is consulted, otherwise YP is not used. See **hosts.equiv** (5).
/etc/aliases	Always consulted. Local aliases take precedence over those in the YP database. See /etc/aliases.
/etc/netmasks	Never consulted. The data that was formerly read from this file now comes from the YP netmasks map. See the man page **netmasks** (5).

8.3 How to Set up a Master YP Server

Before setting up the master YP server, there are several steps you must take. You need to set up the YP domain name, if it is to differ from the than the name selected for your network domain during installation. You also need to set up the hostname. By default, /etc/rc.local sets up *domainname* and /etc/rc.boot sets up *hostname*.

To create a new server on an existing network, you go to the /var/yp directory and run /var/yp/ypinit . You are asked whether you want the procedure to die at the first non-fatal error (in which case, you can fix the problem and restart **ypinit**, recommended if you haven't done the procedure before), or to continue despite non-fatal errors. In this second case you can try to fix all the problems by hand, or fix some, then restart **ypinit**. **ypinit** prompts you for a list of other hosts that will also be YP servers. (Initially, this is the set of YP slave servers, but at some future time any of them might become the YP master server.) You need not add any other hosts at this time, but if you know that you will be setting up more YP servers, add them now. You will save yourself some work later, and there is little runtime penalty for doing it. (However, do not name every host in the network.)

Before running **ypinit**, the following files in /etc should be complete and reflect an up-to-date picture of your system: **passwd, hosts, ethers, group, networks, protocols**, and **services**. Also, if you know how /etc/netgroup is going to be set up, do that before running **ypinit**. If you don't know, **ypinit** makes an empty netgroup map. Also, /etc/aliases should be complete.

For security reasons, you may restrict access to the master YP machine to a smaller set of users than that defined by the complete /etc/passwd file. To do this, copy the complete file to some place other than /etc/passwd, and edit out undesired users from the remaining /etc/passwd. For a security-conscious system, this smaller file should not include the YP escape entry discussed in the next section.

After performing these steps, you are ready to create a new master server. Become superuser and change directory to /var/yp. Then run **ypinit** with the **-m** option.

To start providing YP services, invoke /usr/etc/ypserv. It then starts up automatically from /etc/rc.local every time the server boots.

8.4 Altering a YP Client's Files To Use YP Services

Once you decide to run YP at your site, you should have all hosts on the network access the YP maps, rather than potentially out-of-date information in their local administrative files. That policy is enforced by running a **ypbind** process on the client machine (including machines that may be running YP servers), and by abbreviating or eliminating the files that traditionally implemented the YP maps. The files in question are:

```
/etc/passwd
/etc/hosts
/etc/ethers
/etc/group
/etc/networks
/etc/protocols
/etc/services
/etc/netgroup
/etc/aliases
/etc/netmasks
/.rhosts
```

The treatment of each file is discussed in this section.

- /etc/networks, /etc/protocols, /etc/ethers, /etc/services, and /etc/netgroup need not exist on any YP clients.
- /etc/hosts.equiv is never served by YP. However, you can add escape sequences to reference YP. This reduces problems with **rlogin** or **rsh**, which are sometimes caused by different /etc/hosts.equiv files on the two machines.

  To let anyone log on to a machine, you can edit /etc/hosts.equiv to contain a single line, with only the character, + (plus) on it. A line with only a + means that all further entries are retrieved from YP rather than the local file.

  Alternatively, you can exercise more control over logins by using lines of the form:

  ```
  +@trusted_group1
  +@trusted_group2
  -@distrusted_group
  ```

  Each of the names to the right of the at sign (**@**) is assumed to be a netgroup name, defined in the global netgroup database. The netgroup database is served by YP.

  If none of the escape sequences is used, only the entries in /etc/hosts.equiv are used; YP is not used.
- /.rhosts also is never served by YP. Its format is identical to that of /etc/hosts.equiv. However, because this file controls remote root access to the local machine, unrestricted access to it is not recommended. Make the list of trusted hosts explicit, or use netgroup names for the same purpose. You can not use secondary hostnames in your .rhosts, hosts.equiv, or netgroup files. You can, however, use secondary hostnames in /etc/hosts. All of the above files are related in that they enable local machines to access remote machines in some fashion.
- /etc/hosts must contain entries for the local host's name, and the local loopback name. These are accessed at boot time when the YP service is not yet available. After the system is running, and after the **ypbind** process is up, the /etc/hosts file is not accessed at all. An example of the hosts file for YP client raks' is:

```
127.1       localhost
192.42.67.9 raks  # Stefania
```

- /etc/passwd should contain entries for the root user name and the primary users of the machine, and the + escape entry to force the use of the YP service. A few additional entries are recommended: daemon, to allow file-transfer utilities to work; and operator, to let a dump operator log in. A sample YP client's /etc/passwd file looks like:

```
root:9wxntql2tHT.k:0:1:Operator:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/var/spool/uucppublic:
news:*:6:6::/var/spool/news:/bin/csh
sync::1:1::/:/bin/sync
raks:7kjDXZD/Hug2s:624:20:Stefania:/home/dancer/raks:/bin/csh
+::0:0:::
```

  The last line informs the library routines to use the YP service. If you remove the last line in the **passwd** file, you will disable YP password access.

  A program that calls /etc/passwd first looks in the password file on your machine; it will then look in the YP password file only if your machine's password file contains + (plus sign) entries, as shown in the above example. Also, earlier entries in the file take precedence over, or mask later ones with the same user name, or the same user ID. Therefore, please note the order of the entries for daemon and for sync (which have the same user ID) and duplicate it in your own file.

- /etc/group may be reduced to a single line:

```
+:
```

  which forces all translation of group names and group IDs to be made via the YP service. This is the recommended procedure.

8.5 How To Set Up a Slave YP Server

The network must be working to set up a slave YP server — in particular, you must be able to **rcp** files from the master YP server to YP slaves.

To create a new slave server, change directory to /var/yp. From there run **ypinit** with the **-s** option. You must be superuser when you run **ypinit**. Name a host already set up as a YP server as the master. Ideally, the named host really is the master server, but it can be any host that has its YP database set up. The host must be reachable. The default domain name on the machine intended to be the YP slave server must be set up, and must be set to the same domain name as the default domain name on the machine named as the master. Also, an entry for daemon must exist in the /etc/passwd files of both slave and master, and that entry must precede any other entries which have the same user ID. Note the example shown in the section above. You won't be prompted for a list of other servers, but you will have the opportunity to choose whether or

not the procedure gives up at the first non-fatal error.

After running **ypinit**, make copies of /etc/passwd, /etc/hosts, /etc/group, /etc/networks, /etc/protocols, /etc/netgroup, and /etc/services. For instance on a machine named ypslave:

```
ypslave% cp /etc/passwd /etc/passwd-
```

Edit the original files in accordance with the preceding section, Altering a YP Client's Files To Use YP Services, to insure that processes on the slave YP server actually use the YP services, rather than the local ASCII files. (That is, make sure the YP slave server is also a YP client) Make backup copies of the edited files, as well. For instance:

```
ypslave% cp /etc/passwd /etc/passwd+
```

After the YP database gets set up by **ypinit**, type /usr/etc/ypserv to begin supplying YP services. On subsequent reboots, it will start automatically from /etc/rc.local.

8.6 How To Set Up a YP Client

To set up a YP client, edit the local files as described in the, Altering a YP Client's Files to Use YP Services, section. If /usr/etc/ypbind is not running already, start it. With the ASCII databases of /etc abbreviated and /usr/etc/ypbind running, the processes on the machine will be clients of the YP services. At this point, there must be a YP server available; processes will hang if no YP server is available while **ypbind** is running. Note the possible alterations to the client's /etc database as discussed above in the section on altering the client. Because some files may not be there, or some may be specially altered, it is not always obvious how the ASCII databases are being used. The escape conventions used within those files to force data to be included or excluded from the YP databases are found in the following man pages: **passwd, hosts, netgroup, hosts.equiv,** and **group**. In particular, notice that changing passwords in /etc/passwd (by editing the file, or by running **passwd**), only affects the local client's environment. Change the YP password database by running **yppasswd**.

8.7 Reference Information on Troubleshooting YP

For help in troubleshooting problems with YP services, see the *Solbourne System and Network Administration* manual, Sections 14.2.15 through 14.2.20. The topics included are as follows:

- 14.2.15 On Client: Commands Hang
- 14.2.16 On Client: YP Service Unavailable
- 14.2.17 On Client: **ypbind** Crashes
- 14.2.18 On Client: **ypwhich** Inconsistent
- 14.2.19 Debugging a YP Server
- 14.2.20 On Server: **ypserv** Crashes

Section 9: Miscellaneous and 'How To ...' Information

9.1 Setting the Correct Timezones

This subsection tells how to set the correct time zone once OS has been installed.

As root, change directories to /usr/lib/zoneinfo. The file named 'localtime' keeps the correct time information. Replace 'localtime' with a copy of the correct timezoneinfo file. You will find the correct timezoneinfo file either in this current working directory or in the directory of the appropriate country. For example if you want to change to Eastern US time zone and check the change, change to user root and type:

```
#cd /usr/lib/zoneinfo/US
#cp Eastern ../localtime
#date
```

Files beginning with a lower-case letter in the /usr/lib/zoneinfo directory, except for 'localtime', are text files offering helpful info for deciding which timezone from Greenwhich Mean Time (GMT) serves a particular location.

9.2 Extending Swap Space with a File Using swapon

There are several ways to increase swap space without having to repartition your disk: You could mount another partition, or set the default environment variable DEFAULTSWAP to another partition or another disk's partition. For the most flexibility, you can create a file to extend the existing swap space:

As root, create a file on any partition you want to use, using mkfile(8). mkfile creates one or more files that are suitable for use as swap areas. The sticky bit is set, and the file is padded with zeroes by default. The default size is in bytes, but it can be flagged as kilobytes, blocks, or megabytes, with the k, b, or m suffixes, respectively. Modify the /etc/fstab file to mount the newly-created file.

```
# mkfile -v 30m /usr/addswap
```

Add this line to the /etc/fstab file:

```
/usr/addswap      swap      swap      rw  0 0
```

Mount the swap file, invoke swapon(8), and see the change in swap space:

```
# mount -a
# swapon -a
# pstat -s
```

The last command, pstat -s, will report the total swap space available. To delete, remove the swap entry from /etc/fstab, delete the "addswap" file, and reboot.

9.3 Setting up a Modem

First check the kernel configuration to enable hardware carrier detect by looking at the kernel config file in /usr/share/sys/kbus/conf/KERNEL_NAME:

```
device zs0 at kbslot ? csr 0x00012000 flags 0x002 priority 3
```

Following is the table for software/hardware detect flags: e.g., 0x003 Supply carrier in hardware = 0; supply carrier in software = 1.

Carrier		
PortB	PortA	Flag
0	0	0
0	1	1
1	0	2
1	1	3

So you could put a modem on portA if the flag = 0x000 or 0x002
And you could put a modem on portB if the flag = 0x000 or 0x001

If the current kernel is not configured with the correct flags for the port you want, reconfigure and install a new kernel and reboot the system.

Secondly, specify the system logical devices. In order to use a physical device that requires both dial-in and dial-out, you must create two logical devices in /dev that are related to each other by their minor numbers. Minor numbers separated by a value of 128 will separate the port into the two logical devices.

```
# cd /dev
# mv ttya ttyd0
# mknod cua0 c 12 128
# chmod 600 cua0
# chown uucp cua0
# vi /etc/ttytab
```

When editing ttytab, comment out the line for ttya and add this line:

```
ttyd0 "usr/etc/getty std.2400" dialup on secure
```

(if baud rate is 2400)

notify init of the change by typing:

```
# kill -HUP 1
# vi /etc/remote
```

edit /etc/remote to include your tip aliases, phone numbers, baud rates, etc.

```
cua0:dv=/dev/cua0:br#2400
myhost::pn=7723400%:tc=UNIX-2400:
dialers::dv=/dev/cua0:
```

Settings for trailblazers

Regarding official support of modems: We support the serial ports but not what is attached to them.

T2000:	T2500:
AT&F	AT&F
ATS51=255	ATS52=2
ATS52=2	ATS54=3
ATS53=3	ATS111=30
ATS54=3	ATS131=1
ATS111=30	AT&W
AT&W	

A quick reference card note on commands and registers: You may enter a command line of up to 80 characters in upper or lower case with the first command in the line preceded by an "AT" or "at" and the last command followed by "&W <return>." You can repeat the last command issued by entering "A/" or "a/" without first entering the "AT" prefix.

Cabling: For modems, use the straight-through cable. For terminals, use the null-modem cable (pins 2 and 3 switched).

9.4 Setting up a VT100 on a tty Port

Check the kernel configuration to enable hardware carrier detect. See the section above (setting up a modem) for the carrier detect table.

So you could put a terminal on portA if the flag = 0x001 or 0x003
So you could put a terminal on portB if the flag = 0x002 or 0x003

edit /etc/ttytab to specify which serial port will have a login process created. Edit the file to get the following (or whatever baud rate applies):

```
ttyb    "/usr/etc/getty std.9600"  unknown    on    secure
```

terminal type

notify init of the change by typing:

```
# kill -HUP 1
```

Change the setup of the terminal # set term [terminal type]

For a VT100:

```
Set 7, 2, noparity
9600 baud
```

tset

For a VT2XX:

```
Set 7, 2, markparity
9600 baud
```

Note: You may have to try different settings, but typically 7-2-noparity should work.

Cabling: For modems, use the straight-through cable. For terminals, use the null-modem cable (pins 2 and 3 switched).

When in ~~Block~~ the Console is lock,
go to set-up & Key in F2 → under COMM Mode

Change from Block to Full Duplex!

9.5 Installing into a Sun environment

Setting up a Solbourne as a server to Sun clients. It is advisable to have a separate file system for exporting to client machines; i.e., /dev/rsd0h. Create and check this file system:

```
# newfs -n -v /dev/rsd0h
# fsck /dev/sd0h
```

In /etc/fstab, add this (or a similar) entry to mount the export file system:

```
/dev/sd0h          /export          4.2 rw     1 4
```

```
# mkdir /export
# mount /export
```

Add the client information to the /etc/hosts and /etc/ethers files: To the ethers file, add the Ethernet address and machine name:

```
0:0:8e:10:0:ab     soljazz
```

Configure the server by invoking a script: (client_arch are Sun2, Sun3, Sun4, Sun4c, Sun386, Series4, and Series5). Read the beginning of the script for options and usage.

```
# /usr/etc/setup/config_server client_arch
```

Set up the client file system on the server (-b specifies swap size):

```
# /usr/etc/setup/install_client -b 32m clientname client_arch
```

TFTP Boot Process: Nothing is required to boot Solbourne diskless clients, but tftpboot/ipaddress link is needed to boot Sun clients and X terminals (where 'ipaddress' is the Internet address of client in hex). Here is a long listing of some files in a diskless server's /tftpboot directory:

```
lrwxrwxrwx  1 root    19 Feb  3 14:47 C009CC9F -> /tftpboot/boot.sun3*
lrwxrwxrwx  1 root    19 Feb  3 14:47 C009CCAF -> /tftpboot/boot.sun4*
lrwxrwxrwx  1 root    19 Feb  3 14:47 C009CCCB -> /tftpboot/boot.sun3*
lrwxrwxrwx  1 root    12 Feb  3 14:47 C009CC64 -> Xncd16.2.0.0
-rwxr-xr-x  1 root   29800 Feb  7  1989 tpboot.sun3*
-rwxr-xr-x  1 root   43240 Feb  3 13:38 tpboot.sun4*
-rw-r--r--  1 root  683556 Feb  4 21:24 Xncd16.2.0.0
-rw-r--r--  1 root  602484 Feb  4 21:24 Xncd16_s.2.0.0
```

In these examples, machine C009CC9F is a Sun3 machine, C009CCAF is a Sun4 machine, and C009CC64 is an NCD 16 inch X-terminal.

If running YP, update YP maps:

```
# cd /var/yp
# make
```

9.6 How Much Swap Space is Recommended?

For most engineering and scientific applications, the following rule applies:

```
Recommended    Swap Space  =  2  X  Physical Memory  +  10%
```

9.7 Minimal UNIX: Which Files May Be Shared

In the interest of making as much disk space available on a disk, some UNIX files may be deleted or network-shared:

```
/usr/sccs       directory       0.4 Mbytes
/usr/old        directory       0.4 Mbytes
/usr/local      directory       41.3 Mbytes
/usr/share      directory       28.3 Mbytes
/usr/man        directory       6 Mbytes
/usr/games      directory       2.7 Mbytes
/usr/demo       directory       2.1 Mbytes
```

If you don't need any of the System V software:

```
/usr/5lib       directory       2.5 Mbytes
/usr/5bin       directory       .7 Mbytes
/usr/5include   directory       .1 Mbytes
```

9.8 Setting up mail

This section gives the basics for setting up **mail**.
How **mail** is Sent:

1. The user addresses and transmits message via /usr/ucb/Mail or /usr/bin/mail.
2. **sendmail** (8) picks up message from /var/spool/mqueue.
3. The address is parsed according to the rule set of /etc/sendmail.cf.
   - If the address is local, **sendmail** checks the /etc/aliases file and sends the message to the appropriate machine.
   - If the address is off-site **sendmail** forwards message to a **mail** gateway for off-site delivery via UUCP or other network transport medium.
   - If the address is bad **sendmail** notifies the postmaster and originator.

How **mail** is Received:

1. The mailhost receives **mail** message.
2. **sendmail** looks in /etc/aliases file or in the YP name service (passwd).
   - If an alias is found **sendmail** delivers **mail** to /var/spool/mail/user on the appropriate machine and notifies the user of the **mail** delivery.
   - If an alias is not found **sendmail** bounces a message back with an error header to originator and postmaster at originator's site.

sendmail is the main internet electronic **mail** router daemon, not the user interface to the **mail** facility. It parses addresses and routes messages. Commonly-used options are:

-bd	run as a daemon
-bi	initialize the alias database

-bp	print a summary of the **mail** queue
-bv	verify names only
-bz	create the configuration freeze file
-q[*time*]	process messages at given intervals

/etc/sendmail.cf is the **sendmail** configuration file. This file contains the the functional configuration for **sendmail** daemons, general configuration info, rewriting rules (optional) name conversion rules, and rule sets. If the machine is a main machine (one which relays **mail**), this configuration file is a copy of /usr/lib/sendmail.main.cf. If not a main machine, it is a copy of /usr/lib/sendmail.subsidiary.cf.

Configuring for Electronic **mail**

1. Choose a machine on the net to be the mailhost, usually the YP master with UUCP connections. On this host, after making a back-up copy of /etc/sendmail.cf, copy /usr/lib/sendmail.main.cf to /etc/sendmail.cf. Add write permissions to the new config file and edit it.

   Enter the name of the local **mail** server on these lines:
   DR ddn-gateway enter: DRhostname
   CR ddn-gateway enter: CRhostname

   Other configuration file sections include:

macros	Defines items such as the **mail** domain, relay mailer, relay host, names for error messages, and **mail** header format.
options	Includes info message delivery mode and how messages are queued.
precedence	Indicates **mail** class.
trusted users	For UUCP.
header control	A template for the message header. Lines like Date: and and Subject: and To: are defined for the format of headers.
rule set	List of rules for interpreting addresses.

2. Update the /etc/hosts file to add a mailhost alias to the chosen host entry:

   ```
   ipaddress    host_name    mailhost
   ```

   Also make sure all client hosts are entered in this file.

3. Start the **sendmail** daemon: e.g., sendmail -bp -lqh &

4. Update the /etc/aliases file to define a postmaster. The postmaster is a person's login name, usually the system administrator, who troubleshoots **mail**.

   ```
   postmaster: loginname@hostname
   ```

5. Update the YP databases by changing directory to /var/yp and running **make**.

Troubleshooting **mail**:
Make sure only one **sendmail** daemon is running.
Check for write permissions on /usr/spool/mail and /usr/spool/mqueue directories.
Verify that /etc/sendmail.cf is appropriate for mailhost or subsidiary.
Verify correct machine names in /etc/aliases, and verify that they match machine names in /etc/hosts.

9.9 Hostid Conversion from Hex to Decimal

Invoke the arithmetic calculator **bc** (1) and tell it that input will be base 16:

```
# bc
# ibase=16
# 2300056B   (HOSTID value in hex, using only capital letters)
# 587203947  (Returned decimal value)
# quit       (or ^D)
```

To convert from decimal to hex, define the output base to be 16:

```
# bc
# obase=16
# 587203947  (Decimal value to be converted)
# 2300056B   (Returned hex value)
quit
```

9.10 Getting Started with swm and X

Refer to the *swm User's Guide* (part number 103286) for reference on **swm** (1). The following files in the user's home directory are involved:

```
~/.xinitrc
~/.swmrc
~/.Xdefaults
~/.swmdefs
```

Usually you will want to set up an alias to start up the X window environment running the Solbourne Window Manager, either in the ~/.alias or ~/.cshrc files. A simple example alias to start X may be:

```
alias x '/usr/bin/X11/xinit;  kbd_mode -a;  clear
```

The .swmrc file is the configuration file for **swm**, as specified in the .Xdefaults file. The **swm** configuration is specified on one line such as:

```
Swm*configuration: OpenLook+ ~/.swmdefs
```

Called by xinit, .xinitrc is generally used to start up X clients to begin an X session. It starts up the apps listed, including the specified window manager. See Appendix A of the *swm User Guide* for line by line explanation of this file.

In general, the .Xdefaults file is used to set user preferences for X clients. For example, background and foreground colors of windows and the fonts that will be used in windows are set in the .Xdefaults file. This file is read first. See Appendix A of the *swm Users Guide* for line by line explanation of this file.

The .swmdefs file sets up the **swm** user preferences that will override some of the configurations found in any of the default configuration files in /usr/lib/X11/swm. The types of preferences you can set include colors, root panels, menu contents, and bindings. See Appendix A of the *swm User Guide* for line by line explanation of this file.

9.11 Field Notes

Section 10: General Diagnostics Information

10.1 Introduction

Information that may be useful while using any of the Solbourne diagnostics programs is available in the following books:

- *Series4/600 Service Manual*, Part number 101249-AA
- *Series4/600 Theory Manual*, Part number 101250-AA
- *Series4/500 Service Manual*, Part number 102161-AA
- *Bootable/Standalone Multiprocessor Diagnostics Manual*, Part number 101686-AB
- *Bootable/Standalone Diagnostics Manual*, Part number 101490-AB
- *System Power On Self Test Manual*, Part number 101486-AB
- *Extended ROM Resident Diagnostics Manual*, part number 101489-AB

10.2 System Board LEDs

The following table shows the System Board LEDs. See Figures 2-2 and 2-3 for the location of the LEDs. LEDs are numbered 1-10 with number 1 on the left and number 10 on the right.

LED	State	Meaning
1	On	Serial ports and monochrome graphics fuse blown (fuse 4)
2	On	SCSI bus termination power fuse blown
3	On	Keyboard/mouse power fuse blown
4	On	Ethernet +12 VDC fuse blown
5	On	Ethernet +5 VDC fuse blown
6	Blinking	Kbus busy
7	Blinking	VMEbus busy
8	On	VMEbus failed
9	Blinking	Ethernet busy
10	N/A	N/A

10.3 CPU Board LEDs

The first code to be executed in the Boot ROM is the System Power-On Self Tests. The system self test diagnostic routines must execute to completion, without error, before the system can bootstrap any stand alone program or OS/MP.

If no prompt appears on the console after several minutes, the two seven-segment light emitting diodes (LEDs) on the CPU Board(s) should be examined to determine the status of the system.

If the LEDs become locked on a particular state during self test execution, this means that a catastrophic failure has occurred during the test indicated by the last LED state.

When a self test program fails, error information is displayed in on the LEDs. The error information consists of the test number and a unique error code that identifies the failure. Since the LEDs cannot display both the test number and error code simultaneously, the test number and error code must be displayed in a cyclic fashion on the LEDs.

Figure 10-1 illustrates how the error information is displayed.

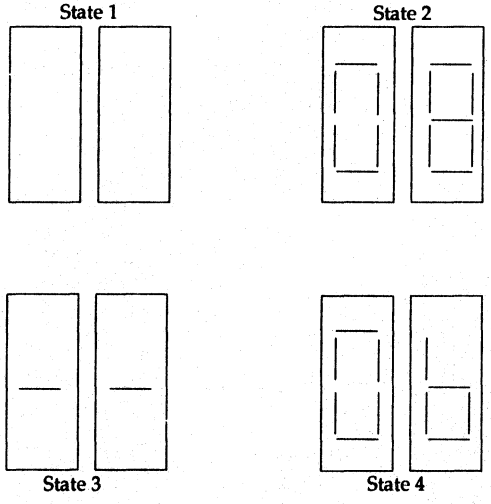


Figure 10-1. Example Display of Error Information

The states from Figure 10-1 are explained below:

- **State 1** - This marks the beginning of the cycle with both LEDs displaying blanks.
- **State 2** - The test number of the failing test is displayed in both LEDs.
- **State 3** - Both LEDs display dashes that indicate the separation of the test number from the error code.
- **State 4** - A unique error code that identifies the failure is displayed.

☆ ☆ ☆ **NOTE** ☆ ☆ ☆

State 4 reads "b," not "6." Also, "5" and "S" are displayed identically. As in "SL" for slave CPU idling or "05" for Test 5 (see Figure 1-3).

Figure 10-2 shows that an unexpected exception occurred. The number following the "- -" block represents the exception (trap) type a data access exception. See the *SPARC Architecture Manual* for additional information on exception trap types.

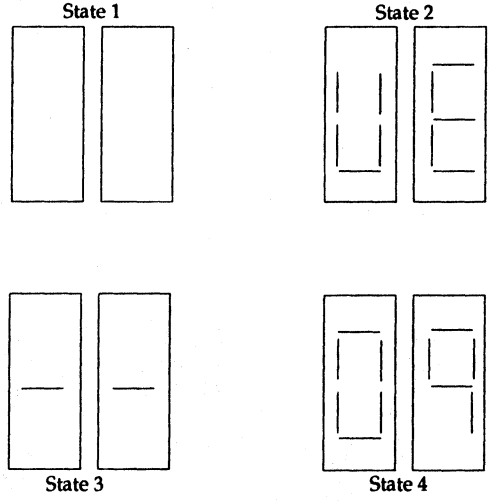


Figure 10-2. Example of Unexpected Exception

For the Series4 CPU, there are the following additional error codes:

00 -- 00	Double trap during tests 1 through 4.
00 -- 01	Double trap during tests 5 through 3X.
00 -- 02	Double trap occurred; no vector defined.

For the Series5 CPU, there are the following additional error codes:

00 -- 04	Double trap occurred; DGRAM not initialized.
00 -- 01	Watchdog trap occurred; no reset vector defined.
00 -- 02	Double trap occurred; no reset vector defined.
00 -- 03	Watchdog and double trap occurred; no reset vector defined.
00 -- 05	Cold start, cannot clear MMCR<CS> bit.

When the ROM monitor program or a stand alone program is checking for input, a dash (-) is alternately displayed between the two LEDs.

When OS/MP is idle, a small "o" moves around from one corner to another in the LEDs.

10.4 Multiprocessor Configuration Self Tests

In Solbourne's master-slave multiprocessor implementation, the power-on self-test is performed in the following sequence:

1. When power is turned on, all installed processors execute the first half of the self test concurrently.
2. The processors determine which CPU board is the master, as defined by the ROM environment variable MASTER (e.g., MASTER=1 for slot one on the Kbus). If the MASTER environment variable has not been set or points to an empty slot, the CPU in the lowest numbered slot will assume mastership.
3. Once the master is defined, the master CPU Board finishes its portion of the self test, while the slave CPU boards enter an idle loop. The master then directs each slave to finish their portion of the self test. The slaves continue to execute their self tests in descending slot order, starting with the slave CPU in the highest slot number. When the slaves complete the self test, they return to their idle loop.

   In state 4 of Figure 10-3, a 6 is displayed in the LED on the right. This number represented the number of the Kbus slot occupied by the slave CPU Board.

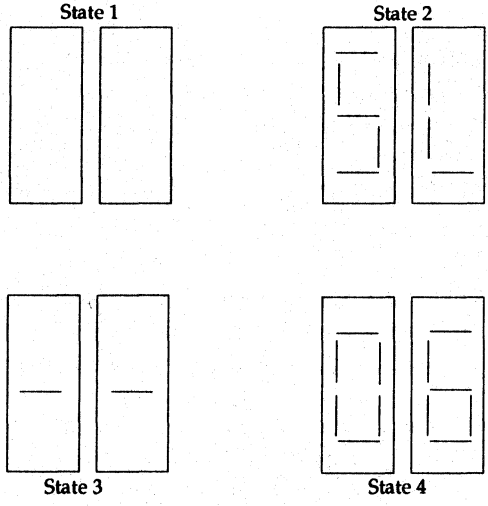


Figure 10-3. Normal Multiprocessor Slave States

4. The master CPU continues to the ROM prompt after all the slave CPU Boards have been directed to complete the self test and have reported their status back to the master.

Section 11: System Power-On Self-Tests

11.1 Series4 Test Descriptions

This section describes the system power-on self-tests that are used on Solbourne Series4 systems. In the these test descriptions, all test numbers and error codes are represented in two digit hex values.

11.1.1 Test 01 - Bootrom Checksum Test

This test computes the checksum on the contents of the four, 27C512 EPROMS on the CPU Board that are used to contain the boot code and data. The expected checksum is burned into the EPROM when the roms are programed during manufacturing. Legal error codes for this test are:

```
00 - Checksum 0 incorrect        04 - Checksum 4 incorrect
01 - Checksum 1 incorrect        05 - Checksum 5 incorrect
02 - Checksum 2 incorrect        06 - Checksum 6 incorrect
03 - Checksum 3 incorrect        07 - Checksum 7 incorrect
```

11.1.2 Test 02 - Diagnostic RAM Addressing and Data Test

This is an addressing and data test for the diagnostic RAM on the CPU Board. The diagnostic RAM a two Kbyte static RAM which is accessed through alternate space (ASI) 0x38 (see H/W description), and responds to every eighth address in the range 0 through 0x3ff8 inclusive. Legal error codes for this test are:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
```

11.1.3 Test 03 - Interrupt Registers Test

This is a write/read test of the interrupt registers. There are four test cases, one for each register tested. The registers tested are: Device ID Register (DIR), Interrupt Priority Register (IPR), Interrupt Transmit Register (IXR), and the Interrupt Pending Vector Register (IPV). Legal error codes for this test are:

```
01 - DIR register write read error
02 - IPR register write read error
03 - IXR register write read error
04 - IPV register write read error
```

11.1.4 Test 04 - Directed Interrupt Test

This test verifies that the CPU Board interrupt logic can send a directed interrupt to itself. There are two test cases:

1. Case 1 verifies that directed interrupts can be transmitted and received
2. Case 2 verifies that the interrupt receiver priority level (set in the IPR register) effectively inhibits interrupts from being received.

Legal error codes for test 04, case 1 are:

```
01 - Interrupt was never acknowledged (ITXC<gone> bit not set).
02 - Interrupt was acknowledged but no incorrect interrupt
     type was generated.
03 - Interrupt occurred, but the IRXC<P> bit was not set.
04 - Interrupt occurred, but the IPV register did not
     contain the transmitted vector.
```

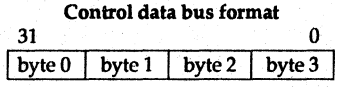
Legal error code for test 04, case 2 is:

```
05 - Interrupt was acknowledged (ITXC<gone> bit set).
06 - Unexpected interrupt was generated
```

11.1.5 Test 05 - Control-Data Bus Test

This test reads the quiescent (undriven) state of the CPU's 32-bit, control data bus 64 Kbyte times and verifies that the bus is floats high (all ones). A common cause of failure for this test is that one of the bootrom outputs is sinking too much current and pulling the control data bus to the low state. The test examines each eight-bit field of the 32-bit bus and reports errors using the following convention:

Control data bus format



Error encoding convention:

```
01 - byte 3 corrupted                  09 - bytes 0 and 3 corrupted
02 - byte 2 corrupted                  0a - bytes 0 and 2 corrupted
03 - bytes 2 and 3 corrupted           0b - bytes 0, 2, and 3 corrupted
04 - byte 1 corrupted                  0c - bytes 0 and 1 corrupted
05 - bytes 1 and 3 corrupted           0d - bytes 0, 1 and 3 corrupted
06 - bytes 1 and 2 corrupted           0e - bytes 0, 1 and 2 corrupted
07 - bytes 1, 2, and 3 corrupted       0f - bytes 0, 1, 2 and 3 corrupted
08 - byte 0 corrupted
```

11.1.6 Test 06 - Control Registers Test

This test verifies that the MMCR and PDEP registers can be written and read. Aside from the interrrupt registers (see test 3) these are the only other two registers that are write-readable. Legal error codes for this test case are:

```
01 - MMCR write/read error
02 - PDEP register write/read error
```

11.1.7 Test 07 - TLB Instruction/Data Uniqueness Test

This test verifies that the instruction portion of the TLB is unique from the data portion of the TLB.

The first part of the test writes the instruction TLB followed by the data TLB, then reads the instruction TLB followed by the data TLB. The legal error codes for test 07, case 1 are:

```
01 - The instruction TLB physical address field does not contain
     the data that was written.
02 - The instruction TLB tag and status field does not contain
     the data that was written.
03 - The data TLB physical address field does not contain the
     data that was written.
04 - The data TLB tag and status field does not contain the data
     that was written.
```

The second part of the test writes the data TLB followed by the instruction TLB, then reads the data TLB followed by the instruction TLB. The legal error codes for this test 07, case 2 are:

```
05 - The data TLB physical address field does not contain the
     data that was written.
06 - The data TLB tag and status field does not contain the data
     that was written.
07 - The instruction TLB physical address field does not contain
     the data that was written.
08 - The instruction TLB tag and status field does not contain
     the data that was written.
```

11.1.8 Test 08 - Instruction TLB RAM Addressing and Data Test

This is a test of the physical address, TAG, and status fields of the instruction TLB rams. The legal error codes for test 08 are:

```
01 - Data error on first forward pass read
02 - Tag or status error on first forward pass read
03 - Data error on second forward pass read
04 - Tag or status error on second forward pass read
05 - Data error on first reverse pass read
06 - Tag or status error on first reverse pass read
07 - Data error on second reverse pass read
08 - Tag or status error on second  reverse pass read
```

11.1.9 Test 09 - Data TLB RAM Addressing and Data Test

This is a test of the physical address, TAG, and status fields of the data TLB rams. The legal error codes for test 09 are:

```
01 - Data error on first forward pass read
02 - Tag or status error on first forward pass read
03 - Data error on second forward pass read
04 - Tag or status error on second forward pass read
05 - Data error on first reverse pass read
06 - Tag or status error on first reverse pass read
07 - Data error on second reverse pass read
08 - Tag or status error on second  reverse pass read
```

11.1.10 Test 0a - TLB Tag Comparitors Test

This is a test of the TLB match detection logic. The test program loads a series of patterns into the tag portion of the TLB then performs a series of TIR reads to verify that the TLB match comparitor works correctly (see Appendix A for more information on TIR reads). There are four cases for test 0a, as follow:

Case 1 - TLB tag set to walking 1 pattern with logical address set to zero. The legal error codes for test 0a, case 1 are:

```
01 - TLB comparitor match error using instruction TLB.
02 - TLB comparitor match error using data TLB.
```

Case 2 - TLB tag set to walking 0 pattern with logical address set to all ones. The legal error codes for test 0a, case 2 are:

```
03 - TLB comparitor match error using instruction TLB.
04 - TLB comparitor match error using data TLB.
```

Case 3 - TLB tag set to zero with logical address set to walking 1 pattern. The legal error codes for test 0a, case 3 are:

```
05 - TLB comparitor match error using instruction TLB.
06 - TLB comparitor match error using data TLB.
```

Case 4 - TLB tag set to all ones with logical address set to walking zero pattern. The legal error codes for test 0a, case 4 are:

```
07 - TLB comparitor match error using instruction TLB.
08 - TLB comparitor match error using data TLB.
```

11.1.11 Test 0b - Cache RAM Bank Uniqueness Test

This test verifies the the cache RAM bank selection mechanism works. The cache RAM bank is selected on the basis of logical address bit 2. The test also verifies that byte, half-word, word and double-word loads and stores to the cache can be performed. It is verified for each access type, that the data is placed in the correct byte/half-word/word/double-word position in the cache.

Initial state: Address 0 written with 0x55555555, address 4 written with 0xaaaaaaaa.

Byte:	0	1	2	3	4	5	6	7
Data:	55	55	55	55	aa	aa	aa	aa

```
Error code 01 - word read at address 0 is not 0x55555555
Error code 02 - word read at address 4 is not 0xaaaaaaaa
```

Second state: Address 0 written with 0xaaaa, address 4 written with 0x5555.

Byte:	0	1	2	3	4	5	6	7
Data:	aa	aa	55	55	55	55	aa	aa

```
Error code 03 - word read at address 0 not 0xaaaa5555
Error code 04 - word read at address 4 not 0x5555aaaa
Error code 05 - double byte read at address 2 not 0x5555
Error code 06 - double byte read at address 6 not 0xaaaa
```

Third state: Address 0 and 7 written with 0x55, address 3 and 4 written with 0xaa.

Byte:	0	1	2	3	4	5	6	7
Data:	55	aa	55	aa	aa	55	aa	55

```
Error code 07 - word read at address 0 not 0x55aa55aa
Error code 08 - word read at address 4 not 0xaa55aa55
Error code 09 - double byte read at address 0 not 0x55aa
Error code 0a - double byte read at address 4 not 0xaa55
Error code 0b - byte read at address 0 not 0x55
Error code 0c - byte read at address 4 not 0xaa
Error code 0d - byte read at address 1 not 0xaa
Error code 0e - byte read at address 5 not 0x55
Error code 0f - byte read at address 2 not 0x55
Error code 10 - byte read at address 6 not 0xaa
Error code 11 - byte read at address 3 not 0xaa
Error code 12 - byte read at address 7 not 0x55
```

Fourth state: Address 0 written with 0xaaaaaaaa55555555 (double word write).

Byte:	0	1	2	3	4	5	6	7
Data:	aa	aa	aa	aa	55	55	55	55

```
Error code 13 - double word read at address 0, first word not 0xaaaaaaaa
Error code 14 - double word read at address 0, second word not 0x55555555
```

11.1.12 Test 0c - Atomic Load/Store Cache Test

This test verifies that the SPARC atomic load/store instruction "ldstub" works correctly. The legal error codes for test 0c are:

```
01 - load portion of ldstub instruction did not read 0x55
02 - store portion of ldstub instruction did not write 0xff
```

11.1.13 Test 0d - Cache RAM Addressing and Data Test

This is an addressing and data test for the Cache Data RAMs. The legal error codes for test 0d are:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
      :
```

11.1.14 Test 0e - Corrupted Block RAM Reset Test

This verifies that all the bits in the Corrupted Block RAM can be reset. The legal error code for test 0e is:

```
01 - Corrupted bit not zero after reset
```

11.1.15 Test 0f - Virtual Tag RAM Addressing and Data Test

This is an addressing and data test for the Virtual Tag RAMs. The legal error codes for this test case are:

```
01 - Virtual Tag match error on first forward pass read
02 - Virtual Tag match error when upper 16 bits of Logical
     Address was complemented on forward pass
03 - Virtual Tag match error on second forward pass read
04 - Virtual Tag match error on first reverse pass read
05 - Virtual Tag match error when upper 16 bits of Logical
     Address was complemented on reverse pass
06 - Virtual Tag match error on second reverse pass read
```

11.1.16 Test 10 - Virtual Tag Comparitors Test

This is a test of the Virtual Tag match detection logic. The four test cases are outlined below:

Case 1 - Virtual tags set to walking 1 pattern with logical address bits 31:16 set to zero. The legal error codes for test 10, case 1 are:

```
01 - VMATCH0 status error
02 - VMATCH1 status error
```

Case 2 - Virtual tags set to walking 0 pattern with logical address bits 31:16 set to all ones. The legal error codes for test 10, case 2 are:

```
03 - VMATCH0 status error
04 - VMATCH1 status error
```

Case 3 - Virtual tags set to zero with logical address bits 31:16 set to walking 1 pattern. The legal error codes for test 10, case 3 are:

```
05 - VMATCH0 status error
06 - VMATCH1 status error
```

Case 4 - Virtual tags set to all ones with logical address bits set to walking zero pattern. The legal error codes for test 10, case 4 are:

```
07 - VMATCH0 status error
08 - VMATCH1 status error
```

11.1.17 Test 11 - Physical Tag RAM Address and Data Test

This is an addressing and data test for the physical tag RAMs. The legal error codes for test 11 are:

```
01 - Physical tag match or status error on first forward pass read
02 - Physical tag match or status error when TLB physical address
     field was complemented on forward pass
03 - Physical tag match or status error on second forward pass read

04 - Physical tag match or status error on first reverse pass read
05 - Physical tag match or status error when TLB physical address
     field was complemented on reverse pass
06 - Physical tag match or status error on second reverse pass read
```

11.1.18 Test 12 - Physical Tag Comparitors Test

This is a test of the Physical Tag match detection logic. There are 4 test cases as outlined below:

Case 1 - Physical tags set to walking 1 pattern with physical address field of TLB set to zero. The legal error codes for test 12, case 1 are:

```
01 - PMATCH0 status error
02 - PMATCH1 status error
03 - PMATCH2 status error
```

Case 2 - Physical tags set to walking 0 pattern with physical address field of TLB set to all ones. The legal error codes for test 12, case 2 are:

```
04 - PMATCH0 status error
05 - PMATCH1 status error
06 - PMATCH2 status error
```

Case 3 - Physical tags set to zero with physical address field of TLB set to walking 1 pattern. The legal error codes for test 12, case 3 are:

```
07 - PMATCH0 status error
08 - PMATCH1 status error
09 - PMATCH2 status error
```

Case 4 - Physical tags set to all ones with physical address field of TLB set to walking zero pattern. The legal error codes for test 12, case 4 are:

```
0a - PMATCH0 status error
0b - PMATCH1 status error
0c - PMATCH2 status error
```

11.1.19 Test 13 - Purge RAM Addressing and Data Test

This is an addressing and data test for the Purge RAMs. The legal error codes for this test are:

```
01 - purge address or valid bit error on first read of forward pass
02 - purge address or valid bit error on second read of forward pass
03 - purge address or valid bit error on first read of reverse pass
04 - purge address or valid bit error on second read of reverse pass
```

11.1.20 Test 14 - Virtual Tag Even Block Revalidation Test

Virtual tag re-validations occur when there is a physical cache hit and a virtual cache miss. This test creates valid physical cache blocks, invalidates the virtual cache, then re-accesses the target cache block. The virtual tags should be updated with VALID and OWN from the physical cache, and RO and UP from the TLB status bits. The legal error codes for test 14, case 1 are:

```
01 - error in even block status after first access
02 - error in odd block status after first access
03 - error in even block status after second access
04 - error in odd block status after second access
```

The legal error codes for test 14, case 2 are:

```
05 - error in even block status after first access
06 - error in odd block status after first access
07 - error in even block status after second access
08 - error in odd block status after second access
```

The legal error codes for test 14, case 3 are:

```
09 - error in even block status after first access
0a - error in odd block status after first access
0b - error in even block status after second access
0c - error in odd block status after second access
```

The legal error codes for test 14, case 4 are:

```
0d - error in even block status after first access
0e - error in odd block status after first access
0f - error in even block status after second access
10 - error in odd block status after second access
```

11.1.21 Test 15 - Virtual Tag Odd Block Revalidation Test

Virtual tag re-validations occur when there is a physical cache hit and a virtual cache miss. This test creates valid physical cache blocks, invalidates the virtual cache, then re-accesses the target cache block. The virtual tags should be updated with VALID and OWN from the physical cache, and RO and UP from the TLB status bits. The legal error codes for test 15, case 1 are:

```
01 - error in even block status after first access
02 - error in odd block status after first access
03 - error in even block status after second access
04 - error in odd block status after second access
```

The legal error codes for test 15, case 2 are:

```
05 - error in even block status after first access
06 - error in odd block status after first access
07 - error in even block status after second access
08 - error in odd block status after second access
```

The legal error codes for test 15, case 3 are:

```
09 - error in even block status after first access
0a - error in odd block status after first access
0b - error in even block status after second access
0c - error in odd block status after second access
```

The legal error codes for test 15, case 4 are:

```
0d - error in even block status after first access
0e - error in odd block status after first access
0f - error in even block status after second access
10 - error in odd block status after second access
```

11.1.22 Test 16 - Virtual Tag Even/Odd Block Revalidation Test

Virtual tag re-validations occur when there is a physical cache hit and a virtual cache miss. This test creates valid physical cache blocks, invalidates the virtual cache, then re-accesses the target cache block. The virtual tags should be updated with VALID and OWN from the physical cache, and RO and UP from the TLB status bits. The legal error codes for test 16, case 1 are:

```
01 - error in even block status after first access
02 - error in odd block status after first access
03 - error in even block status after second access
04 - error in odd block status after second access
```

The legal error codes for test 16, case 2 are:

```
05 - error in even block status after first access
06 - error in odd block status after first access
07 - error in even block status after second access
08 - error in odd block status after second access
```

The legal error codes for test 16, case 3 are:

```
09 - error in even block status after first access
0a - error in odd block status after first access
0b - error in even block status after second access
0c - error in odd block status after second access
```

The legal error codes for test 16, case 4 are:

```
0d - error in even block status after first access
0e - error in odd block status after first access
0f - error in even block status after second access
10 - error in odd block status after second access
```

11.1.23 Test 17 - Virtual Tag Odd/Even Block Revalidation Test

Virtual tag re-validations occur when there is a physical cache hit and a virtual cache miss. This test creates valid physical cache blocks, invalidates the virtual cache, then re-accesses the target cache block. The virtual tags should be updated with VALID and OWN from the physical cache, and RO and UP from the TLB status bits. The legal error codes for test 17, case 1 are:

```
01 - error in even block status after first access
02 - error in odd block status after first access
03 - error in even block status after second access
04 - error in odd block status after second access
```

The legal error codes for test 17, case 2 are:

```
05 - error in even block status after first access
06 - error in odd block status after first access
07 - error in even block status after second access
08 - error in odd block status after second access
```

The legal error codes for test 17, case 3 are:

```
09 - error in even block status after first access
0a - error in odd block status after first access
0b - error in even block status after second access
0c - error in odd block status after second access
```

The legal error codes for test 17, case 4 are:

```
0d - error in even block status after first access
0e - error in odd block status after first access
0f - error in even block status after second access
10 - error in odd block status after second access
```

11.1.24 Test 18 - Virtual Tag Block Invalidation Test

Virtual tag re-validations occur when there is a physical cache hit and a virtual cache miss. This test creates valid physical cache blocks, invalidates the virtual cache, then generates a TLB fault when the target block is re-accessed. In this case, the virtual tags should be updated with OWN from the physical cache, RO and UP from the TLB status bits as in the virtual block re-validation tests, but in this case the virtual VALID bit should be cleared (set to invalid). The legal error codes for test 18, case 1 are:

```
01 - No write protect fault was generated
02 - Write Protect Fault bit not set in Fault Cause Register
03 - Error in even block status after exception occurred
04 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 2 are:

```
05 - No write protect fault was generated
06 - Write Protect Fault bit not set in Fault Cause Register
07 - Error in even block status after exception occurred
08 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 3 are:

```
09 - No write protect fault was generated
0a - Write Protect Fault bit not set in Fault Cause Register
0b - Error in even block status after exception occurred
0c - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 4 are:

```
0d - No write protect fault was generated
0e - Write Protect Fault bit not set in Fault Cause Register
0f - Error in even block status after exception occurred
10 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 5 are:

```
11 - No write protect fault was generated
12 - Write Protect Fault bit not set in Fault Cause Register
13 - Error in even block status after exception occurred
14 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 6 are:

```
5 - No write protect fault was generated
16 - Write Protect Fault bit not set in Fault Cause Register
17 - Error in even block status after exception occurred
18 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 7 are:

```
19 - No write protect fault was generated
1a - Write Protect Fault bit not set in Fault Cause Register
1b - Error in even block status after exception occurred
1c - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 8 are:

```
1d - No write protect fault was generated
1e - Write Protect Fault bit not set in Fault Cause Register
1f - Error in even block status after exception occurred
20 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 9 are:

```
21 - No write protect fault was generated
22 - Write Protect Fault bit not set in Fault Cause Register
23 - Error in even block status after exception occurred
24 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 10 are:

```
25 - No write protect fault was generated
26 - Write Protect Fault bit not set in Fault Cause Register
27 - Error in even block status after exception occurred
28 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 11 are:

```
29 - No write protect fault was generated
2a - Write Protect Fault bit not set in Fault Cause Register
2b - Error in even block status after exception occurred
2c - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 12 are:

```
2d - No write protect fault was generated
2e - Write Protect Fault bit not set in Fault Cause Register
2f - Error in even block status after exception occurred
30 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 13 are:

```
31 - No write protect fault was generated
32 - Write Protect Fault bit not set in Fault Cause Register
33 - Error in even block status after exception occurred
34 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 14 are:

```
35 - No write protect fault was generated
36 - Write Protect Fault bit not set in Fault Cause Register
37 - Error in even block status after exception occurred
38 - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 15 are:

```
39 - No write protect fault was generated
3a - Write Protect Fault bit not set in Fault Cause Register
3b - Error in even block status after exception occurred
3c - Error in odd block status after exception occurred
```

The legal error codes for test 18, case 16 are:

```
3d - No write protect fault was generated
3e - Write Protect Fault bit not set in Fault Cause Register
3f - Error in even block status after exception occurred
40 - Error in odd block status after exception occurred
```

11.1.25 Test 19 - MMU Fault Test

This test verifies that all types of MMU exceptions can be generated. In addition, the auto-read sequence is verified to return the correct registers values. The 13 cases for test 19 follow:

Case 1: TMISS fault (TTVALID false)

This test case maps logical address patterns to physical address zero, but reads the TLB entries to make them invalid (clears TTVALID). The logical address pattern is then used as the address in ld instruction. The legal error codes test 19, case 1 are:

```
01 - Data exception was not generated
02 - FCR<TMISS> bit was not set
03 - PDEP register did not contain correct value when read through
     auto read space.
04 - FVAR register did not contain correct value when read through
     auto read space.
05 - FCR register was not cleared when the FVAR was read.
```

Case 2: TMISS fault (TTVALID true) AND (TMATCH0 = false)

This test case maps logical address zero to physical address zero to create a valid TLB entry in TLB location zero, then performs a ld instruction at logical address `0xaa000000`. The legal error codes test 19, case 2 are:

```
06 - Data exception was not generated
07 - FCR TMISS fault bit was not set
08 - PDEP register did not contain correct value when read through
     auto read space.
09 - FVAR register did not contain correct value when read through
     auto read space.
0a - FCR register was not cleared when the FVAR was read.
```

Case 3: TMISS fault (TTVALID true) AND (TMATCH0 = false)

This is the same as case 2 except that `0x55000000` is used for the logical address for the ld instruction. The legal error codes test 19, case 3 are:

```
0b - Data exception was not generated
0c - FCR TMISS fault bit was not set
0d - PDEP register did not contain correct value when read through
     auto read space.
0e - FVAR register did not contain correct value when read through
     auto read space.
0f - FCR register was not cleared when the FVAR was read.
```

Case 4: UPF fault (TLBUP true)

This test case maps logical address 0x55555555 to physical address 0 and sets the UP bit (user protect) in the TLB, then accesses logical address 0x55555555 through user data space (ASI=10) to cause a UP fault. The legal error codes test 19, case 4 are:

```
10 - Data exception was not generated
11 - FCR UPF bit was not set
12 - PDEP register did not contain correct value when read through
     auto read space.
13 - FVAR register did not contain correct value when read through
     auto read space.
14 - FCR register was not cleared when the FVAR was read.
```

Case 5: UPF fault (TLBUP true)

This is the same as case 4 except logical address 0xaaaaaaaa is used. The legal error codes test 19, case 5 are:

```
15 - Data exception was not generated
16 - FCR UPF bit was not set
17 - PDEP register did not contain correct value when read through
     auto read space.
18 - FVAR register did not contain correct value when read through
     auto read space.
19 - FCR register was not cleared when the FVAR was read.
```

Case 6: UPF fault (FE space)

This test case accesses logical address 0xff555555 through user data space (ASI=10) to cause a User Protection fault. The legal error codes test 19, case 6 are:

```
1a - Data exception was not generated
1b - FCR UPF bit was not set
1c - PDEP register did not contain correct value when read through
     auto read space.
1d - FVAR register did not contain correct value when read through
     auto read space.
1e - FCR register was not cleared when the FVAR was read.
```

Case 7: UPF fault (FE space)

This is the same as case 6 except logical address 0xfeaaaaaa is used. The legal error codes test 19, case 7 are:

```
1f - Data exception was not generated
20 - FCR UPF bit was not set
21 - PDEP register did not contain correct value when read through
     auto read space.
22 - FVAR register did not contain correct value when read through
     auto read space.
23 - FCR register was not cleared when the FVAR was read.
```

Case 8: WPF fault (TLBRO)

This test case maps logical address zero to physical address zero and sets the TLB RO bit. Performs st instruction to logical address zero to cause a WPF fault to occur. The legal error codes test 19, case 8 are:

```
24 - Data exception was not generated
25 - FCR WPF bit was not set
26 - PDEP register did not contain correct value when read through
     auto read space.
27 - FVAR register did not contain correct value when read through
     auto read space.
28 - FCR register was not cleared when the FVAR was read.
```

Case 9: WPF fault (TLBRO)

This is the same as case 8 except logical address `0xfdffffff` is mapped to physical address zero. The legal error codes test 19, case 9 are:

```
29 - Data exception was not generated
2a - FCR WPF bit was not set
2b - PDEP register did not contain correct value when read through
     auto read space.
2c - FVAR register did not contain correct value when read through
     auto read space.
2d - FCR register was not cleared when the FVAR was read.
```

Case 10: WPF fault (TLBIOB)

This test case maps logical address `0xfdffffff` to physical address 0 and sets the TLB IOB bit, then performs a st instruction to logical address `0xfdffffff` to cause a WPF fault to occur. The legal error codes test 19, case 10 are:

```
2e - Data exception was not generated
2f - FCR WPF bit was not set
30 - PDEP register did not contain correct value when read through
     auto read space.
31 - FVAR register did not contain correct value when read through
     auto read space.
32 - FCR register was not cleared when the FVAR was read.
```

Case 11: WPF fault (TLBIOB)

This is the same as case 10 except logical address zero is mapped to physical address 0. The legal error codes test 19, case 11 are:

```
33 - Data exception was not generated
34 - FCR WPF bit was not set
35 - PDEP register did not contain correct value when read through
     auto read space.
36 - FVAR register did not contain correct value when read through
     auto read space.
37 - FCR register was not cleared when the FVAR was read.
```

Case 12: POF fault (TLBPVALID false)

This test case maps logical address 0x66666666 to physical address zero and clears the TLB page valid bit, then performs ld instruction to logical address 0x66666666 to cause a POF fault. The legal error codes test 19, case 12 are:

```
38 - Data exception was not generated
39 - FCR POF bit was not set
3a - PDEP register did not contain correct value when read through
     auto read space.
3b - FVAR register did not contain correct value when read through
     auto read space.
3c - FCR register was not cleared when the FVAR was read.
```

Case 13: POF fault (TLBPVALID false)

This is the same as case 12 except logical address 0x99999999 is mapped to physical address zero. The legal error codes test 19, case 13 are:

```
3d - Data exception was not generated
3e - FCR POF bit was not set
3f - PDEP register did not contain correct value when read through
     auto read space.
40 - FVAR register did not contain correct value when read through
     auto read space.
41 - FCR register was not cleared when the FVAR was read.
```

11.1.26 Test 1a - Timeout Fault Test

This test verifies that the timeout logic on the System Board is functional, that the Kbus Address lines are good and that the timeout detection logic in the bus watcher section of the CPU Board is functional. This is the first test which generates Kbus cycles. The legal error codes test 1a are:

```
01 - Data fault exception was not generated
02 - FCR TOFIO bits was not set
03 - FVAR register did not contain correct logical RIO address.
04 - FCR register was not cleared when the FVAR was read.
05 - FPAR register did not contain correct physical RIO adddress.
```

11.1.27 Test 1b - Slot Probe and Configuration Test

This test probes each slot of the system by performing ID space reads and determines the board types which occupy each slot. In the following error codes, the *X* represents the slot number of the target board.

```
0X - Exception other than Data Fault occurred during ID SPACE
     read of slot X.
1X - Data exception occurred during initial probe, but FCR TOFIO
     was not set.
2X - FVAR contained incorrect logical RIO address.
3X - FCR not cleared after reading FVAR.
4X - Unrecognizeable board type code read from slot X.
5X - Data exception fault occurred during ID space read after valid
     board was previously located in slot X.
6X - Data exception fault occurred during RIO read of optional header
     in IDPROM on board in slot X.
7X - Data fault exception occurred during RIO read of graphics minor
     board number from IDPROM in slot X.
8X - Data fault exception occurred during RIO read of device identifier
     string from IDPROM in slot X.
9X - Data fault exception occurred during RIO read of Memory Board size
         from IDPROM in slot X.
ax - Zero size parameter read from IDPROM on Memory Board in slot X.
bx - Invalid Memory Board size read from IDPROM in slot X.
         Not an even 16 Mbyte multiple.
c0 - System Board count is not 1 (0 or more than 1).
c1 - No Memory Boards were located.
c2 - No CPU Boards were located.
d0 - Data exception occurred reading EAROM BOOTMODE variable.
```

11.1.28 Test 1c - IDPROM Checksum Test

This test examines the configuration information obtained from the Slot Probe and configuration test and for each board identified, performs an IDPROM checksum test. The legal error codes for test 1c are:

```
1X - Data exception fault occurred while reading IDPROM size from
     the board in slot X.
2X - Zero size field for IDPROM on board in slot X.
3X - Data exception fault occurred while performing checksum on
     IDPROM on board in slot X.
4X - IDPROM checksum error for board in slot X.
5X - Data exception fault occurred while reading the optional header
     field of the IDPROM on board in slot X.f1
```

11.1.29 Test 1d - Master/Slave CPU Determination Test

This test determines which CPU in the system is to become the master CPU when multiple CPUs exist. The legal error codes for test 1d are:

```
10    - Data exception fault occurred while reading the
        CPUSTAT register
20    - Data exception fault occurred while writing CPUSTAT
        register of slave CPU Board.
0x10 - Data fault occurred accessing own cpustat register
0x2x - Data fault occurred accessing cpustat register of CPU in slot "X"
0x30 - Data fault occurred accessing EAROM
0x4x - Data fault occurred accessing CPUHR register of CPU in slot "X"
```

11.1.30 Test 1e - Bus Watcher Tag Reset Test

The legal error codes for test 1e are:

```
01 - Match, Own or Valid status error after initial write of tags
     and status.
02 - Match, Own or Valid status error after reset of Own and Valid
     status bits.
```

11.1.31 Test 1f - Bus Watcher Tag RAM Addressing Test

This test verifies the address lines for the Bus Watcher tag and status RAMs. The legal error codes for test 1f, case 1 are:

```
01 - Bus watcher tag match status error on read of tag location
     other than zero.
02 - Bus watcher tag match status error on read of tag location
     zero.
```

The test is then repeated using locations corresponding to a single address bit off (0xffc0, 0xffa0..., 0x7fe0), and address 0xffe0 is written with zero.

The legal error codes for this test 1f, case 2 are:

```
03 - Bus watcher tag match status error on read of tag location
     other than 0xffe0.
04 - Bus watcher tag match status error on read of tag location
     0xffe0.
```

11.1.32 Test 20 - Bus Watcher Tag Comparitors Test

This is a test of the Bus Watcher Tag match detection logic. There are two test cases. Both test cases use bus watcher tag location zero to contain the test patterns. The legal error codes for test 10, case 1 are:

```
01 - PM0 match status error
02 - PM1 match status error
```

The legal error codes for test 20, case 2 are:

```
03 - PM0 match status error
04 - PM1 match status error
```

11.1.33 Test 21 - Bus Watcher Tag RAM Address and Data Test

This is an addressing and data test for the bus watcher tag RAMs. The legal error codes for test 21 are:

```
01 - PM0/PM1/VAL/OWN status error on first read of forward pass
02 - PM0/PM1/VAL/OWN status error on second read of forward pass
03 - PM0/PM1/VAL/OWN status error on first read of reverse pass
04 - PM0/PM1/VAL/OWN status error on second read of reverse pass
```

11.1.34 Test 22 - Memory Board Base Address and Enable Register Test

This is a test for the Base Address Register and Enable Register on each installed Memory Board. Part 1 is the test of the Base Address register. Part 2 is the test of the Enable register. The legal error codes for test 22 are:

```
1X - Data exception fault occurred writing base address register of
     Memory Board in slot X.
2X - Data exception fault occurred reading base address register of
     Memory Board in slot X.
3X - Data miscompare error for base address register on Memory Board
     in slot X.
4X - Data exception fault occurred writing enable register of Memory
     Board in slot X.
5X - Data exception fault occurred reading enable register of Memory
     Board in slot X.
6X - Data miscompare error for enable register on Memory Board
     in slot X.
```

11.1.35 Test 23 - Memory Board Uniqueness Test

This test verifies that all installed Memory Boards can be accessed independently of all others.

This is the first test which attempts to write and read memory and thereby test the bus watchers ability to perform Kbus transactions other than RIO types. The legal error codes for test 23 are:

01-06: Indicates an exception occurred on the initial "stb" instruction. The error code is the slot number of the target Memory Board.

11-16: Indicates that a read of the data cached on the initial "stb" is not readable from the cache. The low nibble of the error code is the slot number.

21-26: Indicates an exception occurred on the flush operation. This is the instruction which causes the block flush back to memory. The low nibble of the error code is the slot number.

31-36: Indicates an exception occurred on the re-read of target byte. The low nibble of the error code is the slot number.

41-46: Indicates a data error on the re-read on target byte. This is the instruction which causes the target block to be re-cached and supplied to the CPU. The low nibble of the error code is the slot number.

11.1.36 Test 24 - Memory Board Address Uniqueness Test

This test verifies the uniqueness of the upper bits of the memory address. The legal error codes for test 24, case 1 are:

01-06: Indicates that the Memory Board responded when it should not have. The base address register of the target board is set to 0x00 and it responded to some other board address. The low nibble of the error code is the slot number of the target Memory Board.

11-16: Indicates that the wrong exception type occurred when the Memory Board was read. The expected exception vector is 9. The low nibble of the error code is the slot number of the target Memory Board.

21-26: FCR <TOFM> bit was not set when exception occurred. The low nibble of the error code is the slot number of the target memory board.

31-36: FVAR register did not contain the correct address when the exception occurred. The low nibble of the error code is the slot number of the target memory board.

41-46: FCR was not cleared when FVAR was read. The low nibble of the error code is the slot number of the target memory board.

The test is repeated using 0xff in the Base Address register and a walking zero pattern on the upper significant bits of the address. The legal error codes for test 24, case 2 are:

51-56: Indicates that the Memory Board responded when it should not have. The base address register of the target board is set to 0x00 and it responded to some other board address. The low nibble of the error code is the slot number of the target Memory Board.

61-66: Indicates that the wrong exception type occurred when the Memory Board was read. The expected exception vector is 9. The low nibble of the error code is the slot number of the target Memory Board.

71-76: FCR <TOFM> bit was not set when exception occurred. The low nibble of the error code is the slot number of the target memory board.

81-86: FVAR register did not contain the correct address when the exception occurred. The low nibble of the error code is the slot number of the target memory board.

91-96: FCR was not cleared when FVAR was read. The low nibble of the error code is the slot number of the target memory board.

11.1.37 Test 25 - Memory Board Addressing Test

This test verifies that each installed Memory Board can respond to all 256 unique Memory Board addresses (0x00 through 0xff). The legal error codes for test 25 are:

01-06: Indicates that an exception occurred when the board address was written. The low nibble of the error code is the slot number of the target Memory Board.

11-16: Indicates that an exception occurred when flushing the target block back to memory. The low nibble of the error code is the slot number of the target Memory Board.

21-26: Indicates that an exception occurred when the board address was read. The low nibble of the error code is the slot number of the target Memory Board.

31-36: Indicates the wrong data was returned from the target Memory Board. The low nibble of the error code is the slot number of the target Memory Board.

11.1.38 Test 26 - Memory Board Block Addressability Test

This test verifies the uniqueness of the address lines on each installed Memory Board. The 16 Mbyte Memory Board address is broken down as follows:

```
0x00000000 - 0x007fffe0: Low 8 Mbyte Bank
0x00800000 - 0x00ffffe0:  High 8 Mbyte Bank
```

The legal error codes for test 26 are:

```
0X - Data fault exception occurred on ld instruction using walking
     0/1 address from Memory Board in slot X
1X - Data miscompare occurred on ld instruction using walking 0/1
     address from Memory Board in slot X
2X - Data fault exception occurred on ld instruction using all
     zeroes/ones address from Memory Board in slot X
3X - Data miscompare occurred on ld instruction using all zeroes/ones
     address from Memory Board in slot X

4X - Data fault exception occurred on st instruction using walking
     0/1 address on Memory Board in slot X
5X - Data fault exception occurred on ld instruction after data from
     Memory Board X was already cached.
6X - Data miscompare occurred on ld instruction after data from
     Memory Board X was already cached.

7X - Data fault exception occurred on st instruction using all
     zeroes/ones address on Memory Board in slot X
8X - Data fault exception occurred on ld instruction after data from
     Memory Board X was already cached.
9X - Data miscompare occurred on ld instruction after data from
     Memory Board X was already cached.
```

11.1.39 Test 27 - Memory Board RAM Addressing and Data Test

This is an addressing and data test for the first 1 Mbyte of memory. Only the first 1 Mbyte of memory is tested to keep execution time during power-up selftest to a minimum. Legal error codes for test 27 are:

```
1X - Data fault exception occurred during write of memory
     with initial data pattern.
2X - Data fault exception occurred on first read of forward pass
3X - Data miscompare occurred on first read of forward pass
4X - Data fault exception occurred during flush of target
     memory block back to memory during forward pass.
5X - Data fault exception occurred on second read of forward pass
6X - Data miscompare occurred on second read of forward pass
7X - Data fault exception occurred on first read of reverse pass
8X - Data miscompare occurred on first read of reverse pass
9X - Data fault exception occurred during flush of target
     memory block back to memory during reverse pass.
aX - Data fault exception occurred on second read of reverse pass
bX - Data miscompare occurred on second read of reverse pass
```

11.1.40 Test 28 - Cache Fill-Flush Test

This test fills the entire 64 Kbytes of cache RAM with the first 64 Kbytes of the bootrom code. Next, the second 64 Kbytes of bootrom code is then written to the cache. This should displace the contents of the cache out to physical memory. The legal error codes for test 28 are:

```
01 - Data fault exception occurred on st instruction to
     memory while loading cache with first 64 Kbytes of
     bootrom code.
02 - Data fault exception occurred on st instruction to
     memory while loading cache with second 64 Kbytes of
     bootrom code.
03 - Data fault exception occurred on ld instruction from
     memory while verifying first 64 Kbytes of data.
04 - Data miscompare occurred while verifying first 64 Kbytes
     of data.
05 - Data fault exception occurred on ld instruction from
     memory while verifying second 64 Kbytes data.
06 - Data miscompare occurred while verifying second 64 Kbytes
     of data.
```

11.1.41 Test 29 - Virtual Fault Cache Corruption Test

This test verifies that exceptions which occur due to cache writes do not corrupt the cache data. There are eight test cases. The legal error codes for the eight cases in test 29 are:

Case 1: Single precision misaligned store exception to FF space

```
01 - Address Alignment fault did not occur on st to misaligned
     word address.
02 - First word of cache line corrupted on st to misaligned word
     address.
03 - Second word of cache line corrupted on st to misaligned word
     address.
```

Case 2: Double precision misaligned store exception to FF space

```
04 - Address Alignment fault did not occur on std to misaligned
     double-word address.
05 - First word of cache line corrupted on std to misaligned
     double-word address.
06 - Second word of cache line corrupted on std to misaligned
     double-word address.
```

Case 3: Single precision misaligned store operation with MMU enabled

```
07 - Address Alignment fault did not occur on st to misaligned
     word address.
08 - First word of cache line corrupted on st to misaligned word
     address.
09 - Second word of cache line corrupted on st to misaligned word
     address.
```

Case 4: Double precision misaligned store operation with MMU enabled

```
0a - Address Alignment fault did not occur on std to misaligned
     double-word address.
0b - First word of cache line corrupted on std to misaligned
     double-word address.
0c - Second word of cache line corrupted on std to misaligned
     double-word address.
```

Case 5: Single precision read only store exception

```
0d - Data fault exception did not occur on st to page marked read
     only in TLB.
0e - First word of cache line corrupted on st to page marked read
     only in TLB.
0f - Second word of cache line corrupted on st to page marked read
     only in TLB.
```

Case 6: Double precision read only store exception

```
10 - Data fault exception did not occur on std to page marked read
     only in TLB.
11 - First word of cache line corrupted on std to page marked read
     only in TLB.
12 - Second word of cache line corrupted on std to page marked read
     only in TLB.
```

Case 7: Single precision TLB miss store exception

```
13 - Data fault exception did not occur on st to page marked as
     invalid in TLB.
14 - First word of cache line corrupted on st to page marked as
     invalid in TLB.
15 - Second word of cache line corrupted on st to page marked as
     invalid in TLB.
```

Case 8: Double precision TLB miss store exception

```
16 - Data fault exception did not occur on std to page marked as
     invalid in TLB.
17 - First word of cache line corrupted on std to page marked as
     invalid in TLB.
18 - Second word of cache line corrupted on std to page marked as
     invalid in TLB.
```

11.1.42 Test 2a - Corrupted Block RAM Addressing and Data Test

This is an addressing and data test for the Corrupted Block RAM. The legal error codes for test 2a are:

```
01 - Corrupted Bit not zero on first read of forward pass
02 - No memory timeout fault was generated to target block address.
03 - FVAR does not contain the correct FF space address after memory
     timeout fault.
04 - Corrupted Bit did not toggle to one after memory timeout fault.
```

11.1.43 Test 2b - Corrupted Block Flush Inhibit Test

This test verifies that cache transactions which reference a corrupted block result in a Kbus timeout. The legal error codes for test 2b are:

```
01 - No memory timeout fault was generated when Memory Boards
     disabled.
02 - No memory timeout fault was generated on reference to corrupted
     block.
03 - FCR TOFM bit not set
04 - FVAR does not contain the correct logical address
05 - FPAR does not contain the correct physical address
```

11.1.44 Test 2c - Cache Purge Transaction Test

This test verifies that the cache and bus watcher logic can correctly perform a cache purge operation. The legal error codes for test 2c are:

```
01 - Data returned on ld from logical address 0x4000 is not 0xff010000.
02 - Status of logical block 0x4000 is either invalid, unowned, or
     both invalid and unowned.
03 - Data read from logical block 0x2000 is not zero.
04 - Status of logical block 0x2000 is not invalid and unowned.
05 - FPAR does not contain 0x10000.
```

11.1.45 Test 2d - Cache Purge/Flush Transaction Test

This test verifies that the cache and bus watcher logic can correctly perform a cache purge and flush operation. Legal error codes for test 2d are:

```
01 - Data returned on ld from logical address 0x4020 is not 0xffffffdf.
02 - Status of logical block 0x4020 is either invalid, unowned, or
     both invalid and unowned.
03 - Data returned on ld from logical address 0x4024 is not 0xff010024.
04 - Status of logical block 0x2020 is not invalid and unowned.
05 - FPAR does not contain 0x10020.
06 - Data returned on re-read of logical address 0x2020 is not 0x20.
07 - Data returned on re-read of logical address 0x14020 is not 0x4020.
```

11.1.46 Test 2e - Virtual Cache Block Replacement Test

This test exercises the cache and bus watcher cache block purge and flush logic by performing writes and reads to common physical addresses through all different logical addresses including FF space. Legal error codes for test 2e are:

```
01 - Data fault exception occurred during creation of the valid
     owned and dirty cache blocks.
02 - Data fault exception occurred during read of target physical
     cache block.
03 - Physical data read through logical address does not match
     expected physical address data.
04 - Data fault exception occurred during read of target physical
     cache blocks using FF space addresses.
05 - Physical data read through FF space address does not match
     expected physical address data.
```

11.1.47 Test 2f - ECC Write/Read Test

This test verifies the ECC data path to and from each installed Memory Board. The legal error codes for test 2f are:

```
01 - ECCS or data fault exception occurred on store of data
     pattern with ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of test
     pattern back to memory.
03 - Single Bit ECC exception generated on re-read of test
     pattern from memory with ECC checking enabled.
04 - Multi-bit ECC exception generated on re-read of test
     pattern from memory with ECC checking enabled.
05 - Exception other than ECCS or ECCM generated on re-read
     of test pattern from memory with ECC checking enabled.
06 - Data error in first word of cache line 0
07 - Data error in second word of cache line 0
08 - Data error in first word of cache line 1
09 - Data error in second word of cache line 1
0a - Data error in first word of cache line 2
0b - Data error in second word of cache line 2
0c - Data error in first word of cache line 3
0d - Data error in second word of cache line 3
```

11.1.48 Test 30 - ECC Single Bit Correction to 1 Test

This test verifies that the ECC data correction logic can correct a bit from a zero to a one for all 64 data bit positions. The test is performed independly for each all four cache lines. The legal error codes for test 30 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.1.49 Test 31 - ECC Single Bit Correction to 0 Test

This test verifies that the ECC data correction logic can correct a bit from a one to a zero for all 64 data bit positions. The test is performed independly for each all 4 cache lines. The legal error codes for test 31 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.1.50 Test 32 - ECC Single Bit Checkbyte Error Test

This test verifies that single bit errors in the checkbyte are detectable and causes no cache line data corruption. The legal error codes for test 32 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.1.51 Test 33 - ECC Multibit Error Detection Test

This test verifies that all syndrome values which map to a two bit or more than two bit error results in the generation of a multibit ECC exception. The legal error codes for test 33 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than Data Fault exception was generated on
     re-read of target cache block.
05 - FCR ECCM bit not set.
06 - FVAR register contains incorrect address.
07 - FCR not cleared after read of FVAR.
```

11.1.52 Test 34 - ECC RAM Addressing and Data Test

This is an addressing and data test for the first megabyte of ECC memory. Only the first megabyte of memory are tested to keep execution time during power-up selftest to a minimum. Legal error codes for test 34 are:

```
1X - Data fault exception occurred during write of memory
     with initial data pattern.
2X - ECCS or data fault exception occurred on first read of
     forward pass
3X - Data miscompare occurred in upper 32 bits of cache line
     during forward pass
4X - Data miscompare occurred in lower 32 bits of cache line
     during forward pass
5X - Data fault exception occurred during flush of target
     memory block back to memory during forward pass.
6X - ECCS or data fault exception occurred on second read of
     forward pass
7X - Data miscompare occurred in upper 32 bits of cache line
     during forward pass
8X - Data miscompare occurred in lower 32 bits of cache line
     during forward pass
9X - ECCS or data fault exception occurred on first read of
     reverse pass
aX - Data miscompare occurred in upper 32 bits of cache line
     during reverse pass
bX - Data miscompare occurred in lower 32 bits of cache line
     during reverse pass
cX - Data fault exception occurred during flush of target
     memory block back to memory during reverse pass.
dX - ECCS or data fault exception occurred on second read of
     reverse pass
eX - Data miscompare occurred in upper 32 bits of cache line
     during reverse pass
fX - Data miscompare occurred in lower 32 bits of cache line
     during reverse pass
```

11.1.53 Test 35 - FPU Register Load/Store Test

This test verifies the primary interaction between the floating point unit and the memory system by performing a write/read test on one of the floating point register pairs. There are two test cases, one for single precision values and one for double-precision values. This test as well as all other floating point unit tests are only executed if the floating point unit is available on the CPU Board. Legal error codes for test 35 are:

```
01 - After attempting to clear the QNE bit on the FPU state
     register, the queue (FQ) is still not empty.
02 - Write read error for single precision load/store.
03 - Write read error for double precision load/store (even register).
04 - Write read error for double precision load/store (odd register).
```

11.1.54 Test 36 - FPU State Register Test

The FPU state register (FSR) contains FPU mode and status information. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 36 are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - FSR write read error.
```

11.1.55 Test 37 - FPU Add/Multiply/Divide Test

This test verifies the path between the FPC and the floating point arithmetic units on the FPC/FALU and the FPC/FMULT interfaces. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 37 are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - Incorrect single precision addition result.
03 - Incorrect single precision multiplication result.
04 - Incorrect double precision addition result (even register).
05 - Incorrect double precision addition result (odd register).
06 - Incorrect double precision multiplication result (even register).
07 - Incorrect double precision multiplication result (odd register).
08 - Incorrect single precision division result.
09 - Incorrect double precision division result (even register).
0a - Incorrect double precision division result (odd register).
0b - FPU did not handled operand dependency correctly.
```

11.1.56 Test 38 - FPU Queue Test

The FPU queue (FQ) keeps tracks of floating point operations that are pending by the FPU when a floating point fp_exception trap occurs. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for this test are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - FSR write read error while setting TEM (NV) bit.
03 - FPU fp_exception trap did not occur when expected.
04 - FSR QNE bit is clear when it should be set.
05 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
```

11.1.57 Test 39 - FPU Exceptions Test

There are two floating point trap types that are generated by the FPU hardware. These are: fp_disabled and fp_exception. The FPU generates four types of exception traps:

1. FPC sequence error exception
2. Unimplemented floating point instruction exception (Not checked by this test. All instructions are implemented.)
3. Unfinished floating point instruction exception
4. IEEE exception

IEEE exceptions are classified as follows:

1. Invalid
2. Overflow
3. Underflow
4. Division by zero
5. Inexact

This test verifies that the FPU generates these traps and exceptions properly by performing test cases for each type. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for the 13 cases in test 39 are:

Test Case 1: fp_disabled trap

```
01 - FPU fp_disabled trap did not occur when expected.
```

Test Case 2: fp_exception IEEE-Invalid while enabled

```
02 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
03 - FPU fp_exception trap did not occur when expected (IEEE-Invalid).
04 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Invalid.
```

Test Case 3: fp_exception IEEE-Invalid while disabled

```
05 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
06 - FPU fp_exception trap occured while traps were disabled (IEEE Invalid).
07 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Invalid.
```

Test Case 4: fp_exception IEEE-Overflow while enabled

```
08 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
09 - FPU fp_exception trap did not occur when expected (IEEE-Overflow).
0a - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Overflow.
```

Test Case 5: fp_exception IEEE-Overflow while disabled

```
0b - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
0c - FPU fp_exception trap occured while traps were disabled (IEEE Overflow).
0d - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Overflow.
```

Test Case 6: fp_exception IEEE-Underflow while enabled

```
0e - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
0f - FPU fp_exception trap did not occur when expected (IEEE-Underflow).
10 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Underflow.
```

Test Case 7: fp_exception IEEE-Underflow while disabled

```
11 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
12 - FPU fp_exception trap occured while traps were disabled (IEEE Underflow)
13 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Underflow.
```

Test Case 8: fp_exception IEEE-Inexact while enabled

```
14 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
15 - FPU fp_exception trap did not occur when expected (IEEE-Inexact).
16 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Inexact.
```

Test Case 9: fp_exception IEEE-Inexact while disabled

```
17 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
18 - FPU fp_exception trap occured while traps were disabled (IEEE Inexact).
19 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Inexact.
```

Test Case 10: fp_exception IEEE-Divide-By-Zero while enabled

```
1a - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
1b - FPU fp_exception trap did not occur when expected (IEEE-Divide-by-Zero).
1c - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Divide-by-Zero
```

Test Case 11: fp_exception IEEE-Divide-By-Zero while disabled

```
1d - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
1e - FPU fp_exception trap occured while traps were disabled.
1f - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Divide-By-Zero.
```

Test Case 12: fp_exception Sequence-Error

```
20 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
21 - FPU fp_exception trap did not occur when expected (IEEE-Divide-by-Zero).
22 - FPU fp_exception trap did not occur when expected (SEQUENCE).
23 - FPU fp_exception trap ocurred, but FSR FTT bits are not set
     for SEQUENCE.
```

Test Case 13: fp_exception Unfinished-Floating-Point-Instruction

```
24 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
25 - FPU fp_exception trap did not occur when expected (UNFINISHED_FPOP).
26 - FPU fp_exception trap ocurred, but FSR FTT bits are not set
     for UNFINISHED_FPOP
```

11.1.58 Test 3a - FPU Condition Codes Test

Floating point compares (FCMPS) and floating point condition (FBfcc) instructions interlock on the floating point condition codes. This condition codes are maintained by the FPU in the FSR. The condition codes supported by the FPU are:

1. Equal Relation
2. Greater-Than Relation
3. Less-Than Relation
4. Unordered Relation

This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 3a are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
```

Test Case 1: Equal Relation when A == B

```
02 - CC should reflect an equal relation, causing FBE
     instruction to fail.
03 - FSR FCC bits not reflecting an equal relation when expected.
```

Test Case 2: Equal Relation when A != B

```
04 - CC should not reflect an equal relation, causing FBE
     instruction to fail.
05 - FSR FCC bits reflecting an equal relation when not expected.
```

Test Case 3: Greater-Than Relation when A > B

```
06 - CC should reflect a greater-than relation, causing FBG
     instruction to fail.
07 - FSR FCC bits not reflecting a greater-than relation when expected.
```

Test Case 4: Greater-Than Relation when A < B

```
08 - CC should not reflect a greater_than relation, causing FBG
     instruction to fail.
09 - FSR FCC bits reflecting a greater-than relation when not expected.
```

Test Case 5: Less-Than Relation when A < B

```
0a - CC should reflect a less-than relation, causing FBL
     instruction to fail.
0b - FSR FCC bits not reflecting a less-than relation when expected.
```

Test Case 6: Less-Than Relation when A > B

```
0c - CC should not reflect a less_than relation, causing FBL
     instruction to fail.
0d - FSR FCC bits reflecting a less-than relation when not expected.
```

Test Case 7: Unordered Relation when A unordered, B ordered

```
0e - CC should reflect an unordered relation, causing FBU
     instruction to fail.
0f - FSR FCC bits not reflecting an unordered relation when expected.
```

Test Case 8: Unordered Relation when A & B ordered

```
10 - CC should not reflect an unordered relation, causing FBU
     instruction to fail.
11 - FSR FCC bits reflecting an unordered relation when not expected.
```

11.1.59 Test 3b - FPU Fast-Mode Enable Bit Test

When the FPU is in fast mode, the operations on denormalized numbers should not generate a fp_exception trap. While this is the case in fast mode, the case for IEEE mode is that it will result in an exception. This test enables the FPU to operate in fast mode by setting the corresponding bit in the FSR. A floating point operation is then performed on a denormalized number in order to verify that a trap does not occur. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 3b are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - FAST mode was enable, but got an exception while operating
     on a denormalized number.
```

11.1.60 Test 3c - Frame Buffer Test

This is an addressing and data test for the frame buffer on the Graphics Boards configured into the system. This test supports the following Graphic Boards:

1. Monochrome Graphics Board (board-minor type = 0)
   Frame buffer size : 256 Kbytes
2. Color Graphics Board (board-minor type = 0x01)
   Overlay Plane 1 size : 128 Kbytes
   Overlay Plane 2 size : 128 Kbytes
   Image Plane size : 2 Mbytes

If an unknown Graphics Board is found in the system the test will not be performed. Legal error codes for test 3c are:

```
0X - Data fault exception occurred during writing to board
     control register for board in slot X
1X - Data fault exception occured during writing to board space
     register for board in slot X
2X - Accumulated miscompares during forward pass exceeded error limit
     on overlay planes/frame-buffer for board in slot X
3X - Accumulated miscompares during reverse pass exceeded error limit
     on overlay planes/frame-buffer for board in slot X
4X - Acccumulated miscompares during forward pass exceeded error limit
     on image plane for board in slot X
5X - Acccumulated miscompares during reverse pass exceeded error limit
     on image plane for board in slot X
6X - Data fault exception occurred during write of initial data
     pattern to frame buffer in slot X
7X - Data fault exception occurred on first read of forward pass
     for frame buffer in slot X
8X - Data fault exception occurred during complement write of forward
     pass for frame buffer in slot X
9X - Data fault exception occurred on second read of forward pass
     for frame buffer in slot X
aX - Data fault exception occurred on first read of reverse pass
     for frame buffer in slot X
bX - Data fault exception occurred during complement write of reverse
     pass for frame buffer in slot X
cX - Data fault exception occurred on second read of reverse pass
     for frame buffer in slot X
dX - Data fault exception occurred during writing to COLOR board video
     control register for board in slot X
eX - Data fault exception occurred during reading of COLOR board video
     control register for board in slot X
fX - Write/read error occurred on COLOR board video control register
     for board in slot X
```

11.1.61 Test 3d - System Board Interrupt Generation Test

This is a test of the interrupt register on the System Board and the ability of the System Board to generate all 16 vectors when enabled after reset. Part 1 first reads RIO address `17030000` to disable transmission of interrupts, then writes RIO address `17030000` with incrementing test patterns from `0` to `0xff`. Each pattern is read back and verified. The legal error codes for test 3d, part 1 are:

```
01 - Data fault exception occurred on initial read to disable
     System Board interrupt register.
02 - Data fault exception occurred on write of pattern to
     System Board interrupt register.
03 - Data fault exception occurred on read of System Board
     interrupt register.
04 - Data pattern written does not match data pattern read from
     System Board interrupt register.
```

Part 2 initializes the System Board interrupt register with the directed bit set and the destination ID set the BID of the CPU Board. System Board interrupts are then enabled by reading address `17031000`. The test verifies that all 16 interrupt vectors are received correctly. Note that this test will fail if a system reset is not performed inbetween passes. The legal error codes for test 3d, part 2 are:

```
05 - Timeout waiting to receive first interrupt (vector 0x8f) from
     System Board.
06 - Exeption other than Serial Interrupt Controller occurred.
07 - Higher priority interrupt vector was received 256 times without
     receiving expected vector.
8X - Lower priority interrupt vector was received. Error code is the
     vector which was expected.
```

11.1.62 Test 3e - Serial Port Reset Test

This test verifies the reset state of both Z8530 SCC chips on the System Board, controlling the keyboard/mouse and serial ports A/B. The legal error codes for test 3e are:

```
10 - Data fault exception occurred during resetting of
     mouse port.
2X - Data fault exception occurred during resetting of port
     X
3X - Unexpected/Invalid reset state of portX
```

The ports are assigned the following port numbers:

```
Keyboard Port = 0
Mouse Port    = 1
Serial Port A = 2
Serial Port B = 3
```

11.1.63 Test 3f - Serial Port Internal Loopback Test

This test performs an internal loopback test of both Z8530 SCC chips on the System Board, controlling the keyboard/mouse and serial ports A/B. The legal error codes for test 3f are:

```
10 - Data fault exception occurred during resetting of mouse
     port
2X - Data fault exception occurred during resetting of port X
3X - Data fault exception occurred during programming of port X
4X - Receive error on port X
5X - Transmit error on port X
6X - Timeout  while waiting for Receive Character Available interrupt
     on port X
7X - Incorrect interrupt vector receive while waiting for Receive
     Character Available interrupt on port X
8X - Data miscompare (write/read) error on port X
9X - Receive Character Available interrupt pending bit is inactive
     (Z8530 RR2 Register) on port X
aX - Timeout while waiting for Transmit Buffer Empty interrupt on
     port X
bX - Incorrect interrupt vector receive while waiting for Transmit
     Buffer Empty interrupt on port X
cX - Transmit Buffer Empty interrupt pending bit is inactive
     (Z8530 RR2 register) on port X
```

The ports are assigned the follwing port numbers:

```
Keyboard Port = 0
Mouse Port    = 1
Serial Port A = 2
Serial Port B = 3
```

11.2 Series5 Test Descriptions

This section describes the system power-on self-tests that are used on Solbourne Series5 systems. In the these test descriptions, all test numbers and error codes are represented in two digit hex values. Refer to the *System Power-On Self-Test Manual* for more complete descriptions of these tests. Also refer to Appendix B, "Series5 Considerations" of that manual.

11.2.1 Test 01 - Bootrom Checksum Test

This test computes the checksum on the contents of the four, 27C512 EPROMS on the CPU Board that are used to contain the boot code and data. The expected checksum is burned into the EPROM when the ROMs are programed during manufacturing. Legal error codes for this test are:

```
00 - Checksum 0 incorrect        04 - Checksum 4 incorrect
01 - Checksum 1 incorrect        05 - Checksum 5 incorrect
02 - Checksum 2 incorrect        06 - Checksum 6 incorrect
03 - Checksum 3 incorrect        07 - Checksum 7 incorrect
```

11.2.2 Test 02 - Diagnostic RAM Addressing and Data Test

This is an addressing and data test for the diagnostic RAM on the CPU Board. The diagnostic RAM a two Kbyte static RAM which is accessed through alternate space (ASI) 0xe0 (see H/W description), and responds to every fourth address in the range 0 through 0x1ffc inclusive. Legal error codes for this test are:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
```

11.2.3 Test 03 - Control-Data Bus Test

This test reads the quiescent (undriven) state of the CPU's 32-bit, control data bus 64 Kbyte times and verifies that the bus is floats high (all ones). A common cause of failure for this test is that one of the bootrom outputs is sinking too much current and pulling the control data bus to the low state.

Control data bus format

31			0
byte 0	byte 1	byte 2	byte 3

Error encoding convention:

```
01 - byte 3 corrupted                09 - bytes 0 and 3 corrupted
02 - byte 2 corrupted                0a - bytes 0 and 2 corrupted
03 - bytes 2 and 3 corrupted         0b - bytes 0, 2, and 3 corrupted
04 - byte 1 corrupted                0c - bytes 0 and 1 corrupted
05 - bytes 1 and 3 corrupted         0d - bytes 0, 1 and 3 corrupted
06 - bytes 1 and 2 corrupted         0e - bytes 0, 1 and 2 corrupted
07 - bytes 1, 2, and 3 corrupted     0f - bytes 0, 1, 2 and 3 corrupted
08 - byte 0 corrupted
```

11.2.4 Test 04 - Control Registers Test

This test verifies that the PDBA register can be written and read. Aside from the interrupt registers (see test 08) this is the only other register that is write-readable. Legal error codes for this test case are:

```
01 - PDBA register write/read error
```

11.2.5 Test 05 - GTLB/MTRAN Bus Data Test

This test verifies that data path to the GTLB translation data and the PTE permissions bits are unique across the MTRAN bus. There are five cases for test 05, as follows:

1. Write ones to the GTLB at index 0 and zeroes at index 1. The data is read back and verified. The legal error codes for test 05, case 1 are:

   ```
   01 - The instruction GTLB physical address field at index 0
        does not contain the data that was written.
   02 - The instruction GTLB physical address field at index 1 does
        not contain the data that was written.
   ```

2. Walk a one across the status bits of the GTLB entry The legal error codes for test 05, case 2 are:

   ```
   03 - The data GTLB physical address field does not contain the
        data that was written.
   04 - The data GTLB tag and status field does not contain the data
        that was written.
   ```

3. Walk a one across the physical address field of the GTLB entry The legal error codes for test 05, case 3 are:

   ```
   05 - The data GTLB physical address field does not contain the
        data that was written.
   06 - The data GTLB tag and status field does not contain the data
        that was written.
   ```

4. Walk a zero across the status bits of the GTLB entry The legal error codes for test 05, case 4 are:

   ```
   07 - The data GTLB physical address field does not contain the
        data that was written.
   08 - The data GTLB tag and status field does not contain the data
        that was written.
   ```

5. Walk a zero across the physical address field of the GTLB entry The legal error codes for test 05, case 5 are:

   ```
   09 - The data GTLB physical address field does not contain the
        data that was written.
   0a - The data GTLB tag and status field does not contain the data
        that was written.
   ```

11.2.6 Test 06 - GTLB RAM Addressing and Data Test

This is a test of the physical address field of the GTLB RAMs. The legal error codes for test 06 are:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
```

11.2.7 Test 07 - ROM Addressing Test

This is a read test of FE space data through the PTRAN address multiplexor. Legal error codes for this test are:

```
01 - Data mismatch on first word read
02 - Data mismatch on second word read
```

11.2.8 Test 08 - Interrupt Registers Test

This is a write/read test of the interrupt registers. There are four test cases, one for each register tested. The registers tested are: Device ID Register (DIR), Interrupt Priority Register (IPR), Interrupt Transmit Register (IXR), and the Interrupt Pending Vector Register (IPV). Legal error codes for this test are:

```
01 - DIR register write read error
02 - IPR register write read error
03 - IXR register write read error
04 - IPV register write read error
```

11.2.9 Test 09 - Directed Interrupt Test

This test verifies that the CPU Board interrupt logic can send a directed interrupt to itself. There are two test cases:

1. Verifies that directed interrupts can be transmitted and received
2. Verifies that the interrupt receiver priority level (set in the IPR register) effectively inhibits interrupts from being received.

Legal error codes for test 09, case 1 are:

```
01 - Interrupt was never acknowledged (ITXC<gone> bit not set).
02 - Interrupt was acknowledged but no incorrect interrupt
     type was generated.
03 - Interrupt occurred, but the IRXC<P> bit was not set.
04 - Interrupt occurred, but the IPV register did not
     contain the transmitted vector.
```

Legal error codes for test 09, case 2 is:

```
05 - Interrupt was acknowledged (ITXC<gone> bit set).
06 - Unexpected interrupt was generated
```

11.2.10 Test 0a - GTLB TAG Addressing and Data Test

This is a test of the TAG, and status fields of the GTLB. The legal error codes for test 0a are:

```
01 - Tag or status error on first forward pass read
02 - Tag or status error on second forward pass read
03 - Tag or status error on first reverse pass read
04 - Tag or status error on second reverse pass read
```

11.2.11 Test 0b - GTLB Tag Match Test

This is a test of the GTLB Tag RAM chips. There are six cases for test 0b, as follows:

1. GTLB tag set to walking 1 pattern with logical address set to zero. The legal error codes for test 0b, case 1 are:

```
01 - GTLB tag RAM match error using instruction GTLB.
02 - GTLB tag RAM match error using data GTLB.
```

2. GTLB tag set to walking 0 pattern with logical address set to all ones. The legal error codes for test 0b, case 2 are:

```
03 - GTLB tag RAM match error using instruction GTLB.
04 - GTLB tag RAM match error using data GTLB.
```

3. GTLB tag set to zero with logical address set to walking 1 pattern. The legal error codes for test 0b, case 3 are:

```
05 - GTLB tag RAM match error using instruction GTLB.
06 - GTLB tag RAM match error using data GTLB.
```

4. GTLB tag set to all ones with logical address set to walking zero pattern. The legal error codes for test 0b, case 4 are:

```
07 - GTLB tag RAM match error using instruction GTLB.
08 - GTLB tag RAM match error using data GTLB.
```

5. GTLB tag set to all ones, clear GTLB tags with clear GTLB tags ASI, verify with logical address set to walking one pattern. The legal error codes for test 0b, case 5 are:

```
09 - GTLB tag RAM match error using instruction GTLB.
0a - GTLB tag RAM match error using data GTLB.
```

6. GTLB tag set to all ones, clear GTLB tags with clear GTLB/FTLB tags ASI, verify with logical address set to walking one pattern. The legal error codes for test 0b, case 6 are:

```
0b - GTLB tag RAM match error using instruction GTLB.
0c - GTLB tag RAM match error using data GTLB.
```

11.2.12 Test 0c - FTLB/TAGADD Bus Data Test

This test verifies that data path from the GTLB/MTRAN/FTLB input to the FTLB translation data and the PTE permissions bits are unique across the TAGADD bus to the FTIR. There are five cases for test 0c, as follows:

1. Walk a one across the status bits of the FTLB entry The legal error codes for test 0c, case 1 are:

```
01 - The instruction FTLB physical address field read from the
     FTIR does not contain the data that was written.
02 - The instruction FTLB tag and status field does not contain
     the data that was written.
```

2. Walk a one across the physical address field of the FTLB entry The legal error codes for test 0c, case 2 are:

```
03 - The instruction FTLB physical address field read from the
     FTIR does not contain the data that was written.
04 - The instruction FTLB tag and status field does not contain
     the data that was written.
```

3. Walk a zero across the status bits of the FTLB entry The legal error codes for test 0c, case 3 are:

```
05 - The instruction FTLB physical address field read from the
     FTIR does not contain the data that was written.
06 - The instruction FTLB tag and status field does not contain
     the data that was written.
```

4. Walk a zero across the physical address field of the FTLB entry The legal error codes for test 0c, case 4 are:

```
07 - The instruction FTLB physical address field read from the
     FTIR does not contain the data that was written.
08 - The instruction FTLB tag and status field does not contain
     the data that was written.
```

11.2.13 Test 0d - FTLB RAM Addressing and Data Test

This is a test of the physical address, TAG and status fields of the FTLB rams. The legal error codes for test 0d are:

```
01 - Data error on first forward pass read
02 - Tag or status error on first forward pass read
03 - Data error on second forward pass read
04 - Tag or status error on second forward pass read
05 - Data error on first reverse pass read
06 - Tag or status error on first reverse pass read
07 - Data error on second reverse pass read
08 - Tag or status error on second reverse pass read
```

11.2.14 Test 0e - FTLB Tag Match Test

This is a test of the FTLB Tag RAM chips. The test program loads a series of patterns into the tag portion of the FTLB then performs a series of TIR reads to verify that the FTLB tag RAMs work correctly (see Appendix B for more information on TIR reads). There are six cases for test 0e, as follows:

1. FTLB tag set to walking 1 pattern with logical address set to zero. The legal error codes for test 0e, case 1 are:

```
01 - FTLB tag RAM match error using instruction FTLB.
02 - FTLB tag RAM match error using data FTLB.
```

2. FTLB tag set to walking 0 pattern with logical address set to all ones. The legal error codes for test 0e, case 2 are:

```
03 - FTLB tag RAM match error using instruction FTLB.
04 - FTLB tag RAM match error using data FTLB.
```

3. FTLB tag set to zero with logical address set to walking 1 pattern. The legal error codes for test 0e, case 3 are:

```
05 - FTLB tag RAM match error using instruction FTLB.
06 - FTLB tag RAM match error using data FTLB.
```

4. FTLB tag set to all ones with logical address set to walking zero pattern. The legal error codes for test 0e, case 4 are:

```
07 - FTLB tag RAM match error using instruction FTLB.
08 - FTLB tag RAM match error using data FTLB.
```

5. FTLB tag set to all ones, clear FTLB tags with clear FTLB tags ASI, verify with logical address set to walking one pattern. The legal error codes for test 0e, case 5 are:

```
09 - FTLB tag RAM match error using instruction FTLB.
0a - FTLB tag RAM match error using data FTLB.
```

6. FTLB tag set to all ones, clear FTLB tags with clear GTLB/FTLB tags ASI, verify with logical address set to walking one pattern. The legal error codes for test 0e, case 6 are:

```
0b - FTLB tag RAM match error using instruction FTLB.
0c - FTLB tag RAM match error using data FTLB.
```

11.2.15 Test 0f - Corrupted Block RAM Reset Test

This test verifies that all the bits in the Corrupted Block RAM can be reset. The legal error code for test 11 is:

```
01 - Corrupted bit not zero after reset
```

11.2.16 Test 10 - Cache Tag RAM Address and Data Test

This is an addressing and data test for the cache tag RAMs. The legal error codes for test 0f are:

```
01 - Cache tag match or status error on first forward pass read
02 - Cache tag match or status error when TLB physical address
     field was complemented on forward pass
03 - Cache tag match or status error on second forward pass read
04 - Cache tag match or status error on first reverse pass read
05 - Cache tag match or status error when TLB physical address
     field was complemented on reverse pass
06 - Cache tag match or status error on second reverse pass read
```

11.2.17 Test 11 - Cache Tag Match Test

This is a test of the Cache Tag match detection logic. There are 5 test cases as outlined below:

1. Cache tags set to walking 1 pattern with physical address field of TLB set to zero. The legal error codes for test 10, case 1 are:

```
01 - Cache tag status error
```

2. Cache tags set to walking 0 pattern with physical address field of TLB set to all ones. The legal error codes for test 10, case 2 are:

```
02 - Cache tag status error
```

3. Cache tags set to zero with physical address field of TLB set to walking 1 pattern. The legal error codes for test 10, case 3 are:

```
03 - Cache tag status error
```

4. Cache tags set to all ones with physical address field of TLB set to walking zero pattern. The legal error codes for test 10, case 4 are:

```
04 - Cache tag status error
```

5. Cache tags set to all ones, clear cache tags with ASI 0x94 (cache tag clear ASI), and us physical address field of TLB set to walking zero pattern. The legal error codes for test 10, case 5 are:

```
05 - Cache tag status error
```

11.2.18 Test 12 - Cache RAM Bank Uniqueness Test

This test verifies the the cache RAM bank selection mechanism works. The cache RAM bank is selected on the basis of logical address bit 2. The test also verifies that byte, half-word, word and double-word loads and stores to the cache can be performed. It is verified for each access type, that the data is placed in the correct byte/half-word/word/double-word position in the cache. This test writes patterns of various sizes into the first eight bytes (addresses 0-7) of the cache. The seqeunces of pattern writes and reads and associated error codes are shown below:
Initial state: Address `0` written with `0x55555555`, address 4 written with `0xaaaaaaaa`.

Byte:	0	1	2	3	4	5	6	7
Data:	55	55	55	55	aa	aa	aa	aa

```
Error code 01 - word read at address 0 is not 0x55555555
Error code 02 - word read at address 4 is not 0xaaaaaaaa
```

Second state: Address 0 written with `0xaaaa`, address 4 written with `0x5555`.

Byte:	0	1	2	3	4	5	6	7
Data:	aa	aa	55	55	55	55	aa	aa

```
Error code 03 - word read at address 0 not 0xaaaa5555
Error code 04 - word read at address 4 not 0x5555aaaa
Error code 05 - double byte read at address 2 not 0x5555
Error code 06 - double byte read at address 6 not 0xaaaa
```

Third state: Address 0 and 7 written with 0x55, address 3 and 4 written with 0xaa.

Byte:	0	1	2	3	4	5	6	7
Data:	55	aa	55	aa	aa	55	aa	55

```
Error code 07 - word read at address 0 not 0x55aa55aa
Error code 08 - word read at address 4 not 0xaa55aa55
Error code 09 - double byte read at address 0 not 0x55aa
Error code 0a - double byte read at address 4 not 0xaa55
Error code 0b - byte read at address 0 not 0x55
Error code 0c - byte read at address 4 not 0xaa
Error code 0d - byte read at address 1 not 0xaa
Error code 0e - byte read at address 5 not 0x55
Error code 0f - byte read at address 2 not 0x55
Error code 10 - byte read at address 6 not 0xaa
Error code 11 - byte read at address 3 not 0xaa
Error code 12 - byte read at address 7 not 0x55
```

Fourth state: Address 0 written with `0xaaaaaaaa55555555` (double word write).

Byte:	0	1	2	3	4	5	6	7
Data:	aa	aa	aa	aa	55	55	55	55

```
Error code 13 - double word read at address 0, first word not 0xaaaaaaaa
Error code 14 - double word read at address 0, second word not 0x55555555
```

11.2.19 Test 13 - Cache RAM Addressing and Data Test

This is an addressing and data test for the Cache Data RAMs. The legal error codes for test 13 are:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
```

11.2.20 Test 14 - Flush RAM Addressing and Data Test

This is an addressing and data test for the Flush RAMs. The legal error codes for this test are:

```
01 - flush address data error on first read of forward pass
02 - flush address data error on second read of forward pass
03 - flush address data error on first read of reverse pass
04 - flush address data error on second read of reverse pass
```

11.2.21 Test 15 - Dirty Block RAM Addressing and Data Test

This is an addressing and data test for the Dirty Block Data RAM. The legal error codes for test 15 is:

```
01 - Data error on first forward pass read
02 - Data error on second forward pass read
03 - Data error on first reverse pass read
04 - Data error on second reverse pass read
```

11.2.22 Test 16 - MMU Fault Test

This test verifies that all types of MMU exceptions can be generated. The 15 cases for test 16 follow:

Case 1: TMISS fault (TLBINV true)

This test case maps logical address patterns to physical address zero, but the TLB entries are invalid (sets TLBINV). The logical address pattern is then used as the address in ld instruction. The legal error codes test 16, case 1 are:

```
01 - Data exception was not generated
02 - FCR<TMISS> bit was not set
03 - FVAR register did not contain correct value when read
```

Case 2: TMISS fault (TLBINV false) AND (GM1 = false) AND (GM0 = true)

This test case maps logical address zero to physical address zero to create a valid TLB entry in TLB location zero, then performs a ld instruction at logical address `0x0a000000`. This causes a TLB miss to occur due to a tag mismatch. The FCR and FVAR are read after the exception is verified. The legal error codes test 16, case 2 are:

```
04 - Data exception was not generated
05 - FCR TMISS fault bit was not set
06 - FVAR register did not contain correct value when read
```

Case 3: TMISS fault (TLBINV false) AND (GM1 = true) AND (GM0 = false)

This is the same as case 2 except that `0x50000000` is used for the logical address for the ld instruction (GM1 = true, GM0 = false). The legal error codes test 16, case 3 are:

```
07 - Data exception was not generated
08 - FCR TMISS fault bit was not set
09 - FVAR register did not contain correct value when read
```

Case 4: UPF fault (UP true)

This test case maps logical address `0xaaaaaaaa` to physical address 0 and sets the UP bit (user protect) in the TLB, the FTLB is invalidated, then accesses logical address `0xaaaaaaaa` through user data space to cause a GTLB UP fault. The FCR and FVAR are read after the exception is verified. The legal error codes test 16, case 4 are:

```
0a - Data exception was not generated
0b - FCR UPF bit was not set
0c - The FCR was found with more than just the UPF bit set
0d - FVAR register did not contain correct value when read
```

Case 5: UPF fault (UP true)

This is the same as case 4 except logical address `0x55555555` is used. The legal error codes test 16, case 5 are:

```
0e - Data exception was not generated
0f - FCR UPF bit was not set
10 - The FCR was found with more than just the UPF bit set
11 - FVAR register did not contain correct value when read
```

Case 6: UPF fault (UP true)

This test case maps logical address `0xaaaaaaaa` to physical address 0 and sets the UP bit (user protect) in the TLB, then accesses a half word at logical address `0xaaaaaaaa` through user data space to cause a FTLB UP fault.

```
12 - Data exception was not generated
13 - FCR UPF bit was not set
14 - The FCR was found with more than just the UPF bit set
15 - FVAR register did not contain correct value when read
```

Case 7: UPF fault (UP true)

This is the same as case 6 except an atomic ldstub through logical address 0x55555555 is used. The legal error codes test 16, case 7 are:

```
16 - Data exception was not generated
17 - FCR UPF bit was not set
18 - The FCR was found with more than just the UPF bit set
19 - FVAR register did not contain correct value when read
```

Case 8: UPF fault (FE space)

This test case loads a byte from logical address `0xff555555` through user data space to cause a User Protection fault. The legal error codes test 16, case 8 are:

```
1a - Data exception was not generated
1b - FCR UPF bit was not set
1c - The FCR was found with more than just the UPF bit set
1d - FVAR register did not contain correct value when read
```

Case 9: UPF fault (FE space)

This is the same as case 8 except it loads a half word and logical address `0xfeaaaaaa` is used. The legal error codes test 16, case 9 are:

```
1e - Data exception was not generated
1f - FCR UPF bit was not set
20 - The FCR was found with more than just the UPF bit set
21 - FVAR register did not contain correct value when read
```

Case 10: WPF fault (RO)

This test case maps logical address zero to physical address zero and sets the TLB RO bit. Performs st word instruction to logical address zero to cause a WPF fault to occur.

```
22 - Data exception was not generated
23 - FCR WPF bit was not set
24 - The FCR was found with more than just the WPF bit set
25 - FVAR register did not contain correct value when read
```

Case 11: WPF fault (RO)

This is the same as case 10 except an atomic ldstub instruction is used. The legal error codes test 16, case 11 are:

```
26 - Data exception was not generated
27 - FCR WPF bit was not set
28 - The FCR was found with more than just the WPF bit set
29 - FVAR register did not contain correct value when read
```

Case 12: WPF fault (RO)

This test case maps logical address `0xfdffffff` to physical address zero and sets the TLB RO bit. The legal error codes test 16, case 12 are:

```
2a - Data exception was not generated
2b - FCR WPF bit was not set
2c - The FCR was found with more than just the WPF bit set
2d - FVAR register did not contain correct value when read
```

Case 13: WPF fault (RO)

This is the same as case 12 except logical address `0x00ffffff` is mapped to physical address zero and an atomic ldstuba instruction is used. The legal error codes test 16, case 13 are:

```
2e - Data exception was not generated
2f - FCR WPF bit was not set
30 - The FCR was found with more than just the WPF bit set
31 - FVAR register did not contain correct value when read
```

Case 14: POF fault (PV false)

This test case maps logical address `0x66666666` to physical address zero and clears the TLB page valid bit, then performs ld halfword instruction to logical address `0x66666666` to cause a POF fault. The legal error codes test 16, case 14 are:

```
32 - Data exception was not generated
33 - FCR POF bit was not set
34 - The FCR was found with more than just the POF bit set
35 - FVAR register did not contain correct value when read
```

Case 15: POF fault (PV false)

This is the same as case 14 except logical address `0x99999999` is mapped to physical address zero and a byte load is used. The legal error codes test 16, case 15 are:

```
36 - Data exception was not generated
37 - FCR POF bit was not set
34 - The FCR was found with more than just the POF bit set
39 - FVAR register did not contain correct value when read
```

11.2.23 Test 17 - Double Trap Reset Test

This test verifies that the CPU generates a reset when a double trap condition is detected. The legal error codes for test 17 are:

```
01 - A double trap did not occur
02 - No reset was flagged in the FCR
03 - FCR<DTRAP> was found low after a double trap
04 - FCR<WDOG> was found active after a double trap
```

11.2.24 Test 18 - Watch Dog Timer Reset Test

This test verifies that the CPU generates a reset when a watch dog timer trap is detected. The legal error codes for test 18 are:

```
01 - A watch dog reset occurred after .75 times the time period
02 - A watch dog reset occurred after the timer was cleared
03 - A watch dog reset did not occur after 1.1 times the time period
04 - FCR<WDOG> was found low after a watch dog reset
05 - FCR<DTRAP> was found active after a watch dog reset
```

11.2.25 Test 19 - Timeout Fault Test

This test verifies that the timeout logic on the System Board is functional, that the Kbus Address lines are good and that the timeout detection logic in the bus watcher section of the CPU Board is functional. This is the first test which generates Kbus cycles. The 4 cases for test 18 follow:

Case 1: RIO timeout

Case 1 executes 50 Kbus transactions; all of which are to ID space of non-existant slot 0. Patterns consisting of all zeroes, all ones, walking 1 and walking 0's are used for the low 24 bits of the RIO address. The legal error codes test 18, case 1 are:

```
01 - Data fault exception was not generated
02 - FCR TOF bits was not set
03 - FVAR register did not contain correct RIO address.
```

Case 2: Double-word RIO timeout

Case 2 verifies that double-word RIO accesses generate an exception and that the PDR register specifies that a double-word RIO transaction type was issued. The legal error codes test 18, case 2 are:

```
04 - Data fault exception was not generated for a double-word ld
     instruction
05 - The PDR did not specify that a double-word RIO transaction was
     issued
06 - Data fault exception was not generated for a double-word st
     instruction
07 - The PDR did not specify that a double-word RIO transaction was
     issued
```

Case 3: Fast RIO (FIO) timeout

Case 3 verifies that Fast RIO accesses generate a level 8 interrupt when the cycle times out and that the MMCR<PIO> bit indicates the correct status. The legal error codes test 18, case 3 are:

```
08 - A level 8 interrupt was not generated for an FIO timeout
09 - The pending I/O (PIO) bit was not set when the level 8 interrupt
     occurred
0a - FTOR register did not contain correct physical RIO address after
     a fast RIO timeout
0b - The pending I/O (PIO) bit did not clear after the FTOR was rearmed
0c - The pending I/O (PIO) bit did not set when the fast RIO was started
```

Case 4: Double-word fast RIO (FIO) timeout

Case 4 verifies that Fast RIO accesses generate a level 8 interrupt when a double-word fast RIO cycle is used. Space 1 through 15 is also used to verify that the FTOR latches the correct address. The legal error codes test 18, case 4 are:

```
0d - A level 8 interrupt was not generated for a double-word FIO
     timeout
0e - The PDR did not specify that a double-word RIO transaction was
     issued.
0f - The FTOR did not contain the correct address after a double-word
     RIO transaction was issued
```

11.2.26 Test 1a - Slot Probe and Configuration Test

This test probes each slot of the system by performing ID space reads and determines the board types which occupy each slot. In the following error codes, the X represents the slot number of the target board.

```
0X - Exception other than Data Fault occurred during ID SPACE
     read of slot X
1X - Data exception occurred during initial probe, but FCR TOF
     was not set.
2X - FVAR contained incorrect logical RIO address.
4X - Unrecognizeable board type code read from slot X
5X - Data exception fault occurred during ID space read after valid
     board was previously located in slot X
6X - Data exception fault occurred during RIO read of optional header
     in IDPROM on board in slot X
7X - Data fault exception occurred during RIO read of graphics minor
     board number from IDPROM in slot X
8X - Data fault exception occurred during RIO read of device identifier
     string from IDPROM in slot X
9X - Data fault exception occurred during RIO read of Memory Board size
         from IDPROM in slot X
ax - Zero size parameter read from IDPROM on Memory Board in slot X
bx - Invalid Memory Board size read from IDPROM in slot X
         Not an even 16 Mbyte multiple.
c0 - System Board count is not 1 (0 or more than 1).
c1 - No Memory Boards were located.
c2 - No CPU Boards were located.
d0 - Data exception occurred reading EAROM BOOTMODE variable.
```

11.2.27 Test 1b - IDPROM Checksum Test

This test examines the configuration information obtained from the Slot Probe and configuration test and for each board identified, performs an IDPROM checksum test. The legal error codes for test 1a are:

```
1X - Data exception fault occurred while reading IDPROM size from
     the board in slot X
2X - Zero size field for IDPROM on board in slot X
3X - Data exception fault occurred while performing checksum on
     IDPROM on board in slot X
4X - IDPROM checksum error for board in slot X
5X - Data exception fault occurred while reading the optional header
     field of the IDPROM on board in slot X
```

11.2.28 Test 1c - CPU Status Register Test

This test verifies the ability of the CPU status register to retain data. The legal error code for test 1b are:

```
01 - Data error was found in the CPUSTAT register
```

11.2.29 Test 1d - Master/Slave CPU Determination Test

This test determines which CPU in the system is to become the master CPU when multiple CPUs exist. The legal error codes for test 1c are:

```
10 - Data fault occurred accessing own CPUSTAT register
2x - Data fault occurred accessing CPUSTAT register of CPU in slot X
30 - Data fault occurred accessing EAROM
```

11.2.30 Test 1e - Bus Watcher Tag Reset Test

The legal error codes for test 1d are:

```
01 - Match or Own status error after initial write of tags
     and status.
02 - Match or Own status error after reset of bus watcher tags
     status bits.
```

11.2.31 Test 1f - Bus Watcher Tag RAM Addressing Test

This test verifies the address lines for the Bus Watcher tag and status rams. The legal error codes for test 1e, case 1 are:

```
01 - Bus watcher tag match status error on read of tag location
     other than zero.
02 - Bus watcher tag match status error on read of tag location
     zero.
```

The test is then repeated using locations corresponding to a single address bit off (`0x1ffc0`, `0x1ffa0`..., `0xffe0`), and address `0x1ffe0` is written with zero. The legal error codes for this test 1e, case 2 are:

```
03 - Bus watcher tag match status error on read of tag location
     other than 0x1ffe0.
04 - Bus watcher tag match status error on read of tag location
     0x1ffe0.
```

11.2.32 Test 20 - Bus Watcher Tag Comparitors Test

This is a test of the Bus Watcher Tag match detection logic. There are two test cases. The legal error codes for test 1f, case 1 are:

```
01 - match status error
```

The legal error codes for test 1f, case 2 are:

```
02 - match status error
```

11.2.33 Test 21 - Bus Watcher Tag RAM Address and Data Test

This is an addressing and data test for the bus watcher tag RAMs. The legal error codes for test 20 are:

```
01 - PM0/PM1/OWN status error on first read of forward pass
02 - PM0/PM1/OWN status error on second read of forward pass
03 - PM0/PM1/OWN status error on first read of reverse pass
04 - PM0/PM1/OWN status error on second read of reverse pass
```

11.2.34 Test 22 - Kbus Transaction Type Test

This test verifies that the CPU presents the correct TTYPE to the KBus for the operations used. There are two test cases for test 21: Case 1: Generate cacheable transactions and verify proper types in the PDR. The legal error codes for test 21, case 1 are:

```
01 - The PDR contained the wrong ttype for a read and invalidate
     bus cycle
02 - Access to KBus diagnostic transaction for write and invalidate
     did not generate an expected data exception
03 - The PDR contained the wrong ttype for a write and invalidate
     bus cycle
04 - The PDR contained the wrong ttype for a cacheable read
     bus cycle
```

Case 2: Generate RIO transactions of various sizes to unused slot 0 and verify proper types in the PDR. The legal error codes for test 21, case 2 are:

```
05 - A data exception error was not generated for an 8 bit RIO
     read cycle
06 - The PDR contained the wrong ttype for an 8 bit RIO read cycle
07 - A data exception error was not generated for a 16 bit RIO
     read cycle
08 - The PDR contained the wrong ttype for a 16 bit RIO read cycle
09 - A data exception error was not generated for a 32 bit RIO
     read cycle
0a - The PDR contained the wrong ttype for a 32 bit RIO read cycle
0b - A data exception error was not generated for an 8 bit RIO
     read-modify-write cycle
0c - The PDR contained the wrong ttype for an 8 bit RIO
     read-modify-write cycle
0d - A data exception error was not generated for an 8 bit RIO
     write cycle
0e - The PDR contained the wrong ttype for an 8 bit RIO write cycle
0f - A data exception error was not generated for a 16 bit RIO
     write cycle
10 - The PDR contained the wrong ttype for a 16 bit RIO write cycle
11 - A data exception error was not generated for a 32 bit RIO
     write cycle
12 - The PDR contained the wrong ttype for a 32 bit RIO write cycle
```

11.2.35 Test 23 - Memory Board Base Address and Enable Register Test

This is a test for the Base Address Register and Enable Register on each installed Memory Board. The legal error codes for test 22 are:

```
1X - Data exception fault occurred writing base address register of
     Memory Board in slot X
2X - Data exception fault occurred reading base address register of
     Memory Board in slot X
3X - Data miscompare error for base address register on Memory Board
     in slot X
4X - Data exception fault occurred writing enable register of Memory
     Board in slot X
5X - Data exception fault occurred reading enable register of Memory
     Board in slot X
6X - Data miscompare error for enable register on Memory Board
     in slot X.
```

11.2.36 Test 24 - Memory Board Uniqueness Test

This test verifies that all installed Memory Boards can be accessed independently of all others. This is the first test which attempts to write and read memory and thereby test the bus watchers ability to perform Kbus transactions other than RIO types. The legal error codes for test 23 are:

```
0X - Indicates an exception occurred on the initial "stb" instruction.
     The error code is the slot number of the target Memory Board.
1X - Indicates that a read of the data cached on the initial "stb"
     is not readable from the cache. The low nibble of the error
     code is the slot number.
2X - Indicates an exception occurred on the flush operation.  This
     is the instruction which causes the block flush back to memory.
     The low nibble of the error code is the slot number.
3X - Indicates an exception occurred on the re-read of target byte.
     The low nibble of the error code is the slot number.
4X - Indicates a data error on the re-read on target byte. This is
     the instruction which causes the target block to be re-cached
     and supplied to the CPU. The low nibble of the error code is
     the slot number.
```

11.2.37 Test 25 - Memory Board Address Uniqueness Test

This test verifies the uniqueness of the upper bits of the memory address. The legal error codes for test 24, case 1 are:

```
0X - Indicates that the Memory Board responded when it should not
     have.  The base address register of the target board is set to
     0x00 and it responded to some other board address.  The low
     nibble of the error code is the slot number of the target
     Memory Board.
1X - Indicates that the wrong exception type occurred when the
     Memory Board was read.  The expected exception vector is
     9. The low nibble of the error code is the slot number of
     the target Memory Board.
2X - FCR <TOF> bit was not set when exception occurred.  The low
     nibble of the error code is the slot number of the target memory
     board.
3X - FVAR register did not contain the correct address when the
     exception occurred.  The low nibble of the error code is the
     slot number of the target memory board.
```

The test is repeated using 0xff in the Base Address register and a walking zero pattern on the upper significant bits of the address. The legal error codes for test 24, case 2 are:

5X - Indicates that the Memory Board responded when it should not have. The base address register of the target board is set to 0x00 and it responded to some other board address. The low nibble of the error code is the slot number of the target Memory Board.

6X - Indicates that the wrong exception type occurred when the Memory Board was read. The expected exception vector is 9. The low nibble of the error code is the slot number of the target Memory Board.

7X - FCR <TOF> bit was not set when exception occurred. The low nibble of the error code is the slot number of the target memory board.

8X - FVAR register did not contain the correct address when the exception occurred. The low nibble of the error code is the slot number of the target memory board.

11.2.38 Test 26 - Memory Board Addressing Test

This test verifies that each installed Memory Board can respond to all unique Memory Board addresses (0x00 through 0x0f). The legal error codes for test 25 are:

0X - Indicates that an exception occurred when the board address was written. The low nibble of the error code is the slot number of the target Memory Board.

1X - Indicates that an exception occurred when flushing the target block back to memory. The low nibble of the error code is the slot number of the target Memory Board.

2X - Indicates that an exception occurred when the board address was read. The low nibble of the error code is the slot number of the target Memory Board.

3X - Indicates the wrong data was returned from the target Memory Board. The low nibble of the error code is the slot number of the target Memory Board.

11.2.39 Test 27 - Memory Board Block Addressability Test

This test verifies the uniqueness of the address lines on each installed Memory Board. The 16 Mbyte Memory Board address is broken down as follows:

```
0x00000000 - 0x007fffe0:  Low 8 Mbyte Bank
0x00800000 - 0x00ffffe0:  High 8 Mbyte Bank
```

The legal error codes for test 26 are:

```
0X - Data fault exception occurred on ld instruction using walking
     0/1 address from Memory Board in slot X.
1X - Data miscompare occurred on ld instruction using walking 0/1
     address from Memory Board in slot X.
2X - Data fault exception occurred on ld instruction using all
     zeroes/ones address from Memory Board in slot X.
3X - Data miscompare occurred on ld instruction using all zeroes/ones
     address from Memory Board in slot X.
4X - Data fault exception occurred on st instruction using walking
     0/1 address on Memory Board in slot X.
5X - Data fault exception occurred on ld instruction after data from
     Memory Board X was already cached.
6X - Data miscompare occurred on ld instruction after data from
     Memory Board X was already cached.
7X - Data fault exception occurred on st instruction using all
     zeroes/ones address on Memory Board in slot X.
8X - Data fault exception occurred on ld instruction after data from
     Memory Board X was already cached.
9X - Data miscompare occurred on ld instruction after data from
     Memory Board X was already cached.
```

11.2.40 Test 28 - Memory Board RAM Addressing and Data Test

This is an addressing and data test for the first 1 Mbyte of memory. Only the first 1 Mbyte of memory is tested to keep execution time during power-up selftest to a minimum. Legal error codes for test 27 are:

```
1X - Data fault exception occurred during write of memory
     with initial data pattern.
2X - Data fault exception occurred on first read of forward pass
3X - Data miscompare occurred on first read of forward pass
4X - Data fault exception occurred during flush of target
     memory block back to memory during forward pass.
5X - Data fault exception occurred on second read of forward pass
6X - Data miscompare occurred on second read of forward pass
7X - Data fault exception occurred on first read of reverse pass
8X - Data miscompare occurred on first read of reverse pass
9X - Data fault exception occurred during flush of target
     memory block back to memory during reverse pass.
aX - Data fault exception occurred on second read of reverse pass
bX - Data miscompare occurred on second read of reverse pass
```

11.2.41 Test 29 - Cache Fill-Flush Test

This test fills the entire 128 Kbytes of cache RAM with the first 128 Kbytes of the bootrom code. Next, the second 128 Kbytes of bootrom code is then written to the cache. This should displace the contents of the cache out to physical memory. The legal error codes for test 28 are:

```
01 - Data fault exception occurred on st instruction to
     memory while loading cache with first 128 Kbytes of
     bootrom code.
02 - Data fault exception occurred on st instruction to
     memory while loading cache with second 128 Kbytes of
     bootrom code.
03 - Data fault exception occurred on ld instruction from
     memory while verifying first 128 Kbytes of data.
04 - Data miscompare occurred while verifying first 128 Kbytes
     of data.
05 - Data fault exception occurred on ld instruction from
     memory while verifying second 128 Kbytes data.
06 - Data miscompare occurred while verifying second 128 Kbytes
     of data.
```

11.2.42 Test 2a - Virtual Fault Cache Corruption Test

This test verifies that exceptions which occur due to cache writes do not corrupt the cache data. There are eight test cases. The legal error codes for the 13 cases in test 29 are:

Case 1: Single precision misaligned store exception to FF space

```
01 - Address Alignment fault did not occur on st to misaligned
     word address.
02 - First word of cache line corrupted on st to misaligned word
     address.
03 - Second word of cache line corrupted on st to misaligned word
     address.
```

Case 2: Double precision misaligned store exception to FF space

```
04 - Address Alignment fault did not occur on std to misaligned
     double-word address.
05 - First word of cache line corrupted on std to misaligned
     double-word address.
06 - Second word of cache line corrupted on std to misaligned
     double-word address.
```

Case 3: Single precision misaligned store operation with MMU enabled

```
07 - Address Alignment fault did not occur on st to misaligned
     word address.
08 - First word of cache line corrupted on st to misaligned word
     address.
09 - Second word of cache line corrupted on st to misaligned word
     address.
```

Case 4: Double precision misaligned store operation with MMU enabled

```
0a - Address Alignment fault did not occur on std to misaligned
     double-word address.
0b - First word of cache line corrupted on std to misaligned
     double-word address.
0c - Second word of cache line corrupted on std to misaligned
     double-word address.
```

Case 5: Single precision read only store exception

```
0d - Data fault exception did not occur on st to page marked read
     only in TLB.
0e - First word of cache line corrupted on st to page marked read
     only in TLB.
0f - Second word of cache line corrupted on st to page marked read
     only in TLB.
```

Case 6: Double precision read only store exception

```
10 - Data fault exception did not occur on std to page marked read
     only in TLB.
11 - First word of cache line corrupted on std to page marked read
     only in TLB.
12 - Second word of cache line corrupted on std to page marked read
     only in TLB.
```

Case 7: Atomic read only store exception

```
13 - Data fault exception did not occur on ldstub to page marked read
     only in TLB.
14 - First word of cache line corrupted on ldstub to page marked read
     only in TLB.
15 - Second word of cache line corrupted on ldstub to page marked read
     only in TLB.
```

Case 8: Single precision TLB miss store exception

```
16 - Data fault exception did not occur on st to page marked as
     invalid in TLB.
17 - First word of cache line corrupted on st to page marked as
     invalid in TLB.
18 - Second word of cache line corrupted on st to page marked as
     invalid in TLB.
```

Case 9: Double precision TLB miss store exception

```
19 - Data fault exception did not occur on std to page marked as
     invalid in TLB.
1a - First word of cache line corrupted on std to page marked as
     invalid in TLB.
1b - Second word of cache line corrupted on std to page marked as
     invalid in TLB.
```

Case 10: Atomic TLB miss store exception

```
1c - Data fault exception did not occur on ldstub to page marked as
     invalid in TLB.
1d - First word of cache line corrupted on ldstub to page marked as
     invalid in TLB.
1e - Second word of cache line corrupted on ldstub to page marked as
     invalid in TLB.
```

Case 11: Single precision User protect store exception

```
1f - Data fault exception did not occur on st to page marked as
     user protected in TLB.
20 - First word of cache line corrupted on st to page marked as
     user protected in TLB.
21 - Second word of cache line corrupted on st to page marked as
     user protected in TLB.
```

Case 12: Double precision User protect store exception

```
22 - Data fault exception did not occur on std to page marked as
     user protected in TLB.
23 - First word of cache line corrupted on std to page marked as
     user protected in TLB.
24 - Second word of cache line corrupted on std to page marked as
     user protected in TLB.
```

Case 13: Atomic User protect store exception

```
25 - Data fault exception did not occur on ldstub to page marked as
     user protected in TLB.
26 - First word of cache line corrupted on ldstub to page marked as
     user protected in TLB.
27 - Second word of cache line corrupted on ldstub to page marked as
     user protected in TLB.
```

11.2.43 Test 2b - Corrupted Block RAM Addressing and Data Test

This is an addressing and data test for the Corrupted Block RAM. The legal error codes for test 2a are:

```
01 - Corrupt Bit not one on first read of forward pass
02 - Corrupt bit not zero on second read of forward pass
03 - Corrupt Bit not zero on first read of reverse pass
04 - Corrupt bit not one on second read of reverse pass
```

11.2.44 Test 2c - Corrupted Block Flush Inhibit Test

This test verifies that cache transactions which reference a corrupted block result in a Kbus timeout. The legal error codes for test 2b are:

```
01 - No memory timeout fault was generated when Memory Boards
     disabled.
02 - No memory timeout fault was generated on reference to corrupted
     block.
03 - FCR TOFM bit not set
04 - FVAR does not contain the correct logical address
```

11.2.45 Test 2d - Virtual Cache Block Replacement Test

This test exercises the cache and bus watcher cache block flush logic by performing writes and reads to common physical addresses through all different logical addresses including FF space. Legal error codes for test 2c are:

```
01 - Data fault exception occurred during creation of the valid
     owned and dirty cache blocks.
02 - Data fault exception occurred during read of target physical
     cache block.
03 - Physical data read through logical address does not match
     expected physical address data.
04 - Data fault exception occurred during read of target physical
     cache blocks using FF space addresses.
05 - Physical data read through FF space address does not match
     expected physical address data.
```

11.2.46 Test 2e - Atomic load/store instruction test

This test exercises the control logic for the LDSTUB instruction in conjunction with cache and TLB miss conditions. There are 8 cases for test 2d, as follows:

1. Execute LDSTUB instruction to FF space and generate a cache hit and an FTLB hit. The legal error codes for test 2d, case 1 are:

   ```
   01 - An exception occurred on the LDSTUB instruction.
   02 - The data read from the cache was incorrect.
   03 - The data written to the cache was not 0xff.
   ```

2. Execute LDSTUB instruction to user space and generate a cache hit and an FTLB hit. The legal error codes for test 2d, case 2 are:

   ```
   04 - An exception occurred on the LDSTUB instruction.
   05 - The data read from the cache was incorrect.
   06 - The data written to the cache was not 0xff.
   ```

3. Execute LDSTUB instruction to FF space and generate a cache hit and an FTLB miss. The legal error codes for test 2d, case 3 are:

   ```
   07 - An exception occurred on the LDSTUB instruction.
   08 - The data read from the cache was incorrect.
   09 - The data written to the cache was not 0xff.
   ```

4. Execute LDSTUB instruction to user space and generate a cache hit and an FTLB miss. The legal error codes for test 2d, case 4 are:

```
0a - An exception occurred on the LDSTUB instruction.
0b - The data read from the cache was incorrect.
0c - The data written to the cache was not 0xff.
```

5. Execute LDSTUB instruction to FF space and generate a cache miss and an FTLB hit. The legal error codes for test 2d, case 5 are:

```
0d - An exception occurred on the LDSTUB instruction.
0e - The data read from the cache was incorrect.
0f - The data written to the cache was not 0xff.
```

6. Execute LDSTUB instruction to user space and generate a cache miss and an FTLB hit. The legal error codes for test 2d, case 6 are:

```
10 - An exception occurred on the LDSTUB instruction.
11 - The data read from the cache was incorrect.
12 - The data written to the cache was not 0xff.
```

7. Execute LDSTUB instruction to FF space and generate a cache miss and an FTLB miss. The legal error codes for test 2d, case 7 are:

```
13 - An exception occurred on the LDSTUB instruction.
14 - The data read from the cache was incorrect.
15 - The data written to the cache was not 0xff.
```

8. Execute LDSTUB instruction to user space and generate a cache miss and an FTLB miss. The legal error codes for test 2d, case 8 are:

```
16 - An exception occurred on the LDSTUB instruction.
17 - The data read from the cache was incorrect.
18 - The data written to the cache was not 0xff.
```

11.2.47 Test 2f - Paged Out Test

This test verifies that simultaneous instruction and data TLB faults are handled correctly. In addition, this is the first test which actually executes instructions out of the cache by jumping from bootrom space (FE space) to cacheable space (FF space). There are 4 cases for test 2e, as follows:

1. Perform JMP instruction to an instruction page in FF space which has the VALID bit cleared (invalid page). The legal error codes for test 2e, case 1 are:

```
01 - An instruction fault did not occur.
02 - The POF bit in the FCR register did not get set.
03 - The FVAR did not contain the correct page address.
04 - The code in the invalid page did not execute correctly.
```

2. Perform JMP instruction to an instruction page in FF space which is invalid (TLB entry has the VALID bit cleared) and execute a LD instruction from an invalid data page in the delay slot of the JMP instruction. The legal error codes for test 2e, case 2 are:

```
05 - Instruction and data faults occurred in the wrong order or did not
     occur.
06 - The POF bit in the FCR register did not get set on the data fault.
07 - The FVAR did not contain the correct data page address.
08 - The POF bit in the FCR register did not get set on the text fault.
09 - The FVAR did not contain the correct text page address.
0a - The LD instruction did not complete correctly (wrong data returned).
0b - The code in the invalid page did not execute correctly.
```

3. Perform JMP instruction to an instruction page in FF space which is invalid (TLB entry has the VALID bit cleared) and execute a ST instruction to an invalid data page in the delay slot of the JMP instruction. The legal error codes for test 2e, case 3 are:

```
0c - Instruction and data faults occurred in the wrong order or did not
     occur.
0d - The POF bit in the FCR register did not get set on the data fault.
0e - The FVAR did not contain the correct data page address.
0f - The POF bit in the FCR register did not get set on the text fault.
10 - The FVAR did not contain the correct text page address.
11 - The ST instruction completed. (store should have been prevented).
12 - The code in the invalid page did not execute correctly.
```

4. Perform LDST instruction to a page in FF space which is read only (TLB entry has the RO bit set). The legal error codes for test 2e, case 4 are:

```
13 - A data fault did not occur.
14 - The WPF bit in the FCR register did not get set.
15 - The FVAR register did not contain the correct data page address.
16 - The store part of the ldst instruction completed.
```

11.2.48 Test 30 - ECC Write/Read Test

This test verifies the ECC data path to and from each installed Memory Board. The legal error codes for test 2f are:

```
01 - ECCS or data fault exception occurred on store of data
     pattern with ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of test
     pattern back to memory.
03 - Single Bit ECC exception generated on re-read of test
     pattern from memory with ECC checking enabled.
04 - Multi-bit ECC exception generated on re-read of test
     pattern from memory with ECC checking enabled.
05 - Exception other than ECCS or ECCM generated on re-read
     of test pattern from memory with ECC checking enabled.
06 - Data error in first word of cache line 0
07 - Data error in second word of cache line 0
08 - Data error in first word of cache line 1
09 - Data error in second word of cache line 1
0a - Data error in first word of cache line 2
0b - Data error in second word of cache line 2
0c - Data error in first word of cache line 3
0d - Data error in second word of cache line 3
```

11.2.49 Test 31 - ECC Single Bit Correction to 1 Test

This test verifies that the ECC data correction logic can correct a bit from a zero to a one for all 64 data bit positions. The test is performed independly for each all four cache lines. The legal error codes for test 30 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.2.50 Test 32 - ECC Single Bit Correction to 0 Test

This test verifies that the ECC data correction logic can correct a bit from a one to a zero for all 64 data bit positions. The test is performed independly for each all 4 cache lines. The legal error codes for test 31 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.2.51 Test 33 - ECC Single Bit Checkbyte Error Test

This test verifies that single bit errors in the checkbyte are detectable and causes no cache line data corruption. The legal error codes for test 32 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than ECCS exception was generated on re-read
     of target cache block.
05 - FPAR register contains incorrect address.
06 - FES register contains incorrect syndrome value.
07 - Data error in cache line 0
08 - Data error in cache line 1
09 - Data error in cache line 2
0a - Data error in cache line 3
```

11.2.52 Test 34 - ECC Multibit Error Detection Test

This test verifies that all syndrome values which map to a two bit or more than two bit error results in the generation of a multibit ECC exception. The legal error codes for test 33 are:

```
01 - ECCS or data fault exception occurred on initial load
     of data patterns when ECC checking disabled.
02 - ECCS or data fault exception occurred on flush of target
     cache block to memory when ECC checking disabled.
03 - No exception was generated on re-read of target cache block.
04 - Exception other than Data Fault exception was generated on
     re-read of target cache block.
05 - FCR ECCM bit not set.
06 - FVAR register contains incorrect address.
07 - FCR not cleared after read of FVAR.
```

11.2.53 Test 35 - ECC RAM Addressing and Data Test

This is an addressing and data test for the first megabyte of ECC memory. Only the first megabyte of memory are tested to keep execution time during power-up selftest to a minimum.

Legal error codes for test 34 are:

```
1X - Data fault exception occurred during write of memory
     with initial data pattern.
2X - ECCS or data fault exception occurred on first read of
     forward pass
3X - Data miscompare occurred in upper 32 bits of cache line
     during forward pass
4X - Data miscompare occurred in lower 32 bits of cache line
     during forward pass
5X - Data fault exception occurred during flush of target
     memory block back to memory during forward pass.
6X - ECCS or data fault exception occurred on second read of
     forward pass
7X - Data miscompare occurred in upper 32 bits of cache line
     during forward pass
8X - Data miscompare occurred in lower 32 bits of cache line
     during forward pass
9X - ECCS or data fault exception occurred on first read of
     reverse pass
aX - Data miscompare occurred in upper 32 bits of cache line
     during reverse pass
bX - Data miscompare occurred in lower 32 bits of cache line
     during reverse pass
cX - Data fault exception occurred during flush of target
     memory block back to memory during reverse pass.
dX - ECCS or data fault exception occurred on second read of
     reverse pass
eX - Data miscompare occurred in upper 32 bits of cache line
     during reverse pass
fX - Data miscompare occurred in lower 32 bits of cache line
     during reverse pass
```

11.2.54 Test 36 - FPU Register Load/Store Test

This test verifies the primary interaction between the floating point unit and the memory system by performing a write/read test on one of the floating point register pairs. There are two test cases, one for single precision values and one for double-precision values. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 35 are:

```
01 - After attempting to clear the QNE bit on the FPU state
     register, the queue (FQ) is still not empty.
02 - Write read error for single precision load/store.
03 - Write read error for double precision load/store (even register).
04 - Write read error for double precision load/store (odd register).
```

11.2.55 Test 37 - FPU State Register Test

The FPU state register (FSR) contains FPU mode and status information. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 36 are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - FSR write read error.
```

11.2.56 Test 38 - FPU Add/Multiply/Divide Test

This test verifies the path between the FPC and the floating point arithmetic units on the FPC/FALU and the FPC/FMULT interfaces. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 37 are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - Incorrect single precision addition result.
03 - Incorrect single precision multiplication result.
04 - Incorrect double precision addition result (even register).
05 - Incorrect double precision addition result (odd register).
06 - Incorrect double precision multiplication result (even register).
07 - Incorrect double precision multiplication result (odd register).
08 - Incorrect single precision division result.
09 - Incorrect double precision division result (even register).
0a - Incorrect double precision division result (odd register).
0b - FPU did not handled operand dependency correctly.
```

11.2.57 Test 39 - FPU Queue Test

The FPU queue (FQ) keeps tracks of floating point operations that are pending by the FPU when a floating point fp_exception trap occurs. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for this test are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
02 - FSR write read error while setting TEM (NV) bit.
03 - FPU fp_exception trap did not occur when expected.
04 - FSR QNE bit is clear when it should be set.
05 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
```

11.2.58 Test 3a - FPU Exceptions Test

There are two floating point trap types that are generated by the FPU hardware. These are: fp_disabled and fp_exception. The FPU generates four types of exception traps:

1. FPC sequence error exception
2. Unimplemented floating point instruction exception. (Not checked by this test. All instructions are implemented.)
3. Unfinished floating point instruction exception
4. IEEE exception

IEEE exceptions are classified as follows:

1. Invalid
2. Overflow
3. Underflow
4. Division by zero
5. Inexact

This test verifies that the FPU generates these traps and exceptions properly by performing test cases for each type. This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for the 13 cases in test 39 are:

Test Case 1: fp_disabled trap

```
01 - FPU fp_disabled trap did not occur when expected.
```

Test Case 2: fp_exception IEEE-Invalid while enabled

```
02 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
03 - FPU fp_exception trap did not occur when expected (IEEE-Invalid).
04 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Invalid.
```

Test Case 3: fp_exception IEEE-Invalid while disabled

```
05 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
06 - FPU fp_exception trap occured while traps were disabled (IEEE Invalid).
07 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Invalid.
```

Test Case 4: fp_exception IEEE-Overflow while enabled

```
08 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
09 - FPU fp_exception trap did not occur when expected (IEEE-Overflow).
0a - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Overflow.
```

Test Case 5: fp_exception IEEE-Overflow while disabled

```
0b - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
0c - FPU fp_exception trap occured while traps were disabled (IEEE Overflow).
0d - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Overflow.
```

Test Case 6: fp_exception IEEE-Underflow while enabled

```
0e - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
0f - FPU fp_exception trap did not occur when expected (IEEE-Underflow).
10 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Underflow.
```

Test Case 7: fp_exception IEEE-Underflow while disabled

```
11 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
12 - FPU fp_exception trap occured while traps were disabled (IEEE Underflow).
13 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Underflow.
```

Test Case 8: fp_exception IEEE-Inexact while enabled

```
14 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
15 - FPU fp_exception trap did not occur when expected (IEEE-Inexact).
16 - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Inexact.
```

Test Case 9: fp_exception IEEE-Inexact while disabled

```
17 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
18 - FPU fp_exception trap occured while traps were disabled (IEEE Inexact).
19 - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Inexact.
```

Test Case 10: fp_exception IEEE-Divide-By-Zero while enabled

```
1a - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
1b - FPU fp_exception trap did not occur when expected (IEEE-Divide-by-Zero).
1c - FPU fp_exception trap ocurred, but FSR FTT and CEXC bits are not
     set for IEEE-Divide-by-Zero
```

Test Case 11: fp_exception IEEE-Divide-By-Zero while disabled

```
1d - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
1e - FPU fp_exception trap occured while traps were disabled.
1f - FPU fp_exception trap did not occured, but FSR CEXC and AEXC bits
     are not set for Divide-By-Zero.
```

Test Case 12: fp_exception Sequence-Error

```
20 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
21 - FPU fp_exception trap did not occur when expected (IEEE-Divide-by-Zero).
22 - FPU fp_exception trap did not occur when expected (SEQUENCE).
23 - FPU fp_exception trap ocurred, but FSR FTT bits are not set
     for SEQUENCE.
```

Test Case 13: fp_exception Unfinished-Floating-Point-Instruction

```
24 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
25 - FPU fp_exception trap did not occur when expected (UNFINISHED_FPOP).
26 - FPU fp_exception trap ocurred, but FSR FTT bits are not set
     for UNFINISHED_FPOP
```

11.2.59 Test 3b - FPU Condition Codes Test

Floating point compares (FCMPS) and floating point condition (FBfcc) instructions interlock on the floating point condition codes. The condition codes supported by the FPU are:

1. Equal Relation
2. Greater-Than Relation
3. Less-Than Relation
4. Unordered Relation

This test as well as all other floating point unit tests are only executed if the floating point unit is available in the CPU Board. Legal error codes for test 3a are:

```
01 - After attempting to clear the QNE bit on the FSR, the queue
     (FQ) is still not empty.
```

Test Case 1: Equal Relation when A == B

```
02 - CC should reflect an equal relation, causing FBE
     instruction to fail.
03 - FSR FCC bits not reflecting an equal relation when expected.
```

Test Case 2: Equal Relation when A != B

```
04 - CC should not reflect an equal relation, causing FBE
     instruction to fail.
05 - FSR FCC bits reflecting an equal relation when not expected.
```

Test Case 3: Greater-Than Relation when A > B

```
06 - CC should reflect a greater-than relation, causing FBG
     instruction to fail.
07 - FSR FCC bits not reflecting a greater-than relation when expected.
```

Test Case 4: Greater-Than Relation when A < B

```
08 - CC should not reflect a greater_than relation, causing FBG
     instruction to fail.
09 - FSR FCC bits reflecting a greater-than relation when not expected.
```

Test Case 5: Less-Than Relation when A < B

```
0a - CC should reflect a less-than relation, causing FBL
     instruction to fail.
0b - FSR FCC bits not reflecting a less-than relation when expected.
```

Test Case 6: Less-Than Relation when A > B

```
0c - CC should not reflect a less_than relation, causing FBL
     instruction to fail.
0d - FSR FCC bits reflecting a less-than relation when not expected.
```

Test Case 7: Unordered Relation when A unordered, B ordered

```
0e - CC should reflect an unordered relation, causing FBU
     instruction to fail.
0f - FSR FCC bits not reflecting an unordered relation when expected.
```

Test Case 8: Unordered Relation when A & B ordered

```
10 - CC should not reflect an unordered relation, causing FBU
     instruction to fail.
11 - FSR FCC bits reflecting an unordered relation when not expected.
```

11.2.60 Test 3c - System Board Interrupt Generation Test

This is a test of the interrupt register on the System Board and the ability of the System Board to generate all 16 vectors when enabled after reset. Part 1 first reads RIO address `17030000` to disable transmission of interrupts, then writes RIO address `17030000` with incrementing test patterns from `0` to `0xff`. Each pattern is read back and verified. The legal error codes for test 3b, part 1 are:

```
01 - Data fault exception occurred on initial read to disable
     System Board interrupt register.
02 - Data fault exception occurred on write of pattern to
     System Board interrupt register.
03 - Data fault exception occurred on read of System Board
     interrupt register.
04 - Data pattern written does not match data pattern read from
     System Board interrupt register.
```

Part 2 initializes the System Board interrupt register with the directed bit set and the destination ID set the BID of the CPU Board. System Board interrupts are then enabled by reading address `17031000`. The test verifies that all 16 interrupt vectors are received correctly. Note that this test will fail if a system reset is not performed inbetween passes. The legal error codes for test 3b, part 2 are:

```
05 - Timeout waiting to receive first interrupt (vector 0x8f) from
     System Board.
06 - Exeption other than Serial Interrupt Controller occurred.
07 - Higher priority interrupt vector was received 256 times without
     receiving expected vector.
8X - Lower priority interrupt vector was received. Error code is the
     vector which was expected.
```

Section 12: rdg Diagnostics

12.1 Introduction

rdg is a ROM-resident diagnostics program. It is used to determine why a Solbourne system will not boot, if problems are encountered while booting the system.

12.2 rdg Tests

The **rdg** (1) debugger tests include:

1. RTC-58321 Real Time Clock Test
2. Memory Data RAM Test (affected by **prompt**)
3. Memory ECC RAM Test (affected by **prompt**)
4. VMEbus Address Map RAM Test
5. VMEbus Data Path Test
6. VMEbus Address Path Test
7. RF3500 SCSI Data Path (Write Buffer) Test
8. I/O ASIC Register Access Test
9. I/O ASIC FIFO/ECC Test
10. 7990 LANCE Initialization Test
11. 7990 LANCE Internal Loopback Test
12. 7990 LANCE External Loopback Test (must be prompted)
13. 33C93 SBIC (SCSI) Enable Test
14. 33C93 SBIC (SCSI) Data Path (Write Buffer) Test
15. Ethernet tftp Read Test
16. Disk Write/Read Test
17. Tape Write/Read Test (must be prompted)

12.3 rdg Commands

A summary of the command usage is displayed on-line when **rdg** is running by typing:

```
RDG> ?
```

The following command set is a subset of the **dg** command set. See Section 13.2.1 for a brief description of these commands. The following is a listing of the **rdg** commands available:

between (1)	**config** (1)	**continue** (1)
errlim (1)	**errors** (1)	**fbconfig** (1)
help (1)	**limit** (1)	**loop** (1)
menu (1)	**names** (1)	**next** (1)
passes (1)	**passlim** (1)	**prompt** (1)
quiet (1)	**quit** (1)	**rdg** (1)
restart (1)	**run** (1)	**status** (1)
tests (1)	**time** (1)	**vmeconf** (1)

12.4 Field Notes

Section 13: dg Diagnostics

13.1 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.

13.2 Invoking dg

Before invoking **dg** it is recommended to reset the system at the ROM prompt by typing:

```
ROM> reset cold
```

At the ROM> prompt, type:

```
ROM> b -f sd.si(,,6)kvm/stand/dg
```

Commands and parameters are case insensitive.

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.

13.2.1 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

13.3 Overview of dg Tests

The **dg** menu of tests is similar to the hierarchal tree-like structure of the UNIX file system. Figure 13-1 illustrates the menu structure of **dg**.

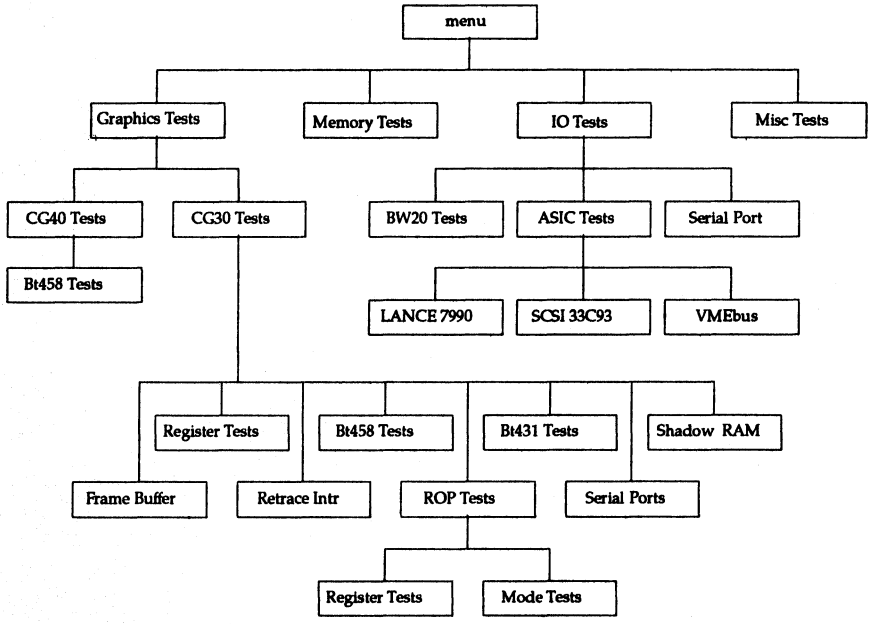


Figure 13-1. The dg menu Structure

Moving about the **dg** menu structure has been made easier by the installation of the UNIX-type commands **cd** (1) and **ls** (1).

Figures 13-2 and 13-3 show where the **dg** tests reside in the menu structure. The test names in these illustrations have been shortened. To see the full path name, refer to Section 13.4.

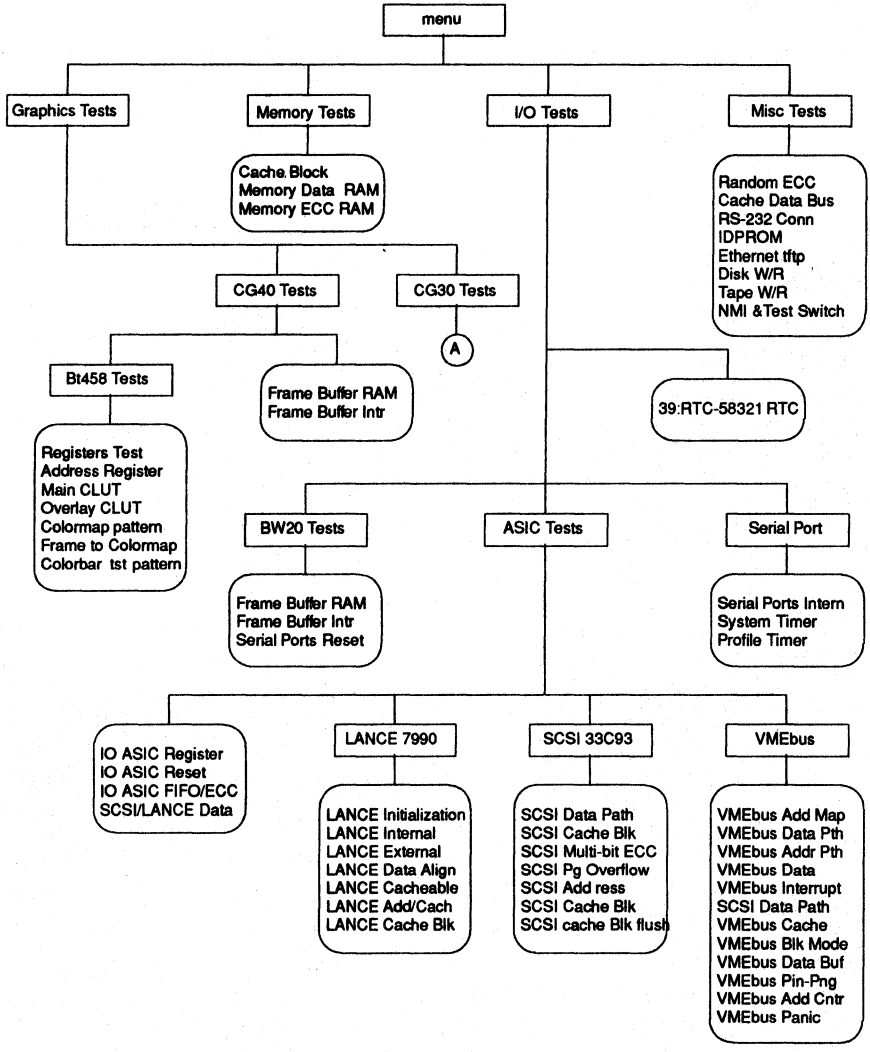


Figure 13-2. Tests and Test Submenus

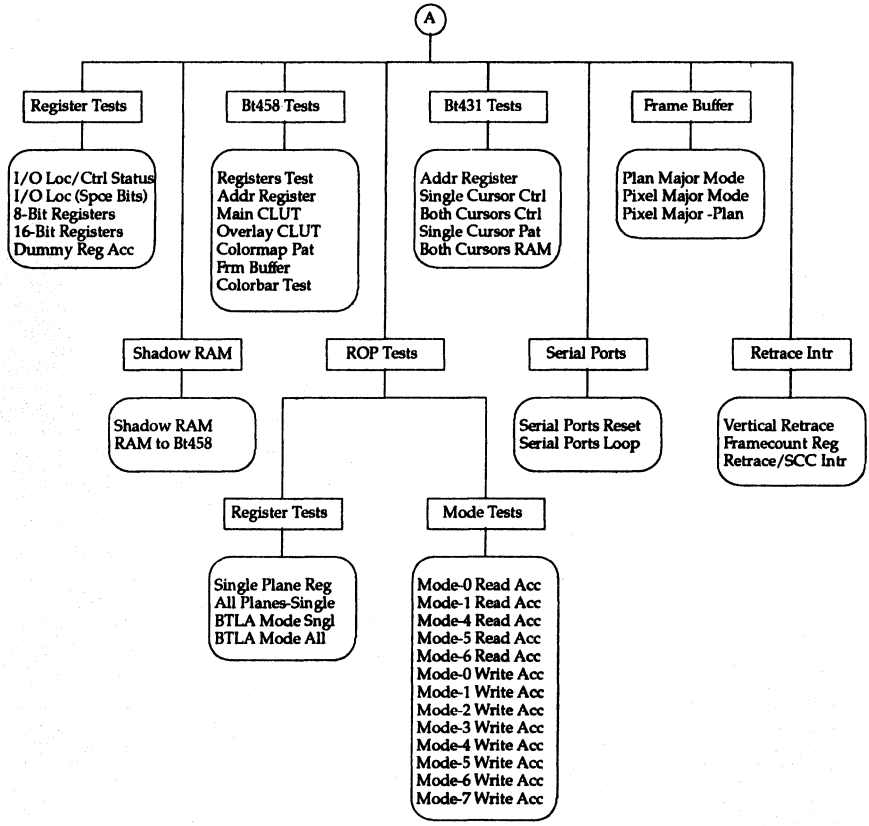


Figure 12-2. Tests and Test Submenus (Continued)

13.4 Example of Using dg Commands

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> ls
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> ls
Menu of : /Graphics Tests
  ==>   CG40 Tests
  ==>   CG30 Tests
DG> tests *
DG> tests
DG> tests
selected tests:  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
DG>
```

13.5 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**)

13.6 Field Notes

Section 14: mdg Diagnostics

14.1 Introduction

mdg is a standalone test controller for use on Solbourne multiprocessor systems. Information that may be useful while using the **mdg** program is available in the following documentation:

14.2 Invoking mdg

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

The user must first bring the Solbourne system to the **ROM>** prompt. If the system has been halted and brought to the **ROM>** prompt, go to step one of the following procedure.

If UNIX is running, the system must be shutdown using the **halt**(1) command. Once you have halted UNIX, you must enter:

```
reset cold
```

before you use the following procedure for invoking **mdg**.

☆ ☆ ☆ **NOTE** ☆ ☆ ☆

Before invoking **mdg**, it is recommended to reset the system by entering "reset cold" at the ROM> prompt.

1. At the ROM> prompt, type:

```
ROM>b sd.si()/kvm/stand/mdg
```

2. When **mdg** starts up, the following message is displayed:

```
MDG - Multiprocessor Diagnostic Test Controller
      Version 1.1 September 25, 1989
      Copyright (c) 1989 Solbourne Computer, Inc.
```

3. As **mdg** starts up, the following steps are undertaken by the MASTER processor:
   - Obtain the number of processors in the system and the results of power-up diagnostics from the diagnostic RAM.
   - Calculate the system-wide (shared memory) and CPU-specific (private memory) test limits.

- Configure the memory configuration table with the number of memory boards in the system as well as their addressing range.
- Configure the frame buffer configuration table with the values found during power-up.
- Initialize the VMEbus configuration table as empty.
- Awake each SLAVE processor in the system that passed the power-up diagnostics. Each SLAVE processor will register with the MASTER processor in order for the MASTER to include it as part of the selected list of available system processors that **mdg** maintains.
- By default, all available tests are selected and all the available processors are included for testing.

4. Upon completion of the previous setup, **mdg** will display the following message:

```
CPU Configuration:

2 CPU boards:

Slot#     Power-Up-State     Selected
M  5           PASS              YES
   6           PASS              YES
```

In this example, **mdg** found two processors in the system, both passed power-up diagnostics, and as a result both were selected for inclusion in the list of available processors. In the case of a processor failing power-up diagnostics, **mdg** will not include it as one of the SELECTED processors. However, **mdg** provides to the user the capability to attempt to include a processor that failed power-up diagnostics at any time.

14.3 The Prompt

The **mdg** prompt follows the following format:

```
{ CPUs not included during test / CPUs included during test } <Pass limit> =>
```

For example: In a system with two processors (in slots 5 and 6), with only the processor in slot 6 to be included during testing, and the pass limit set to 1 the prompt to be displayed will be as follows:

```
          { 5/6 } <1> =>
```

14.4 mdg Tests

the **mdg** (1) debugger tests include:

1. Atomic Load-Store Test
2. Memory Data RAM Test
3. Shared-Memory Pattern Test
4. Cache Block Alias Test
5. Floating Point Store Test
6. Cache Data Request Test
7. Cache Data Bus Pattern Test
8. Interrupt Test

☆ ☆ ☆ **NOTE** ☆ ☆ ☆

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.

14.5 mdg Commands

A summary of command usage is displayed on-line when **mdg** is running by typing:

```
{ /5 6 } <1> => ?.
```

See Section 13.2.1 for a brief description of the **mdg** commands. The following is a listing of the **mdg** commands available:

between (1)	**config** (1)	**continue** (1)
cpus (1)	**cpulim** (1)	**deposit** (1)
errlim (1)	**errors** (1)	**examine** (1)
fbconfig (1)	**halt** (1)	**help** (1)
limit (1)	**loop** (1)	**master** (1)
mdg (1)	**memconfig** (1)	**menu** (1)
names (1)	**next** (1)	**passes** (1)
passlim (1)	**prompt** (1)	**quiet** (1)
quit (1)	**restart** (1)	**run** (1)
status (1)	**time** (1)	**tests** (1)
vmeconf (1)	**wake** (1)	

The command set is similar to the **rdg** command set with the following additions:

cpus	Will select or display the processors included in tests.
cpulim	Will display or set processor specific memory test limits
halt	Will halt processors from the **mdg** environment.

master	Displays or changes the master processor
memconfig	Displays the system memory configuration.
wake	Adds a processor to the **mdg** environment.

14.6 CPU LEDs Displays with mdg Invoked

The CPU LEDs display the following with **mdg** invoked:

- `f0` Making service request while in IDLE state.
- `f1` Making service request while in ACTIVE state.
- `f2` Making service request while in MONITOR state.
- `f5` Defective test error.
- `f8` Waiting to lock SYNC symaphore.
- `f9` Waiting for other processors to get SYNC'd.

14.7 Field Notes

Section 15: Field Notes

15.1 Introduction

This section offers room for SEs to make notes about this reference guide; what is useful, what is not useful and what should be included in the next update of the manual.

ROM> romcopy,,(DESINATION SLOT)

EXP ROM> romcopy,,5

Index

T

V

W

Y

Power-up status indicators for the CPU board:

- **00** ROM started executing
- **90** Bad IDPROM checksum
- **a0** Master failed
- **a1** Relinquishing mastership
- **a2** Slave received mastership
- **a3** Timeout while giving mastership
- **a4** Awaking slave CPU
- **a5** Slave CPU received slave command
- **a6** Slave CPU passed power-up tests
- **a7** Slave failed
- **a8** Slave CPU failed power-up tests
- **a9** Timeout while waking slave CPU
- **ab** Burn-in jumper detected, looping
- **ad** ROM power-up tests completed
- **ae** Initializing ECC

ROM initialization LED codes:

- **b0** ROM main() started
- **b1** Initializing I/O mapping addresses
- **b2** Bad EEROM checksum
- **b3-9** Initializing EEPROM, IOB's, devices, stdin, stdout, stderr, file systems
- **b6** keyboard initialization failure, check the keyboard cable
- **bc** Could not open console device
- **bd** Initializing main before cmdloop
- **be** Waiting for command from console
- **bf** Executing a command
- **c0** Standalone crt0 starting
- **c1** Standalone crt0 calling main
- **c8** No System Board found
- **ce** FCR no zero on reset
- **cf** Executing a reset halt

LED codes for devices (displayed during device initialization):

- **d0** Simulated UART
- **d1** Simulated disk
- **d2** LANCE Ethernet
- **d3** Real Time clock
- **d4** RAM disk
- **d6** VME-to-SCSI controller
- **d7** UART driver
- **d8** Keyboard/mouse
- **d9** Frame buffer

Power-Up Self Tests

- The test numbers and their purpose follow:

Test No.	Test Purpose
Blank	No power
00	CPU alive and Self-Test started
01	BootROM checksum
02	Diagnostic RAM (2K)
03-04	Interrupt Registers and Priority Masking
05-06	CPU control and data busses
07-0a	TLB uniqueness, RAM, Physical Address TAGS
0b-0d	Byte, 1/2 word, word, double word, Cache and RAM
0e-19	Memory Management Unit (MMU)
1a	Bus timeout, tries to access slot 0
1b	Probe all slots and build configuration table in diagnostic RAM Check status of burn-in jumper Must have CPU, Memory, and I/O to proceed
1d	Determine Master
1e-21	Bus Watcher Tags status and RAM
22-26	Memory Addressing - all bases to 4 Gbytes in 256 Mbyte steps Memory PCB uniqueness Low digit of error code is slot #
***27**	1 Mbyte only unless Burn-in set
28-2e	Cache tests
2f-33	Memory ECC paths and correction
***34**	ECC RAM - 1 Mbyte only unless Burn-in