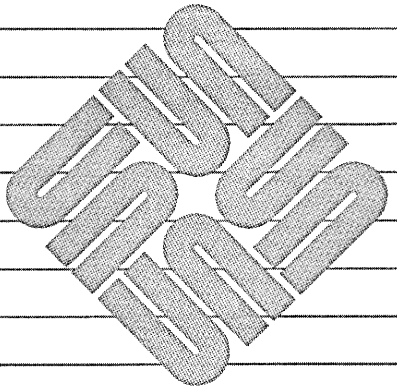
sun®
microsystems

PROM User's Manual



Part Number: 800-1736-10
Revision A, of 9 May 1988

Sun is a trademark of Sun Microsystems, Inc.

Sun Microsystems and **Sun Workstation** are registered trademarks of Sun Microsystems, Incorporated.

The Sun logo is a registered trademark of Sun Microsystems, Inc.

Sun-2, Sun-3 and Sun-4 are trademarks of Sun Microsystems, Inc.

Sun386*i* is a trademark of Sun Microsystems, Inc.

SunOS is a trademark of Sun Microsystems, Inc.

SunIPC is a trademark of Sun Microsystems, Inc.

ALM and **ALM2** are trademarks of Sun Microsystems, Inc.

DVMA is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark of AT&T.

Multibus is a trademark of Intel Corporation

Tapemaster is a trademark of Ciprico, Inc.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Copyright © 1987, 1988 by Sun Microsystems, Inc.

This publication is protected by Federal Copyright Law, with all rights reserved. No part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, or by any means manual, electric, electronic, electro-magnetic, mechanical, chemical, optical, or otherwise, without prior explicit written permission from Sun Microsystems.

Contents

Chapter 1 Overview **3**

1.1. Types of PROMs 3

1.2. Booting Procedures 4

1.3. Sun Architectures 4

1.4. PROM Revisions 5

1.5. Test Equipment 6

- Loopback Connector for Ports A and B 6
- Loopback Connector for Keyboard/Mouse Ports 6
- Sun-3 Loopback Connector for SCSI Bus Connector 7

Chapter 2 Booting Procedures **11**

2.1. The Power-on Sequence 11

- The Diagnostics Switch 12
  - Switch Location 13
- Power-up Tests 13
  - Minimum System Functionality 13
  - Memory Testing 13
  - Test Run Through 13

2.2. Booting the SunOS Operating System 16

- Interrupting the Boot Sequence 17

2.3. Booting From the Monitor 17

Chapter 3 Starting the PROM Monitor **21**

3.1. Power is OFF 21

3.2. Halting the SunOS Operating System 22

Chapter 4 Sun-2 Self-tests and Initialization **25**

4.1. The Power-Up Test Sequence 25

  Power-Up Tests 26

4.2. Host System Initialization 28

4.3. The Boot Sequence 29

Chapter 5 Sun-2 PROM Monitor Commands **33**

5.1. Introduction 33

    Bringing up the PROM Monitor 33

  Old vs New PROMs 33

5.2. Monitor Command Overview 33

  Executing a Command 34

  Default Values 34

  Word Sizes 34

5.3. The Sun-2 Monitor Commands 35

  Displaying and Modifying 35

  Conventions 37

  Special Commands for New Sun-2 PROMs 37

    Address Increment/Decrement Command 37

    The `^T` Command 38

    The `^I` Command 38

    The `^C` Command 39

  Regular Sun-2 Monitor Commands 39

    Monitor `a` Command 39

    Monitor `b` Command 39

    Monitor `c` Command 40

    Monitor `d` Command 40

    Monitor `e` Command 40

    Monitor `f` Command 41

    Monitor `g` Command 41

    Monitor `h` Command 41

Monitor `k` Command 41
Monitor `l` Command 42
Monitor `m` Command 42
Monitor `o` Command 42
Monitor `p` Command 42
Monitor `r` Command 42
Monitor `s` Command 43
Monitor `u` Command 43
Monitor `v` Command 45
Monitor `w` Command 45
Monitor `x` Command 45

Chapter 6 Sun-2 Extended Test Sequence **49**
6.1. Introduction 49
6.2. Extended Test Menu Structure 50
Main Menu 50
6.3. Sun-2 Extended Test Sub-Menus 52
Intel Ethernet Menu 52
Memory Test Menu 54
Mouse/Keyboard Menu 56
Serial Ports Menu 57
Video Test Menu 58
Options Menu 60

Chapter 7 Sun-3 Self-tests and Initialization **63**
7.1. The Power-Up Test Sequence 63
7.2. The Boot Sequence 64
Diagnostic Power-Up 65
Remote Testing 66
Diagnostic Self-test Sequence 67
Test Descriptions 69
Diagnostic Register Test 69
SCC (Z8530) Wr/Rd Test Through MMU 70

Boot PROM Checksum Test 70
DVMA Register Test 71
Context Register Test 71
Segment Map RAM Wr/Rd Test 71
Segment Map Test 72
Page Map Test 72
Memory Path Data Test 73
NXM Bus Error Test 73
Interrupt Test 73
TOD Clock Interrupt Test 74
MMU Protection/Status Tests 74
Parity Error Tests (Not for Sun3/2xx) 75
Parity Error Tests (Not for Sun-3/2xx) 75
ECC Error Tests (Sun-3/2xx only) 75
Cache Data 3-Pattern Test (Sun-3/2xx Only) 76
Cache Tags 3-Pattern Test (Sun-3/2xx Only) 76
Memory Sizing (Sun-3/2xx Only) 76
ECC Memory Initialization (Sun-3/2xx only) 77
Memory Modulo 3's Tests (Sun-3/2xx Only) 77
Non Sun-3/2xx Memory Tests 78
Memory Test (Non Sun-3/2xx CPU Boards) 78

Chapter 8 Sun-3 PROM Monitor Commands **81**
Bringing up the PROM Monitor 81
8.1. Monitor Command Overview 81
Executing a Command 82
Default Values 83
Word Sizes 83
8.2. The Monitor Commands 83
Displaying and Modifying Memory 83
Conventions 85
Special Monitor Commands 86
Address Increment/Decrement Command 86

The `^T` Command ... 86
The `^I` Command ... 87
The `^C` Command ... 87
Regular Monitor Commands ... 87
Monitor `a` Command ... 87
Monitor `b` Command ... 87
Monitor `c` Command ... 89
Monitor `d` Command ... 89
Monitor `e` Command ... 89
Monitor `f` Command ... 90
Monitor `g` Command ... 90
Monitor `h` Command ... 90
Monitor `i` Command — Sun-3/2xx Only ... 90
Monitor `j` Command — Sun-3/2xx Only ... 91
Monitor `k` Command ... 91
Monitor `l` Command ... 91
Monitor `m` Command ... 91
Monitor `n` Command — Sun-3/2xx Only ... 91
Monitor `o` Command ... 91
Monitor `p` Command ... 92
Monitor `q` Command ... 92
Monitor `r` Command ... 92
Monitor `s` Command ... 93
Monitor `u` Command ... 93
Monitor `v` Command ... 95
Monitor `w` Command ... 95
Monitor `x` Command ... 96
Monitor `y` Command — Sun-3/2xx Only ... 96
Monitor `z` Command ... 96

Chapter 9 Sun-3 Extended Test Sequence ... **101**
9.1. Extended Menu Test Descriptions ... 101
Main Menu ... 102

9.2. Sun-3 Extended Test Sub-Menus 107
  AMD Ethernet Test Menu 107
  Intel Ethernet Test Menu 108
  Memory Menu 109
  Mouse/Keyboard Ports Test Menu 111
  Serial Ports Menu 114
  SCSI Interface Menu 117
  Video Menu 118
  Monochrome Video Menu 120
  Video Color Menu 122
  Enable Plane Menu 124
  Color Map Menu 126
  Video Overlay Plane Test Menus 127
  Test Options Submenu 128

Chapter 10 Sun-3 EEPROM Layout **131**

10.1. EEPROM Introduction 131
10.2. Changing EEPROM Parameters 131
10.3. The EEPROM Layout 132
  Diagnostic EEPROM 134
  System Configuration 145
    Configuration Block Examples 148
    How to Program the CPU Configuration Block 150
  Reserved Area 168
  Reserved Area 168
  ROM Area 169
  ROM Area 169
  Software Area 169
  Software Area 170

Chapter 11 Sun-4 Self-tests and Initialization **173**

11.1. The Power-Up Test Sequence 173
    Remote Testing 176

11.2. Diagnostic LED Interpretation 180

- Diagnostic Register Test 181
- Boot PROM Checksum Test 181
- UDVMA Enable Register Test — Sun-4/2xx Only 181
- UDVMA Map Write-Write-Read Test — Sun-4/2xx only 182
- UDVMA Map Address Test — Sun-4/2xx Only 182
- UDVMA Map 3-Pattern Test — Sun-4/2xx Only 182
- Context Register Test 183
- Segment Map Write-Write-Read Test 183
- Segment Map Address Test 184
- Segment Map 3-Pattern Test 184
- Page Map Write-Write-Read Test 184
- Page Map Address Test 185
- Page Map 3-Pattern Test 185
- Software Traps Test 185
- Interrupt Register Test 186
- Software Interrupts Test 186
- TOD Interrupt Test 187
- Video Memory Write-Write-Read Test 187
- Video Memory Address Test 187
- Video Memory 3-Pattern Test 188
- P4 Color Map Test — Sun-4/1xx only 188
- Limited Main Memory Write-Write-Read Test 189
- Limited Main Memory Address Test 189
- Limited Main Memory 3-Pattern Test 189
- MMU Read Access/Modified Bits Test 190
- MMU Write Access/Modified Bits Test 190
- MMU Write To Write Protected Page Test 190
- MMU Read Not-Writable Invalid Page Test 191
- MMU Read Writable Invalid Page Test 191
- MMU Write Not-Writable Invalid Page Test 191
- MMU Write Writable Invalid Page Test 192
- Main Memory Timeout Test 192

Control Space Timeout Test 192
Range Error Test 193
Size Error Test 193
Parity No Fault Test — Sun-4/1xx Only 194
Parity Error Detection and Interrupt Test — Sun-4/1xx Only 194
Parity Error Detection Without Interrupt — Sun-4/1xx 194
Cache Tag Bank Test, Alternating Bit Pairs <31:11> — Sun-4/1xx Only 194
SCSI/DMA Tests — Sun-4/1xx Only 195
DMA Address Register Test — Sun-4/1xx Only 195
DMA Byte Counter Register Test — Sun-4/1xx Only 195
DMA Send and Byte Pack Register Test — Sun-4/1xx Only 195
DMA Conflict Interrupt Check — Sun-4/1xx Only 195
DMA From Invalid Page — Sun-4/1xx Only 195
ECC Circuit Test — Sun-4/2xx Only 196
Cache Tag Memory Write-Write-Read Test — Sun-4/2xx Only 196
Cache Tag Memory Address Test — Sun-4/2xx Only 196
Cache Tag Memory 3-Pattern Test — Sun-4/2xx Only 196
Cache Data Memory Write-Write-Read Test — Sun-4/2xx Only 197
Cache Data Memory Address Test — Sun-4/2xx Only 197
Cache Data Memory 3-Pattern Test — Sun-4/2xx Only 197
Cache Write/Read Hit/Miss Verify Test 198
Cache Write/Read/Flush/Verify Test — Sun-4/2xx Only 199
Main Memory Tests 200
Main Memory Write-Write-Read Test — Diagnostic Mode Only 200
Main Memory Address Test 200
Main Memory 3-Pattern Test — Diagnostic Mode Only 201
Main Memory Ones Complement Address Test — Normal Mode Only 201
11.3. Host System Initialization 201

11.4. The Boot Sequence 203

Chapter 12 Sun-4 PROM Monitor Commands 207

12.1. What's In This Chapter 207
  Bringing up the PROM Monitor 207
12.2. Monitor Command Overview 207
  Getting Help 208
    Sun-4/1xx Additional Menu 208
  Executing a Command 209
  Default Values 209
  Word Sizes 209
12.3. The Monitor Commands 209
  Displaying and Modifying Memory 209
  Special Monitor Commands 212
    Address Increment/Decrement Command 212
    The `^T` Command 212
    The `^I` Command 213
    The `^C` Command 213
  List of Sun-4 Monitor Commands 213
    Monitor `b` Command 213
    Monitor `c` Command 215
    Monitor `d` Command 215
    Monitor `e` Command 216
    Monitor `f` Command 216
    Monitor `g` Command 216
    Monitor `h` Command 217
    Monitor `i` Command 217
    Monitor `j` Command 217
    Monitor `k` Command 217
    Monitor `l` Command 218
    Monitor `m` Command 218
    Monitor `n` Command 218
    Monitor `o` Command 218

Monitor **p** Command ... 218
Monitor **q** Command ... 218
Monitor **r** Command ... 219
Monitor **s** Command ... 220
Monitor **u** Command ... 220
Monitor **v** Command ... 222
Monitor **w** Command ... 222
Monitor **x** Command ... 223
Monitor **y** Command ... 223

Chapter 13 Sun-4 Extended Test System ... **227**
13.1. The User Interface ... 227
? Command ... 227
q Command ... 227
Escape Command ... 227
Loop Command ... 227
Control-C Command ... 228
! Command ... 228
The Command Line ... 228
13.2. Extended Test Descriptions ... 229
Main Menu ... 230
All ... 230
Default ... 231
Boot ... 231
Ethernet ... 232
Keyboard ... 232
Memory ... 232
Serial ... 232
Video ... 232
Colormap ... 232
Options ... 232
Boot Paths Menu ... 233
Sd Bootpath Test ... 233

  `ST` Bootpath Test 233
  `Xy` Bootpath Test 234
  `XT` Bootpath Test 234
  `Options` Command 234
Ethernet Menu 235
  `LOCal` Command 235
  `Encoder` Command 235
  `EXternal` Command 236
  `Options` Menu 236
Keyboard and Mouse Menu 237
  `Register` Command 237
  `Transmit` Command 237
  `Internal` Command 238
  `External` Command 239
  `Keyboard` Command 240
  `Options` Command 240
Memory Menu 241
  `Address` Test 241
  Constant Test 242
  `CHeck` Command 242
  `Fill` Command 243
  `Options` Menu 243
Serial Ports Menu 244
  `Register` Command 244
  `Transmit` Command 244
  `Internal` Command 245
  `External` Command 246
  `Options` Menu 247
Video Menu 248
  `Address` Test Command 248
  `Constant` Command 249
  `CHeck` Command 249
  `Fill` Command 250

`Options` 250
Color Map Menu 251
Options Menu 251
`Default` 251
`Stop=` 251
`Pass=` 251
`Report=` 252
`SCope=` 252
`Cell=` 252

Chapter 14 Sun-4 EEPROM Layout **255**
14.1. EEPROM Introduction 255
14.2. Changing EEPROM Parameters 255
14.3. The EEPROM Layout 256
Diagnostic EEPROM 257
Test Write Area 257
System Configuration 268
How to Program the CPU Configuration Block 272
Reserved Area 290
Reserved Area 290
ROM Area 291
ROM Area 291
Software Area 291
Software Area 292

Chapter 15 Sun386i Self-tests and Initialization **295**
15.1. Hardware Requirements 295
CPU Board Jumper Settings 296
15.2. Memory Test Options 296
15.3. Functional Requirements 296
Normal Boot 297
Diagnostic Boot 298
15.4. Power-On Theory of Operation 298

System Reset Analysis: Hard and Soft Resets 298
Determination of Run-time Mode 298
    Normal (Default) Mode 298
    Diagnostic Mode 299
    Manufacturing (Burn-In) Mode 300
Power On Self Test (POST) 300
    Error Classification and Reporting 301
Test Descriptions 306
    CPU Reset Test 306
    LED Register Test 307
    Verify Wait State Registers Reset 307
    Verify Refresh Register Reset 307
    Verify Relocation Register Reset 307
    Verify 82380 Relocation 307
    Verify DMA Controller Reset 307
    Boot PROM Checksum Test 307
    Serial Port A Internal Loopback Test 307
    NVRAM Diagnostic Area Test 307
    MUSCI Diagnostic Register Test 308
    Refresh (Timer 1) Downcount Test 308
    Start Refresh Timer, Wait State, Refresh Regs. 308
    Timer 0 Downcount Test 308
    Speaker Timer (2) Downcount Test 308
    Clock Tick Timer (3) Downcount Test 308
    Cache MBAR Test 308
    RAM Bus Access Test 309
    RAM Bus Data Path Test 309
    RAM Address Test 309
    RAM Data Test - Phase I 309
    Bus Sizing Routine 309
    Video RAM Address Test 310
    Video RAM Data Test 310
    Color LUT RAM Test (Only for Color Systems) 310

Keyboard/Mouse Internal Loopback Test 310
Keyboard Reset Test 310
Video Initialize Sequence 310
Verify Interrupt Vector Registers 311
Verify Interrupt Request Lines 311
Verify Interrupt Occurs & IRQ8 Status 311
Verify Interrupt Masking 311
RAM Data Test - Phase II 311
DRAM Refresh Test 311
RAM Parity Test 311
ID PROM Checksum Test 311
DMA Verify Transfer Mode Test 312
DMA Block Transfer Test 312
Boot Path Device Checkout 312
Phase 2 Confidence Test Begins At This Point 312
Parallel Port Test 312
Cache CTAGS Test 312
Cache Static RAM Test 312
Optional RAM Tests (After 1 Mbyte) 313
Floppy Controller Check 313
Ethernet Internal Loopback Test 313
Hard Disk Controller Check 313
Tape Drive Controller Check 313
Initialization of System Parameters 313
Sun386i Future Option Board Feature 315

Chapter 16 Sun386i Monitor Commands **319**
16.1. Conventions 319
Special Monitor Commands 319
Address Increment/Decrement Command 319
The `^T` Command 320
The `^I` Command 320
The `^C` Command 321

Displaying and Modifying Memory 321

16.2. List of Sun386i Monitor Commands 323

Monitor `b` Command 323

Monitor `c` Command 325

Monitor `d` Command 325

Monitor `e` Command 326

Monitor `f` Command 326

Monitor `g` Command 326

Monitor `h` Command 327

Monitor `j` Command 327

Monitor `k` Command 327

Monitor `l` Command 328

Monitor `n` Command 328

Monitor `o` Command 328

Monitor `p` Command 328

Monitor `q` Command 328

Monitor `r` Command 329

Monitor `s` Command 329

Monitor `u` Command 329

Monitor `v` Command 331

Monitor `w` Command 331

Monitor `z` Command 332

Chapter 17 Sun386i Extended Menu Tests **335**

17.1. General Description 335

17.2. Test Fixtures 335

Loopback Connector for Serial Port A 335

Loopback Connector for Keyboard/Mouse Ports 335

Loopback Connector for SCSI Bus 336

Loopback Connector for Parallel Port 336

17.3. Sun386i Extended Test Main Menu 337

Test Descriptions 337

Intel Ethernet Test Menu 338

Memory Menu 339
Keyboard and Mouse Menu 341
Serial Port Menu 344
SCSI Interface Menu 346
Monochrome Video Menu 347
Color Video Menu 349
Color Map Menu 351
Parallel Port Test Menu 351
Test Options Submenu 352

Chapter 18 Sun386i NVRAM Layout **355**

18.1. NVRAM Introduction 355
18.2. Changing NVRAM Parameters 356
18.3. The NVRAM Layout 356
Diagnostic NVRAM 356
Test Write Area 356
System Configuration 365
Remaining NVRAM Layout 366
Reserved Area 369
Reserved Area 370
Scratch Area 370
Software Area 370
Software Area 370

Appendix A ASCII/Hex Conversion Chart **373**

Tables

Table 1-1 Serial Port Loopback Connections 6
Table 1-2 Sun-2 Loopback Wiring 6
Table 1-3 Sun-3 SCSI Loopback Wiring 7

Table 2-1 Boot Devices 18

Table 5-1 Miscellaneous Registers for the MC68010 42
Table 5-2 Function Code Values 43
Table 5-3 Port Arguments 44
Table 5-4 Option Arguments 44

Table 8-1 Miscellaneous Registers for the 68020 92
Table 8-2 Function Code Values 93
Table 8-3 Port Arguments 94
Table 8-4 Option Arguments 94

Table 10-1 Default System Configuration Parameters 133
Table 10-2 Configuration Block Layout *Address*[0x0BC-0x18B] 146
Table 10-3 Board Type Values 147
Table 10-4 **No-Board Configuration Block** 149
Table 10-5 **CPU Board Configuration Block** 150
Table 10-6 **Memory Board Configuration Block** 152
Table 10-7 **Color Board Configuration Block** 153
Table 10-8 **Frame Buffer Board Configuration Block** 153
Table 10-9 **FPA Board Configuration Block** 154

Table 10-10 **SMD Disk Board Configuration Block** 155
Table 10-11 **1/2 Inch Tape Controller Configuration Block** 156
Table 10-12 **Second Ethernet Controller Board Configuration Block** 157
Table 10-13 **MTI/ALM Board Configuration Block** 158
Table 10-14 **GP Board Configuration Block** 158
Table 10-15 **SCP Board Configuration Block** 159
Table 10-16 **SCSI Board Configuration Block** 160
Table 10-17 **SunIPC Board Configuration Block** 161
Table 10-18 **GB Board Configuration Block** 162
Table 10-19 **3/75 SCSI Memory Board Configuration Block** 163
Table 10-20 **MAPKIT Configuration Block** 164
Table 10-21 **Channel Adapter Configuration Block** 164
Table 10-22 **ALM-2 Configuration Block** 165
Table 10-23 **MCP/SCP-2 Configuration Block** 165
Table 10-24 **Sentinel Block** 166

Table 12-1 Register Numbers 219
Table 12-2 ASI Values 220
Table 12-3 Port Arguments 221
Table 12-4 Option Arguments 221

Table 14-1 Default System Configuration Parameters for Sun-4 Systems 257
Table 14-2 Configuration Block Layout *Address*[0x0BC-0x18B] 269
Table 14-3 Board Type Values 270
Table 14-4 **No-Board Configuration Block** 271
Table 14-5 **CPU Board Configuration Block** 272
Table 14-6 **Memory Board Configuration Block** 274
Table 14-7 **Color Board Configuration Block** 275
Table 14-8 **Frame Buffer Board Configuration Block** 275
Table 14-9 **FPA Board Configuration Block** 276
Table 14-10 **SMD Disk Board Configuration Block** 277
Table 14-11 **1/2 Inch Tape Controller Configuration Block** 278

Table 14-12 **Second Ethernet Controller Board Configuration Block** 279
Table 14-13 **MTI/ALM Board Configuration Block** 280
Table 14-14 **GP Board Configuration Block** 280
Table 14-15 **SCP Board Configuration Block** 281
Table 14-16 **SCSI Board Configuration Block** 282
Table 14-17 **SunIPC Board Configuration Block** 283
Table 14-18 **GB Board Configuration Block** 284
Table 14-19 **3/75 SCSI Memory Board Configuration Block** 285
Table 14-20 **MAPKIT Configuration Block** 286
Table 14-21 **Channel Adapter Configuration Block** 286
Table 14-22 **ALM-2 Configuration Block** 287
Table 14-23 **MCP/SCP-2 Configuration Block** 287
Table 14-24 **Sentinel Block** 288

Table 15-1 LED Subsystem Code Interpretation 303
Table 15-2 POST LED Interpretation 304
Table 15-3 Initialization Values 314

Table 16-1 Port Arguments 330
Table 16-2 Option Arguments 330

Table 17-1 Serial Port Loopback Signals 335
Table 17-2 Keyboard/Mouse Loopback Signals 336
Table 17-3 SCSI Bus loopback signals 336
Table 17-4 Parallel Port Loopback Signals 336

Table 18-1 **Sun386i CPU Configuration Area** 365

Table A-1 ASCII/Hex Conversion 373

Figures

Figure 2-1 The Power-on Sequence (Sun-3/4 Only) 12

Figure 5-1 Monitor Help Menu (New Sun-2 PROMs Only) 34

Figure 7-1 Sun-3/75, 3/140, 3/150, 3/160, and 3/110 Diagnostic Boot Sequence 67
Figure 7-2 Sun-3/50 and Sun-3/60 Diagnostic Boot Sequence 68
Figure 7-3 Sun-3/260 and Sun-3/280 Diagnostic Boot Sequence 69

Figure 8-1 Sun-3 Monitor Help Menu 82

Figure 11-1 Sun-4/2xx LED Interpretation 179
Figure 11-2 Sun-4/1xx LED Interpretation 180

Figure 12-1 Monitor Help Menu 208

Figure 15-1 Sun386i Jumper Location 296
Figure 15-2 Sample Normal Sun386i Start-Up Display 299
Figure 15-3 Sample Diagnostic Mode Sun386i Start-Up Display 300
Figure 15-4 Normal Mode Sun386i POST Error Handling 302
Figure 15-5 Sun 386i LED Display Interpretation 302

Figure 16-1 Sun386i Monitor Help Menu 327

1

Overview

Overview 3

1.1. Types of PROMs 3
1.2. Booting Procedures 4
1.3. Sun Architectures 4
1.4. PROM Revisions 5
1.5. Test Equipment 6

1

Overview

This chapter describes what PROMs (Programmable Read Only Memory chips) do in Sun systems. It also describes PROM topics found in this manual.

1.1. Types of PROMs

There are a number of different PROMs found on the CPU Board in Sun systems. They include:

- Boot PROMs — contain the boot and diagnostic programs.
- ID PROMs — contain the system's serial number, Ethernet address, and information on system configuration.
- EEPROMs — contain current system configuration (can be altered).
- NVRAMs — perform the same function as the EEPROMs.
- Hardware PROMs — part of the hardware logic; not covered in this manual.

Here is a little more information on each of the PROM types:

Boot PROMs

These PROMs contain the *PROM monitor* program, a command interpreter used for booting, resetting, low-level configuration, and simple test procedures. The newer Sun PROMs also contain the `Extended Test System`, which is a menu-driven diagnostic system. The Boot PROMs also contain power-on tests that are automatically run when the machine is reset. These tests check CPU and Memory Board functions that are needed for program loading and execution.

ID PROMs

These PROMs contain the following information that is unique to a particular system:

- Workstation Serial Number
- Workstation Ethernet Address
- System Configuration Information

The serial number is used to track the system, and to provide information for certain copy protected application programs. The Ethernet address is used by the network software to identify a system during boot-up.

The ID PROM is not discussed further in this document.

EEPROMs and NVRAMs

The EEPROM (Electrically Erasable PROM) and NVRAM (Non-Volatile RAM) contain information about the current system configuration, alternate boot paths, and so on. Information can be written to as well as read from these devices. Most of the information must be entered manually, and updated each time the system configuration is changed (i.e. adding or removing boards, changing monitor types, and so on). The information in these devices can be read or written using a PROM monitor command or the Diagnostic Executive appropriate for your system. The configuration format differs in small ways between various CPU board architectures, so read the chapter that applies to your system before modifying EEPROM contents.

1.2. Booting Procedures

A separate chapter is devoted to boot procedures. This chapter covers both power-up and booting (normal and diagnostic) as well as the PROM monitor's boot command.

1.3. Sun Architectures

Since the PROM contents are hardware dependent, the PROM contents change significantly between system architectures. This manual is divided into sections by Sun architecture, to make reference easier. There are currently five Sun architectures, briefly described below.

Sun-1

The Sun-1 architecture has a smaller, 700 x 800 pixel screen, and steel, rather than plastic workstations. It uses a Multibus backplane. The Sun-1 workstations include the Sun-100U desktop workstation, and the Sun-150U Server. These systems use the MC68010 CPU. Sun-1 architecture is not covered in this manual.

Sun-2

The Sun-2 workstation has a full size 900 x 1152 pixel screen. Sun-2's use a Multibus backplane, or in limited cases, VMEbus. Sun-2s may be desktop, deskside, and fileserver models. Typical workstations include the Sun-2/120, Sun-2/50, and the Sun-2/160. All Sun-2's use the MC68010 CPU. The Sun-2's have a PROM monitor and automatic self-tests. They **don't** have an EEPROM or a diagnostic switch.

Old vs New

There are two major versions of Sun-2 PROMs. The old version has shipped until the 4.0 SunOS software release. The new version, incorporating a number of new features, will be shipped with all Sun-2's after the 4.0 software release.

The two types of PROMs are easy to distinguish from one another. Start up the PROM monitor, (see *Chapter 3* for details), then type

```
>kb [Return]
```

This command displays the PROM banner. Look for the PROM revision number in the banner message. If the revision is R, Q or any other revision *letter*, your system CPU board contains the old PROMs. If it is 1.1, or any other revision *number*, your system contains the new PROMs. The difference between the old and new PROM versions lie in the Monitor commands and extended tests. Some new commands have been added, and only the new PROM has extended tests. See chapter 5 for details on the differences between the two versions.

Sun-3

The Sun-3 workstation uses the MC68020 processor and the VMEbus for higher performance. The Sun-3 line is otherwise very similar to the Sun-2 line, in that there are desktop, deskside, and server versions. Typical Sun-3's include: Sun-3/50M, Sun-3/75, Sun-3/110, Sun-3/150, Sun-3/160, and the Sun-3/200 family. All Sun-3's contain an EEPROM to hold configuration information, an improved PROM monitor, and the Extended Test Sequence. The Sun-3/110 introduced the grayscale monitor and color circuitry on the CPU board. The Sun-3/200 workstations feature 8-Megabyte ECC memory boards.

Sun-4

The Sun-4 uses a proprietary Scalable Processor Architecture SF9010 CPU with an on-board Floating Point Unit. It also uses the VMEbus, and provides fast integer and floating point performance. Like the Sun-3, the Sun-4 contains an EEPROM and the Extended Test system. Both the PROM monitor and the test system are easier to use and more consistent in the Sun-4, when compared to previous versions.

Sun386i

The newest of the Sun workstations uses the Intel 80386 processor and features a floppy disk as well as a hard disk, and a range of video monitor types. Its NVRAM stores the same type information as the EEPROM. The Sun386i also has an extended test system. Two more features unique to this architecture are an on-line help system and slots for AT-compatible hardware.

1.4. PROM Revisions

The Sun PROMs, particularly the Boot PROMs, change with every new architecture. However, significant bug fixes or enhancements may be made to the Boot PROM for an architecture that is already in service. In these cases, the revision number of the Boot PROM becomes important. In this manual, Boot PROM revisions are mentioned if they are required; otherwise naming the architecture should suffice. Use the monitor `kb` command to determine a Boot PROM revision level (refer to the *PROM Monitor Commands* chapter for your system.

If you have a Sun-3 or Sun-4 system and are installing SunOS version 4.0, your boot PROM MUST be Revision 1.8 or later.

If you have a Sun386i Revision 1.5 CPU board with a Revision 4.1 Boot PROM and your boot fails, you may need to set an NVRAM location that specifies the CPU board revision level. Look at the power-up banner to determine your Boot PROM revision level. If you have a 4.1 PROM, refer to *Chapter 16* for information on using the monitor `q` command. Then open location 0x111 and set it to "01" for a 1.5 CPU board revision level, or "02" for a Revision 2.0. You may need to try setting it each way and rebooting to determine which board version you have.

1.5. Test Equipment

Most of the tests run by the PROMs need no external equipment. However, the extended test system requires special test fixtures, which are described next.

Since each architecture may have different I/O connections, the loopback connectors are separated according to architecture in the text that follows.

Loopback Connector for Ports A and B

A loopback connector is required at the handle edge of the CPU Board, to test Serial Ports A and B, using the Extended Menu Tests invoked by the x Monitor command. This shorting connector is a DB25P. A list of interconnects is provided below.

Table 1-1 *Serial Port Loopback Connections*

From Pin	*To Pin*
2(*TxD*)	3(*RxD*)
4(*RTS*)	5(*CTS*)
6(*DSR*)	20(*DTR*)

Loopback Connector for Keyboard/Mouse Ports

A loopback connector at the handle edge of the CPU board is required to test the Keyboard and Mouse ports, using the Extended Menu Tests (accessed with the x Monitor command). This shorting connector is a 15-pin (DB15P) connector. A list of interconnects is provided below.

This description applies only to Sun-2 systems that use the VMEbus, and to Sun-3 systems.

Table 1-2 *Sun-2 Loopback Wiring*

From Pin	*To Pin*
1(*RxD*)mouse	3(*TxD*)mouse
5(*RxD*)keybd	7(*TxD*)keybd

Sun-3 Loopback Connector for SCSI Bus Connector

You must install a loopback connector on the SCSI connector at the backpanel edge of the CPU Board in order to test the SCSI Bus, using the Extended Menu Tests provided by the `x` Monitor command. This shorting connector is a DB50 connector. A list of interconnects is provided below. The set of chip and connector (***Conn***) pins on the left side connect to those on the right side of the table.

Table 1-3 *Sun-3 SCSI Loopback Wiring*

From Signal	*From Chip Pin*	*From Conn Pin*	*To Conn Pin*	*To Chip Pin*	*To Signal*
ACK	14	38	16	2	DB7
BSY	13	36	14	3	DB6
SEL	12	44	12	4	DB5
ATN	15	32	10	5	DB4
REQ	20	48	8	6	DB3
MSG	19	42	6	7	DB2
C/D	18	46	4	8	DB1
I/O	17	50	2	9	DB0
RST	16	40	18	10	DBP

2

Booting Procedures

Booting Procedures **11**

2.1. The Power-on Sequence 11
2.2. Booting the SunOS Operating System 16
2.3. Booting From the Monitor 17

2

Booting Procedures

This chapter explains how Sun Workstations boot up. It covers both Power-Up boots, and boots from the PROM monitor.

The descriptions given here vary between system architectures. The Sun-2 series, for example, doesn't have an EEPROM or diagnostic switch. The Sun-3 and Sun-4 have nearly identical PROM features. For information on the Sun386i boot up and power-up options, refer to *Chapter 15*.

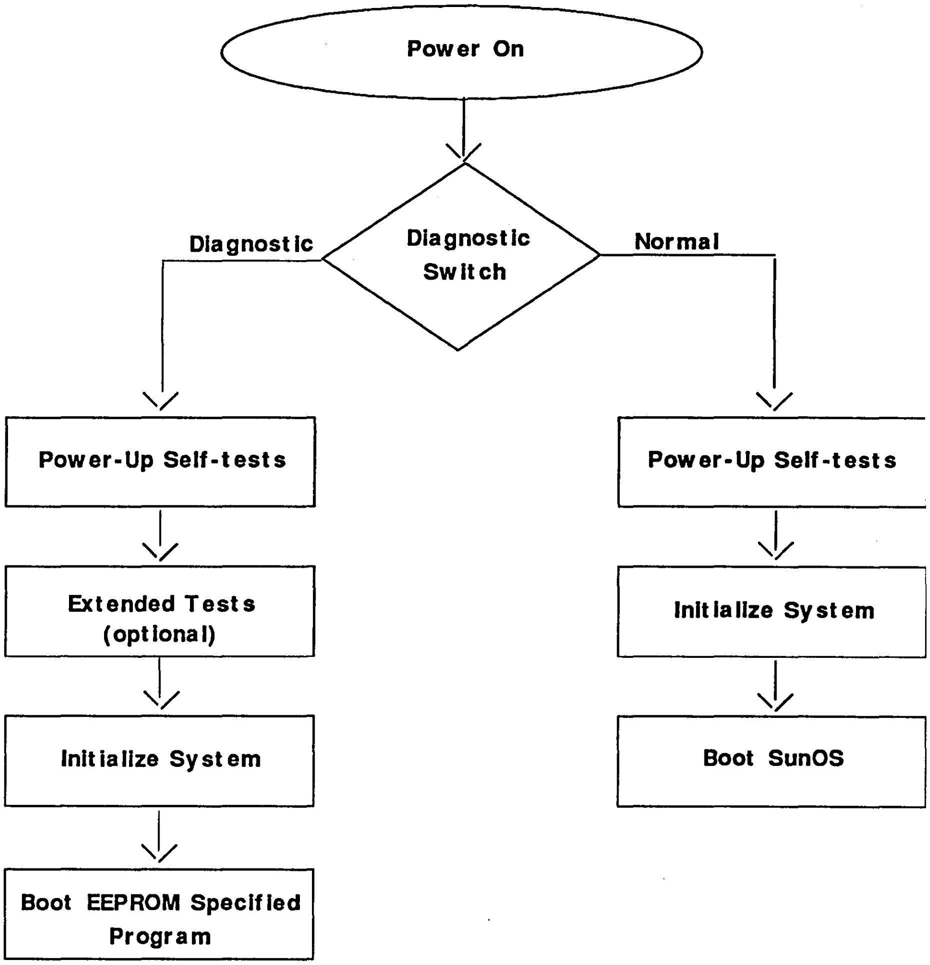
2.1. The Power-on Sequence

When a Sun system is powered-on, the CPU Board goes through a series of distinct phases. They are:

1. Reading the Diagnostic Switch (Sun-3 and Sun-4 only)
2. Power-up Tests
3. System Initialization
4. Boot of the SunOS operating system or, for Sun-3 and Sun-4, a user-defined program

The figure on the following page illustrates this sequence. *Chapter 15* describes a similar sequence for the Sun386i.

Figure 2-1 *The Power-on Sequence (Sun-3/4 Only)*


Power On
Diagnostic Switch
Diagnostic
Normal
Power-Up Self-tests
Extended Tests (optional)
Initialize System
Boot EEPROM Specified Program
Power-Up Self-tests
Initialize System
Boot SunOS

NOTE *This sequence varies with architecture. The Sun-2 has no diagnostic switch or EEPROM; therefore Step 1 is not applicable. Because it doesn't have an EEPROM, a Sun-2 always boots the SunOS operating system as the default — if you want to boot a different program or boot from a different path, you must abort the sequence and enter the boot command manually. For the Sun-386i boot sequence, refer to Chapter 15.*

The Diagnostics Switch

The diagnostics switch (on Sun-3's and Sun-4's) selects one of two power up sequences. The sequences differ in the type of self-tests that run, and which program is booted.

The Sun386i has jumpers and soft switches to select the "run-time" mode (refer to *Chapter 15* for more information.

The diagnostic switch is generally used to boot a diagnostic program if the workstation exhibits hardware problems. However, any bootable program on a legitimate boot device can be set to run if the diagnostics switch is set to DIAG.

Switch Location

The diagnostic switch is located on the back of the Sun-3 or Sun-4 CPU Board. The position of the switch varies with different workstation architectures. It should be marked NORM for "normal" and DIAG for "diagnostic". On some systems, the switch has a center position that will boot diagnostics, but is not intended for use.

Power-up Tests

The power up tests are divided into two groups; the normal and the diagnostic tests. User interaction depends on the diagnostic switch setting on Sun-3 and Sun-4 systems. These tests are designed to establish a minimum functionality of the system. They cover the CPU, its support logic and the Memory system, including the virtual memory support circuitry.

Minimum System Functionality

Power-up tests have an inherent limitation; the system must work to a minimum degree for the tests to run. The Sun power-up tests will run if the CPU and its support logic can fetch and execute instructions from the Boot PROM(s).

Memory Testing

The amount of memory tested during the self-test phase depends on the diagnostic switch setting for Sun-3 and Sun-4 systems. In Sun-2 systems all of memory is tested, and for Sun386i systems, the NVRAM setting determines how much memory is tested..

On Sun-3 and Sun-4 systems, if a switch is present and is set to diagnostic mode, all memory is tested. If the switch is in normal mode, the amount of memory tested depends upon values in EEPROM. The reason for this variation is the amount of time it takes to test memory.

Sun systems typically have a large amount of memory. The time it takes to test this memory, especially on a fully configured system, can be significant. For this reason we provide an EEPROM or NVRAM parameter that sets the quantity of memory to be tested. If you want a short boot time, set a smaller range. If a more thorough test is important, select a larger amount, or the entire memory. Even if a small amount is selected, you can reboot in diagnostic mode and test all of memory when a problem is suspected.

Stating some power-up execution times for Sun-3 products may be helpful. Excluding the memory tests, the power-up execution time for a Sun-3 product is about five seconds. The power-up execution time for a Sun-3 product with four megabytes of main memory is approximately thirty seconds. Given a Sun-3 product with 24 megabytes of main memory, the power-up execution time is around 105 seconds. In diagnostic mode, an 8 Mbyte Sun-4 product takes about 4 minutes, while a 128 Mbyte Sun-4 takes about 56 minutes. In normal mode, an 8 Megabyte Sun-4 takes about 45 seconds and a 128 Mbyte Sun-4 takes about 8 minutes.

Test Run Through

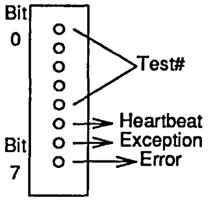
The objective of the power-up test sequence is to determine whether or not the CPU board logic and main memory are functional. Following the successful completion of the power-up tests and subsequent system initialization, an attempt is made to boot the SunOS operating system, an EEPROM-specified program or an operator-specified stand-alone program.

On a Sun-3 or Sun-4, if the diagnostic switch is in the NORM position, the power-up tests execute successfully, and you do *not* terminate the default boot sequence, an attempt is made to down-load the SunOS operating system. Something like the following display then appears on the *workstation* screen:

```
Selftest Completed Successfully.

          Sun Workstation, Model Sun-x/xxx Series
          Sun keyboard type
           ROM Rev __, _  MB memory installed, Serial #____
          Ethernet address __:__:__:__:__:__

Testing __ megabytes of memory ... Completed
```

The Sun386i power-up display is quite different from other Sun products; refer to *Chapter 15* for an example of that display. One requirement of the firmware is to assign a unique test number to most of the power-up tests and display that number in bits *zero* through *four* of the diagnostic LEDs as the test is running. Given that there are fewer test numbers than there are power-up tests, some power-up tests, which check the same part of hardware, will share a test number.

On a Sun-3 or Sun-4 , if one of these power-up tests should fail, bit *seven* of the diagnostic LEDs also lights up. Bit seven indicates that there is a hardware problem. The LED display enables the service person to conclude whether or not there is a problem and determine which type of power-up test is failing.



For most models, LED 5 is the *heartbeat* LED. After the power-up tests have been completed, but prior to booting the SunOS operating system or an EEPROM-specified program, LED 5 blinks on and off to indicate that the CPU is actually executing instructions. LED *six* indicates whether or not the failure is an exception class failure (i.e. unexpected trap or unexpected interrupt). The diagram to the left of this text illustrates LED functions. LED interpretation differs for the Sun386i (refer to *Chapter 15* for more information).

Additionally, if the Diagnostic Switch (on a Sun-3 or Sun-4) is in the DIAG position, status/error messages from the power-up tests are routed to serial ports A and B by way of the MMU (Memory Management Unit) by-pass so that they can be viewed on the screen of a "dumb" terminal. Therefore, even though some power-up tests share the same test number, the output on the terminal's screen tells you exactly which test is failing.

In order to view the messages, the terminal characteristics must be set up as follows: The baud rate for serial port A must be *9600* while the baud rate for serial port B must be *1200*. There must be *eight* data bits and *one* stop bit. Finally, parity must be be turned *off* (no parity).

Upon detection of a failure, the unsuccessful test will enter an infinite "scope loop", continuously re-executing the failing test and thus enhancing the service person's ability to study the problem with test equipment.

When the Diagnostic Switch is in the DIAG position, there are six commands that provide limited interaction with the power-up tests for Sun-3 or Sun-4 systems.

These commands will only be active prior to the display of the `Testing _ megabytes of memory...Completed` message. Here are the commands that control the Sun-3 and Sun-4 power-up self-tests:

1. Press the **`b`** (a mnemonic for *burn-in*) key, prior to the display of the `...Completed` message, to execute the power-up test sequence indefinitely. This option is useful during the manufacturing burn-in stage.
2. Press the **`s`** key prior to the display of the "completed" message to immediately *re-start* the power-up test sequence.
3. If one of the power-up tests fails, it will continue to re-execute forever unless interrupted. Press the **space bar** to terminate the failed test and execute the next power-up test.
4. By default, an unsuccessful power-up test prints one error message before entering an infinite scope loop. Once inside of the scope loop, the failing test will *not* print any more messages. If, however, you would like to see test messages while in the scope loop, press the **`p`** key. Then, to turn the messages back off, press the **`p`** key a second time. In other words, the `p` command acts like a toggle switch to turn message mode on or off.
5. [Esc] skips the remaining power-up tests on a Sun-4 system.
6. On a Sun-4, the **`l`** key causes the currently executing test to run forever.

On a Sun-3 or Sun-4, if the content of location 0x17 of the EEPROM is equal to "0x12" (an arbitrarily chosen value), there is another way to re-start the power-up test sequence. Following the display of the banner

```
Sun Workstation, Model Sun-xxxxx Series,
Type-3 Keyboard...........
```

you may press the User Reset Switch on the CPU Board edge to re-start the power-up test sequence. Pressing the User Reset Switch generates a Watchdog Reset. However, rather than handling the Watchdog Reset in the normal manner, the power-up test sequence is initiated, assuming that location 0x17 of the EEPROM contains "0x12". The power-up tests can be re-started in this manner independent of the position of the Diagnostic Switch.

If the Diagnostic Switch is in the DIAG position, the name of each power-up test will appear on the screens of the *terminals* attached to serial ports A and B. Notice that you will have a *ten* second opportunity to enter the more comprehensive test system by pressing *any* key on a terminal attached to a serial port. The chapters titled *Sun-x Extended Test Sequence (or System)* details these additional tests.

If the Diagnostic Switch is in the DIAG position, the power-up tests executed successfully, and you did *not* enter a character on either of the "dumb terminal" keyboards within the initial ten second period, the following display appears on the *workstation* screen. Again, note that you have *another ten* second opportunity to invoke the additional non-power-up tests by pressing any key on the Sun keyboard. If you do not respond, an attempt is made to boot a program specified in the Diagnostic Boot Path area of the EEPROM. If this attempt fails, the PROM monitor program is invoked.

```
Selftest Completed Successfully.

              Sun Workstation, Model Sun-xxxxx Series
              Type-x keyboard
              ROM Rev __, _  MB memory installed, Serial #____
              Ethernet address __:__:__:__:__:__

Testing __ megabytes of memory ... Completed

Type any character within ten seconds to enter Extended Test System.
```

2.2. Booting the SunOS Operating System

Following the initialization of the Sun-3 or Sun-4 work station, the default boot sequence is executed. There are two issues that must be taken into consideration here. One has to do with *what is to be down-loaded* while the other has to do with *where it is to be loaded from.*

Assuming *no* operator intervention, the position of the Diagnostic Switch will determine *what* is to be booted. If the Diagnostic Switch is in the NORM position, the SunOS operating system is booted. Otherwise, if the Diagnostic Switch is in the DIAG position, the EEPROM-specified program is booted. Be aware that the monitor is invoked if no EEPROM-specified program is available.

If you interrupt the automatic (default) boot sequence, the monitor is invoked. At that point, you may specify *what* is to be booted and *where* it is to be booted from. See command `b` (boot) in the section titled *Basic Monitor Commands* for a description of how to boot user-specified programs from user-specified devices.

If you are interacting with the system through a Sun keyboard, you may interrupt the default boot sequence by typing *L1-a*. That is, hold down the L1 key while pressing the a key. On the other hand, if you are interacting with the system through a "dumb" terminal, press the Break key to terminate the automatic boot sequence.

CAUTION **If the system has a disk and the operating system is running, aborting the auto-boot could damage your file-systems. If the operating system is running, use the procedures described in the *Interrupting the Boot Sequence* subsection.**

The firmware also determines from which device the program is to be loaded. If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is equal to "0x12" (an arbitrarily chosen value), the firmware attempts to boot the SunOS operating system from the UNIX† Boot Path area of the EEPROM, beginning at location 0x19. If the Boot Path, which is supposed to begin at location 0x19, is missing or contains an error, the monitor program is invoked.

---

† UNIX is a registered trademark of AT&T.

If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is *not* equal to "0x12", the firmware will attempt to boot the SunOS operating system using the following boot device polling sequence:

- Xylogics Disk Controller (when applicable)
- SCSI Disk Controller
- Ethernet

On the other hand, if the Diagnostic Switch is in the DIAG position, the firmware will assume that both the path name of the file containing the program to be loaded and the boot device are specified in the UNIX Boot Path area of the EEPROM, beginning at EEPROM location 0x22. If either the file name or the boot device is not present or is in error, the PROM monitor is invoked.

Interrupting the Boot Sequence

It is best not to interrupt the boot sequence; doing so may damage your file systems. Once the operating system is running and you want to access the monitor program, become super-user and type in:

```
example#sync
example#sync
example#/etc/halt
```

If you have no other choice, you may interrupt the boot sequence on the console device — using the `L1-a` sequence on the Sun keyboard, or Break from a terminal. Once you are in the monitor mode (symbolized by the > prompt), you should enter the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press L1-a or Break again when the message above finishes. Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

2.3. Booting From the Monitor

To boot from the PROM monitor, execute the `b` command. Although this command varies with the system architecture, it basically works in the following way:

```
> b boot_device(c,u,p)boot_path
```

boot_device is a two letter designator for any boot device available on that architecture.

To view the boot devices for your system, enter **b**? at the monitor prompt. Boot devices may include:

Table 2-1 *Boot Devices*

Designator	*Boot Device*
`ae`	AMD Ethernet
`ie`	Intel Ethernet
`mt`	Tape Master 9-track tape
`sd`	SCSI disk
`st`	SCSI tape
`xd`	7053 Xylogics disk
`xt`	Xylogics 1/4" tape
`xy`	Xylogics 440/450 disk

c,*u*, and *p* indicate the controller, unit and partition to use when booting. You must enter the parentheses and commas as shown in the example above. If you enter only

```
(,,)
```

the default values will be used.

c is the controller number of the device (a disk controller, for example). if there is only one controller, enter 0.

u is the unit number (if more than one storage unit (disk drive, tape drive, etc.) is driven by the given controller. Enter 0 if there is only one drive for the selected controller.

p is the partition number, which is used to subdivide the storage unit (boot device). If the storage unit has no partitions (a tape for example) enter 0 here.

The *boot_path* gives the path to the file you want to boot, followed by one of seven arguments that is to be passed to the booted program.

NOTE *All of the boot parameters have default settings. The ultimate default is* `b`, *which means boot the default file from the default device.*

The description above is general. For details, refer to the *PROM Monitor* chapter for your architecture.

3

Starting the PROM Monitor

Starting the PROM Monitor **21**

3.1. Power is OFF 21
3.2. Halting the SunOS Operating System 22

3

Starting the PROM Monitor

This chapter tells you how to enter the PROM monitor mode. There are a number of ways to do this. Before you do anything, you must discover the current state of the system. The descriptions below list the different possibilities.

3.1. Power is OFF

You can tell if your workstation is powered down very simply. Check the power switch on the pedestal or base as well as on the video monitor. The "OFF" position is labeled "0".

If your system is powered down, turn ON the power and then, IMMEDIATELY FOLLOWING the

```
Testing __ Megabytes of memory ...Completed
```

message, press the L1 and a keys simultaneously. Or, if your console device is a terminal, press the Break key. You should see the monitor prompt: >.

CAUTION **Do not use the L1-a sequence once the operating system has booted. If disks are powered up and the operating system is running when you use the L1-a sequence, damage to your file systems could result.**

If disks were powered up when you aborted the auto-boot, you should do the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press L1-a or Break again after you see the message shown above. Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

3.2. Halting the SunOS Operating System

Your system is running the SunOS operating system if you see a prompt such as

hostname%

or

hostname#

on the screen. (This is only an example; prompts may be customized to fit the user's needs.)

If your system is up and running the SunOS operating system, you must halt the operating system to bring up the PROM monitor program. This can be done a number of ways. The best way is to use the SunOS `halt` command. To run it, do the following:

```
be sure to shut down all your applications first!

example% su
Password: enter password
example#sync
example#sync
example# /etc/halt

Syncing disks... done

>
```

Another, less preferable way of shutting down the system and bringing up the PROM monitor is to abort the SunOS operating system as described under *Power Is OFF* at the beginning of this chapter.

NOTE *Don't do this unless your system is in such condition that you have no other choice.*

Once you are in the monitor mode (symbolized by the > prompt), you should do the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press L1-a again when the message above finishes. Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

4

Sun-2 Self-tests and Initialization

Sun-2 Self-tests and Initialization **25**

4.1. The Power-Up Test Sequence 25
4.2. Host System Initialization 28
4.3. The Boot Sequence 29

4

Sun-2 Self-tests and Initialization

This chapter describes the specifics of the Sun-2 PROM self-test functions, as well as CPU initialization.

4.1. The Power-Up Test Sequence

The objective of the power-up test sequence is to determine whether or not the CPU board logic is functional. From the beginning, the power-up tests make two assumptions about the hardware; that the MC68010 CPU is functional, and that the ability to fetch instructions from the Boot PROM is intact.

Powering up a Sun-2 workstation resets the MC68010 to boot state. As a result, all instruction fetches are forced to the Boot PROM. Execution of the minimum-confidence power-up tests begin immediately. These tests don't employ any system memory until the memory tests have been successfully completed.

If the power-up tests execute successfully and you do *not* terminate the default boot sequence, the PROMs attempt to load the SunOS operating system. Something like the following display appears on the *workstation* screen to indicate this.

```
Selftest Completed Successfully.

        Sun Workstation, Model Sun-2/xxx, Sun-2 keyboard
         ROM Rev __, _  MB memory installed
        Serial #____, Ethernet address __:__:__:__:__:__

Autoboot in progress...
Boot: sd(0,0,0)
Load: sd(0,0,0)
Boot: sd(0,0,0)vmunix
```

The Sun-2 firmware assigns a unique test number to each of the power-up tests. The test number is displayed in bits *zero* through *seven* of the diagnostic LEDs on the CPU board edge, while that test is running. If one of these tests fails, the LEDs will keep displaying the same pattern, enabling the service person to see which power-up test failed.

The unsuccessful test enters an infinite scope loop, continuously re-executing the failing test. In this way, the service person may study the problem with test equipment.

For the sake of completeness, LED 4 is the *heart-beat* LED. After the power-up tests have been completed, but prior to booting the operating system, LED 4 blinks continuously to indicate that the MC68010 Processor is still executing code successfully.

Power-Up Tests

Here are all of the Sun-2 power-up tests. Included is the name of each test, its function, and the LED pattern displayed while the test is running. A detailed description of each test follows the summary below.

LEDs ■ = ON, □ = OFF 7 6 5 4 3 2 1 0	*What the system is doing when these LEDs are cycling.*	*What might be bad if these LEDs stay on*
● ● ● ● ● ● ● ●	Reset sets LEDs to this state	CPU Board/Boot PROM.
○ ○ ○ ● ○ ○ ○ ●	Checking Context Register	CPU Board (MMU).
○ ○ ● ○ ○ ○ ● ○	Segment Map Address Test	CPU Board (MMU).
○ ○ ● ○ ○ ○ ● ●	Segment Map data lines Test	CPU Board (MMU).
○ ○ ● ○ ○ ○ ○ ●	Segment Map constant data Test	CPU Board (MMU).
○ ○ ● ● ○ ○ ● ○	Checking Page Map Address dependency	CPU Board (MMU).
○ ○ ● ● ○ ○ ● ●	Checking Page Map Data lines	CPU Board (MMU).
○ ○ ● ● ○ ○ ○ ●	Checking Page Constant Data lines	CPU Board (MMU).
○ ● ● ● ○ ○ ○ ○	Checks memory size	CPU or Memory board.
○ ● ● ● ○ ○ ○ ●	Constant Data Memory Test	CPU or Memory board.
○ ● ● ● ○ ○ ● ○	Memory Address Dependency Test	CPU or Memory board.
● ● ● ● ○ ○ ● ●	Setting up frame buffer	
● ● ● ● ○ ● ○ ○	Setting up NMI or keyboard	
○ ○ ○ ● ○ ○ ○ ○	Self-Test done, SunOS system in boot-state	CPU Board
○ ○ ○ ○ ○ ○ ○ ○	(LED is blinking)	

Diagnostic Register Test

This test sequentially lights up each of the LEDs. It indirectly tests the processor's ability to fetch instructions from the Boot PROM and transfer data across the data bus.

Context Register Test

This test performs write-read-compare cycles on the Context Register.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
2	0x11	□□□■□□□■

Segment Map Address Test

If this test fails, the Segment map address lines are shorted or miswired.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
3	0x22	□□■□□□■□

Segment Map data lines Test

If this test fails, the Segment map data wires are shorted or bits are bad.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
4	0x23	□□■□□□■■

Segment Map constant data Test

If this test fails, the Segment map data wires are shorted.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
5	0x21	□□■□□□□■

Page Map Address dependency Test

If this test fails, the Page map address lines are miswired or shorted.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
6	0x32	□□■■□□■□

Page Map data lines Test

If this test fails, the Page map data lines are miswired or shorted.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
7	0x33	□□■■□□■■

Page Map Constant Data Test

If this test fails, there is a hardware problem.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
8	0x31	□□■■□□□■

Memory sizing Test

Prior to testing memory, the memory is sized by mapping it in, then trying to write and read back each location in memory.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
9	0x70	□■■■□□□□

Memory Constant Data Test

The memory is tested by write-read-compare steps.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
10	0x71	□■■■□□□■

Memory Address Dependency Test

The memory address lines is checked for miswiring or defective chips.

Test Number	Hexadecimal Value Of LEDs	Visual Representation
11	0x72	□■■■□□■□

4.2. Host System Initialization

After the *error-free* completion of the power-up tests but before the execution of the default boot sequence, the workstation requires initialization. The initialization steps are listed below.

1. The segment map in each context is initialized.
2. All page map entries are initialized.
3. Memory is sized.
    a. The total amount of main memory that is physically present, both working and non-working, is stored in `*romp->v_memorysize`.

   b. Variable `*romp->v_memoryavail` will contain the amount of available memory, excluding nonworking pages and pages reserved for the monitor itself.

4. All of the available main memory is initialized.
5. All available pages of *working* memory are mapped in linear order, starting at location zero.
6. Assuming that they exist, these devices are initialized and mapped:
   a. Keyboard.
   b. Mouse.
   c. Serial Ports A and B.
   d. Video Memory.
   e. TOD Clock.
   f. Ethernet.
7. As part of the initialization, the monitor's interrupt vectors are employed.
8. Entry points to support routines are set up.

4.3. The Boot Sequence

Following the workstation initialization, the default boot sequence is executed. There are two issues that must be taken into consideration here. One has to do with *what is to be loaded* while, the other has to do with *where it is to be loaded from.*

If you interrupt the automatic (default) boot sequence, the monitor is invoked. At that point, you may specify *what* is to be booted and from what device it is to be booted. See command `b` (boot) in the *Sun-2 Monitor Commands* chapter for a description of how to boot user-specified programs from user-specified devices.

Normally, the firmware must determine from what device the program is to be loaded. Sun-2 firmware uses the default boot sequence:

1. Xylogics Disk.
2. SCSI Disk.
3. Ethernet.

If you interrupt the default boot sequence or specify a non-existent boot device or boot path, the monitor program is invoked.

Refer to *Chapter 2* for more information on booting.

5

Sun-2 PROM Monitor Commands

Sun-2 PROM Monitor Commands **33**

5.1. Introduction 33
5.2. Monitor Command Overview 33
5.3. The Sun-2 Monitor Commands 35

5

Sun-2 PROM Monitor Commands

5.1. Introduction

This chapter describes the PROM monitor (sometimes called the "system monitor") commands available on Sun-2 workstations. Taken as a whole, these commands offer a low-level user interface to the Sun hardware. The commands control a variety of options, including booting from an alternate device, changing the console output, reading or altering registers or memory locations, and so on.

The monitor commands are transient (their affects disappear when the power is turned off).

Bringing up the PROM Monitor

Read chapter 3 for instructions on bringing up the PROM monitor.

Old vs New PROMs

Most Sun-2 systems have what we shall call the "old" PROM set. An alphabetic character represents the revision level of these PROMs. The "new" set, included with systems shipped following Software Release 4.0, have numerical revision designations. They include a number of changes to the monitor program, among which are four new commands: The **f** or block writer command, the **h** or help command, the **v** or display memory block, and the **x** or extended tests command. The only other functionality change is that Sun-2 systems with the new PROMs can now boot from tftp servers. The new commands make Sun-2 PROMs more similar to the Sun-3 and Sun-4 PROM family. The new commands are marked as such in the command descriptions.

5.2. Monitor Command Overview

The listing that follows shows the results of typing an `h` (help) at the monitor prompt.

NOTE *This command only works with the "new" Sun-2 PROMs. See the description above.*

This section describes the general characteristics that all of the commands have in common. Detailed descriptions of each command and its arguments are listed in the next section.

Figure 5-1 *Monitor Help Menu (New Sun-2 PROMs Only)*

```
Boot PROM Monitor Commands
-------------------------------------------------------------
a [digit]                          |Open CPU Addr Reg (0-7)
b [dev([cntrl],[unit],[part])]     |Boot a file
c [addr]                           |Continue program at Addr
d [digit]                          |Open CPU Data Reg (0-7)
e [addr]                           |Open Addr as 16 bit word
f beg_addr end_addr pattn [size]|Fill Memory
g [addr]                           |Go to Addr
h                                  |Help Menu
k [number]                         |Reset (0)CPU, (1)MMU, (2)System
l [addr]                           |Open Addr as 32 bit long
m [addr]                           |Open Segment Map
o [addr]                           |Open Addr as 8 bit byte
p [addr]                           |Open Page Map
r                                  |Open CPU Regs (i.e. PC)
s [digit]                          |Set/Query Function Code (0-7)
t [y/n/c]                          |Trace: Yes/No/Continue
u [arg]                            |Select Console Device
v beg_addr end_addr [size]         |Display Memory
w [addr] [string]                  |vector
x                                  |Extended Diag Tests
z [addr]                           |Set Breakpoint
-------------------------------------------------------------
```

Executing a Command

In general, to execute a command, you type in the appropriate command letter, followed by any required command arguments. For example, if you wanted to execute the hypothetical command θ (which we will say needs the arguments "100" and "200"), you would type a line like this:

```
> θ 100 200
```

The command letter can be upper or lower case, and all commands and arguments are separated by white-space (tabs or spaces). Pressing Return executes the command.

Default Values

Many of the monitor commands have built in, or *default* values, which the command uses if you do not supply an argument. The default values vary from command to command. Check the command descriptions for the default values of interest.

Word Sizes

Word sizes referred to in this chapter are defined as follows: A **byte** is eight bits long; a **word** is 16 bits long; a **long word** is 32 bits long.

5.3. The Sun-2 Monitor Commands

This section provides more detailed information on the commands listed in the previous section. Both the commands and their arguments are described here. Note that any changes made with these commands are valid only until the system is powered down.

Displaying and Modifying

A number of the commands listed below can be used to display and/or modify the system's memory and registers. Regardless of the type of memory (RAM, PROM or register), these commands have the same command syntax. This section describes the monitor command syntax for memory modification. The example here references long words (32 bits) of memory, but the syntax is the same for bytes (8 bits) and words (16 bits).

Memory modification commands accept two types of arguments. The `address` argument specifies the initial memory location that is to be displayed and/or modified.

The second argument determines whether the current content of a memory location is to be displayed and/or modified. Entering *only* the address of a memory location after the command *displays* the content of that address:

```
>command FFE80000 Return
FFE80000: 12345678 ?
```

At this point, you can respond in one of *three* different ways.

1. Simply pressing the Return key displays the contents of the next memory location (in this case, 0xFFE80004) as shown below.

```
>commandFFE80000 Return

FFE80000: 12345678 ?  Return

FFE80004: 00000001 ?
```

Successively pressing the Return key displays the contents of successive memory locations. Assuming that you pressed Return four times, the contents of memory locations 0xFFE80004, 0xFFE80008, FFE8000C and 0xFFE80010 would be displayed.

To exit the command, enter **any** *non-hexadecimal* character (i.e. **q** for quit) before pressing Return Note that you will now return to the monitor's basic command level, with the > prompt:

```
>command FFE80000 Return

FFE80000: 12345678 ? q Return

>
```

2. If you only want to modify one location, use a command such as:

```
>command FFE80000 12345678
```

3. The *third* response to the display of memory location contents is to *modify* those contents. You enter the *new* hexadecimal value immediately following the question mark `?`, BEFORE pressing Return. The following display demonstrates how the value of memory location 0xFFE80000 can be changed from "12345678" to "00ABCDEF". Note: following the assignment of the new value to memory location 0xFFE80000, the new value will *not* be displayed; instead, the contents of the next memory location are shown. You will *not* be returned to the monitor's basic command level.

```
>command FFE80000 Return

FFE80000: 12345678 ? 00ABCDEF Return

FFE80004: 00000001 ?
```

Entering a memory location's virtual address followed *only* by a non-hexadecimal character before pressing the Return key causes the monitor to *display* the contents of that location then exit to the monitor command level.

```
>command FFE80004 q Return
FFE80004: 00000001
>
```

Entering a non-hexadecimal character *and* a hexadecimal value after an address causes the monitor to display the original value at that virtual address, assign a new value, then display that value. For instance, assume that the original content at address 0xFFE80010 is "00000005". The command

```
>command FFE80010 ? 55555550 Return
```

displays the original value "00000005" then assigns the value "55555550" to address FFE80010 before returning to the monitor prompt.

```
>command FFE80010 ? 55555550 Return
FFE80010: 00000005 -> 55555550
>
```

You may also enter multiple display and/or modify commands for multiple memory locations on the same command line. If you enter this command line:

```
command FFE80000 ? 00000000 ? ? 22222220 33333330 q Return
```

You will see this on the screen:

```
FFE80000: 12345678 -> 00000000
FFE80004: 00000001
FFE80008: 00000002 -> 22222220
FFE8000C -> 33333330
FFE80010: 00000004
>
```

The first part of the command line, ***command*** **`FFE80000 ? 00000000`** displays the original contents of location FFE80000 before assigning the new value "00000000" to it.

The next question mark directs the monitor to display the contents of FFE80004. The next part of the command line, **`? 22222220`**, tells the monitor to display the original contents of FFE80008, before assigning the new value "22222220" to it. The **`33333330`** tells the monitor to assign the value "33333330" to memory location FFE8000C. Finally, the **`q`** causes the monitor to exit to the command level after the contents of FFE80010 are displayed.

Conventions

The paragraphs below describe each of the monitor commands in detail. Each paragraph starts with a line describing the command syntax. If a command has more than one distinct format, each one is shown on a separate line. Letters in **`bold typewriter`** font are typed as shown. Plain `typewriter` font represents what you should see on the screen. Words in *`typewriter italic`* show the type of information you are to enter. *italic* or **boldfaced** Roman font is also used for emphasis within the text. Optional arguments and default values are listed in the descriptions.

Special Commands for New Sun-2 PROMs

If your Sun-2 is upgraded with the new PROMs, these special commands will be available.

Address Increment/Decrement Command

By preceding the command with a **+** or **–**, you can cause the address display to increment or decrement to the next location.

While traversing a range of addresses, type a + or – to change direction after the program displays the content and waits for input.

For example:

```
>l 0 [Return]

00000000 00000000 ?  [Return]

00000004 00000001 ?  [Return]

00000008 00000002 ? - [Return]

00000004 00000001 ?  [Return] (contents of previous location are displayed)

00000000 00000000 ? + [Return] (after decrement, you enter a plus sign)

00000004 00000001 ?  [Return] (you increment again)

00000008 00000002 ?  [Return] (you increment again)
```

The `^T` Command

Entering the `^` character and then the **`t`** key, followed by a virtual address, displays the physical address to which that address is mapped, along with a detailed description of all the bits in the page table entry, the segment and page RAM addresses, and what space they are in.

For example, entering

```
>^t 1000 [Return]
```

results in this display:

```
Virtual Addr 1000 is mapped to Physical Addr 1000
Context = 0x0, Seg Map = 0x0, Page Map = 0xC0000000.
Page 0 has these attributes:

        Valid bit  = 0
        Permission = 1
        No Cache   = 0
        Type       = 0
        Access bit = 0
        Modify bit = 0
```

Entering **`^t`** with no arguments provides information with reference to virtual address 0x00.

The `^I` Command

Entering the `^` character and then the **`i`** key, followed with a command, displays compilation information for the system firmware. It includes the date, host name and build directory path. For example:

```
Compiled at 6/7/87 on hostname in /directory_name
```

The `^C` Command

`^C source destination n`

Entering the `^` character and then the **c** key, followed with the parameters shown, causes a block of *n* length to be copied from `source` to `destination` address, byte by byte.

Regular Sun-2 Monitor Commands

Monitor **a** Command

a `register_number action`

The **a** command provides access to the CPU's address registers. `register_number` may be a value from 0 to 7 inclusive. The default value is 0. Register_number 7 accesses A7, the system stack pointer. To see the *user* stack pointer, use the **r** command.

It is important to note that it is *not* possible to display the state of the processor at all times. The processor state may only be observed after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`).
- You have manually "broken" into the monitor mode (by typing the **L1 a** sequence on a Sun keyboard or [Break] on a dumb terminal's keyboard).

Read the section entitled *Displaying and Modifying Memory* for details on how to use this command. Replace the word `command` in the description with the letter **a**.

Monitor **b** Command

b ? or **b** `boot_device (controller,unit,partition) path`

The *boot* command loads and executes the SunOS operating system or a user-specified program. It boots from a default device or the boot device specified in the command argument. A *boot device* is a secondary storage device (disk, Ethernet, or tape) that contains the program to be loaded and executed.

If a boot device is not specified, the program polls to see what devices are available, in this order:

Xylogics Disk
SCSI Disk
Ethernet

If the boot attempt fails, you are returned to the monitor's command level.

In order to boot from a specific device, the **b** command must be followed with a boot device abbreviation, such as those shown below. Enter **b ?** to view the boot device identifier arguments that your PROM monitor will accept.

Some possible boot devices are:

- xy — Xylogics 450/451 disk
- sd — SCSI disk
- ie — Intel Ethernet
- ec — 3COM Ethernet
- st — SCSI 1/4-inch tape drive
- mt — Tapemaster 1/2-inch tape drive

The `sd, st, and xy` arguments are followed with parentheses that optionally enclose the controller number, unit number, and partition or file number:

```
>b sd(0,0,0)
```

The default values 0,0,0 are used if you do not enter values inside the parentheses.

Up to seven optional arguments (which are passed to the boot program) may follow the path argument. If you do not want the system to be reset prior to booting, you must insert **!** before the device identifier argument.

The boot command given below would boot the video diagnostic from the `/stand` directory over the Ethernet. Because the **!** argument does *not* precede the device identifier argument, the system is reset before the booting process begins. Note that one optional argument **-t** is passed on to program the `video.diag` program.

```
> b ie(0,0,0)/stand/video.diag -t
```

Monitor **c** Command

c *virtual_address*

The *continue* command resumes execution of an interrupted program. You can specify the virtual address of the instruction to be executed when the program restarts.

By default, the program resumes execution at the address pointed to by the Program Counter.

Monitor **d** Command

d *`register_number action`*

The *Data register* command provides access to the CPU's data registers. `register_number` can be a value from 0 to 7 inclusive. The default value is 0.

Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **d**.

Monitor **e** Command

e *`virtual_address`*

The *display/modify memory* command displays and/or modifies the content of one or more virtual addresses in *word* mode (i.e. each virtual address will be treated as a 16-bit unit). That is, the content of one or more words can be *displayed*, *modified* or, both *displayed* and *modified.*

See the previous section *Displaying and Modifying Memory* for a description of this command's syntax. Use the letter **e** in place of the word *`command`* in the description.

Monitor **f** Command

f *`start_virtual_address end_virtual_address pattern size`*

The *block write* command (in "new" Sun-2 PROM s only) writes the `pattern` you enter into each byte, word or long word in the range of virtual addresses you specify with the `start_virtual_address` and `end_virtual_address` arguments . By default, if the final, memory-cell-size argument is not present, bytes are written. Arguments `start_virtual_address end_virtual_address` and `pattern` are required while `size` is optional. The possible values for *size* are `b` (8-bit byte), `w` (16-bit word) or `l` (32-bit long word).

Monitor **g** Command

g *virtual_address argument*

When you use the *goto* command, you may go to (jump to) a *pre-determined*, *user-specified* or *default* routine. The argument `virtual_address` is used to indicate the virtual address of a *user-specified* routine, and the optional `argument` is passed along to that routine. If you do not supply the `virtual_address` argument, the value in the Program Counter is used.

In order to set up a *pre-determined* routine, a user program has to set variable `*romp->v_vector_cmd` equal to the virtual address of the routine. Variable `*romp->v_vector_cmd` must be set prior to executing the `g` command. Pre-determined routines may or may not return to the monitor.

The *default* routine, defined by the monitor, will simply print the user-specified `vector` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument `vector` as a hexadecimal number while, format "%d" prints argument `vector` as a decimal number.

Monitor **h** Command

h

The *help* command (in "new" Sun-2 PROMs only) invokes the Help System for the basic monitor commands. Each of the basic monitor commands and its argument(s) are described in a help screen. No arguments are accepted by the `h` command. The initial help menu of the Help System is shown at the beginning of this chapter.

Monitor **k** Command

k *`0, 1, 2 or b`*

The *reset* command performs various levels of system resets. It can also be used to display the system's banner. This command accepts one optional argument. Entering **k 0** resets the VME-bus, interrupt register and video monitor (the Low-Level Reset). Entering **k 1** invokes a Software Reset. Entering **k 2** invokes a Power-On Reset (Hard Reset). Finally, entering **k b** displays the banner on the video monitor or terminal. The default value of the argument is `0`.

Monitor `l` Command

`l` *`virtual_address`*

The *modify long words of memory* command displays and/or modifies the contents of one or more virtual addresses in *long* word mode. Each virtual address is treated as a 32-bit unit (long word). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`l`**.

Monitor `m` Command

`m` *`virtual_address`*

The *modify segment table* command displays and/or modifies the contents of one or more of the Segment Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`m`**.

Monitor `o` Command

`o` *`virtual_address`*

The *modify bytes of memory* command displays and/or modifies the content of one or more virtual addresses in *byte* mode. Each virtual address is treated as an 8-bit unit (byte). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`o`**.

Monitor `p` Command

`p` *`virtual_address`*

The *modify page table* command displays and/or modifies the contents of one or more of the Page Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`p`**.

Monitor `r` Command

`r` The *miscellaneous register* command displays and/or modifies the contents of the CPU's miscellaneous registers. These registers are listed in the table below:

Table 5-1 *Miscellaneous Registers for the MC68010*

Symbol	*Name*
`SS`	Supervisor Stack Pointer
`US`	User Stack Pointer
`SF`	Source Function Code
`DF`	Destination Function Code
`VB`	Vector Base
`SC`	Supervisor Context
`UC`	User Context
`SR`	Status Register
`PC`	Program Counter

Registers modified by this command (except for the System and User Context registers) don't take effect until the next **`c`** command is executed.

It is important to note that it is *not* always possible to display the registers. They are only observable after

1. An unexpected trap
2. A user program has "dropped" into the monitor (by calling monitor function `abortent`) or
3. You have manually "broken" into the monitor (by typing `L1-a` on a Sun keyboard or `Break` on a "dumb" terminal's keyboard).

Read the section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`r`**.

Monitor **s** Command

`s` *`number`*

The `set/query function code` command sets or displays the address space to be used by subsequent memory access commands. The value of *`number`* determines which Function Code is accessed by the memory display and/or modify commands. If given no argument, the current setting is printed. The acceptable *`number`* argument values are shown in the following table.

Table 5-2 *Function Code Values*

Value	*Address Space*
0	Reserved (don't use)
1	User Data
2	User Program
3	MMU Space
4	Reserved (don't use)
5	Supervisor Data
6	Supervisor Program
7	Reserved (don't use)

Monitor **u** Command

`u` *`port options baud_rate`*
or

`u` *`echo`*
or

`u` `u`*`virtual_address`*

The *input/output* command configures the input and output devices and their characteristics or displays the current input and output device set-up.

The **u** command requires arguments to specify from which device(s) you want the system to expect input or which device(s) will display output.

If you do not enter an argument after the `u` command, the program will display the current settings. If no serial port is specified when changing baud rates, the baud rate of the current input device is changed.

The default console output device is the Sun monitor. If the workstation has a color monitor and a color board is unavailable, the program will look for a monochrome monitor as an output device.

You may alter the existing I/O settings while you are in the monitor mode, using the commands listed below; however, the default settings will be reinstated when the system is power cycled.

u Command Arguments:

The `port` argument can be one of the following:

Table 5-3 *Port Arguments*

Port Argument	*Device*
`a`	Serial Port A
`b`	Serial Port B
`k`	Keyboard
`s`	Screen

The `options` arguments are:

Table 5-4 *Option Arguments*

Option Argument	*Meaning*
`i` `o`	input output
`u`	UART
`e` `ne`	input echoed to output input *not* echoed to output
`r`	reset specified serial port

The `baud_rate` argument specifies baud rate of the serial port being discussed. The `virtual_address` argument specifies the virtual address of the UART (Universal Asynchronous Receiver/Transmitter).

Following are examples of port and options arguments:

- Enter **u aio** or **u bio** to select serial port A or B as the input and output device.
- Enter **u ai** or **u bi** to select serial port A or B for input only.
- Enter **u ao** or **u bo** to select serial port A or B for output only.
- Enter **u k** to select the keyboard for input.
- Enter **u ki** to select the keyboard for input.
- Enter **u s** to select the screen for output.
- Enter **u so** to select the screen for output.
- Enter **u ks, sk** to select the keyboard for input and the screen for output.

- Enter **`u a`***baud rate* or **`u b`***baud rate* to set the serial port speed.
- Enter **`u e`** to cause the output to echo the input.
- Enter **`u ne`** to cause the output **not** to echo the input.
- Enter **`u`** *address* to set the serial port virtual address.

A *previously mapped* UART can also be used for input and/or output. Command **`u u`**`virtual_address`, where `virtual_address` is the virtual address of a previously mapped UART, changes the virtual address of the UART. Do not leave a space between the second **`u`** and the address.

If the **`u`** command is not followed by an argument, the current settings are displayed. The current input device, output device, baud rate for serial ports A and B, UART's virtual address and input to output echo indicator are shown.

Monitor **v** Command

`v` *`start_virtual_address end_virtual_address size`*

The `display memory block` command (for "new" Sun-2 PROM s only) displays the contents of each byte, word or long word in the range of virtual addresses specified by *`start_virtual_address`* and *`end_virtual_address`*. The *`word_size`* argument is optional; the default is to display memory in byte format. In this format, sixteen consecutive bytes are displayed on each line. In word format, eight consecutive words appear on each line.

If long word format is enabled, four consecutive long words appear on each line. To the far right of each line, the character corresponding to each byte is also shown. All bytes that contain a non-ASCII code are shown as a period ("."). Legal values for *`size`* are **`b`** (8-bit byte), **`w`** (16-bit word), or **`l`** (32-bit long word).

Monitor **w** Command

`w` *`virtual_address argument`*

The `set execution vector` command allows you to vector to a *pre-determined* or *default* routine. Optional arguments *`virtual_address`* and *`argument`* are passed along to the to-be-executed routine.

In order to set up a *pre-determined* routine, a user program sets the variable `*romp->v_vector_cmd` equal to the virtual address of the *pre-determined* routine. Variable `*romp->v_vector_cmd` must be set prior to executing the **w** command. *Pre-determined* routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified *`virtual_address`* argument according to the user-specified format (given in *`argument`*) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument *`virtual_address`* as a hexadecimal number; format "%d" prints it as a decimal number.

Monitor **x** Command

`x`

In the new Sun-2 PROM s, the `extended test system` command invokes the Extended Test System. See *Chapter 6* for details.

6

Sun-2 Extended Test Sequence

Sun-2 Extended Test Sequence **49**

6.1. Introduction 49
6.2. Extended Test Menu Structure 50
6.3. Sun-2 Extended Test Sub-Menus 52

6

Sun-2 Extended Test Sequence

6.1. Introduction

This chapter shows the menu hierarchy for the Sun-2 Extended Test Sequence. Use this sequence to perform detailed tests on the hardware.

NOTE *This sequence is only included in the* **new** Sun-2 PROMs. See the *Overview* chapter for details.

The Extended Menu Tests are a suite of test routines that provide detailed testing of the system hardware. They provide an additional level of testing beyond the power-up self-tests. You can invoke them with the monitor **x** command if you have the *new* Sun-2 boot PROM.

You are in the monitor when you see the > prompt on the screen.

6.2. Extended Test Menu Structure

The user interface of the monitor's extended test menu structure consists of a Main Menu and multiple sub-menus.

Main Menu

Two Main Menus, which differ slightly according to workstation type, are displayed in the examples below. The options are described after the menus.

Extended Test Menu for Sun-2/50, Sun-2/130, and Sun-2/160

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

kb  - Keyboard Input Test
me  - Memory Test
vi  - Video Test
mk  - Mouse/Keyboard Test
rs  - Serial Ports Test
ie  - Intel Ethernet Test
mt  - Tapemaster Bootpath Test
sd  - SCSI Disk Bootpath Test
st  - SCSI Tape Bootpath Test
xt  - Xylogics Tape Bootpath Test
xy  - Xylogics Disk Bootpath Test

Cmd=>
```

Extended Test Menu for Sun-2/120 and Sun-2/170.

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

kb  - Keyboard Input Test
me  - Memory Test
vi  - Video Test
mk  - Mouse/Keyboard Test
rs  - Serial Ports Test
ar  - Archive Tape Bootpath Test
mt  - Tapemaster Bootpath Test
sd  - SCSI Disk Bootpath Test
st  - SCSI Tape Bootpath Test
xt  - Xylogics Tape Bootpath Test
xy  - Xylogics Disk Bootpath Test

Cmd=>
```

The options available from the two Sun-2 Main Menus are presented here in alphabetical order for your convenience.

`ar`

The *Archive Bootpath Test* command tests the bootpath to the 1/4" Archive tape drive. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code. boot

`ie`

The *Intel Ethernet Test* checks the Intel 82586 Ethernet Coprocessor chip, the Intel 82501 Serial Interface Unit chip, and the connection to the Ethernet network. This command brings up the Ethernet Menu with different test options.

`kb`

The *Keyboard Test* command determines whether or not keyboard characters are correctly transmitted to the CPU. After invoking this test, The ASCII code corresponding to a character and the character itself appear on the screen as you press keys on the keyboard. To exit the test, type an [ESC] character.

`me`

The *Memory Test* command invokes the Memory Menu. This menu contains all of the memory tests .

`mk`

The *Keyboard and Mouse Menu Tests* command invokes the Keyboard and Mouse Menu. This menu contains the keyboard and mouse tests

`mt`

The *Tapemaster Bootpath Test* command tests the bootpath to the 1/2-inch Tapemaster tape drive. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

`rs`

The *Serial Ports Menu Tests* command invokes the Serial Ports Menu. This menu contains all of the serial port tests.

`sd`

The *SCSI Disk Bootpath Test* command tests the bootpath to the SCSI hard disk. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

`st`

The *SCSI Tape Bootpath Test* command tests the bootpath to the SCSI tape drive. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

`vi`

The *Video Test* command invokes the Video Menu. This menu contains all of the video tests.

`xt`

The *Xylogics Tape Bootpath Test* command tests the bootpath to the Xylogics 1/2-inch tape drive. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

`xy`

The *Xylogics Disk Bootpath Test* command tests the bootpath to the Xylogics hard disk. It makes sure the workstation can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

6.3. Sun-2 Extended Test Sub-Menus

The following pages describe the various sub-menus available from the main extended test menu. They are in alphabetical order for your convenience.

Intel Ethernet Menu

This menu comes up only for Sun-2 systems that use the VME bus, when **`ie`** is chosen from the Extended Test Main Menu. The Intel Ethernet Menu contains three *local* options. The three Ethernet tests check the functionality of the Intel 82501 Serial Interface Unit Chip, the Intel 82586 Ethernet LAN Co-processor Chip and the connection to the Ethernet network. These options are explained below.

```
Intel Ethernet Tests:    (Enter 'q' to return to Test Menu)

Cmd - Test

l - Local Loopback Test
e - Encoder Loopback Test
x - External Loopback Test

Cmd=>
```

The Intel Ethernet Test sub-menu choices are described below.

`l`

The *Local Loopback Test* command tests the Intel 82586 Ethernet LAN co-processor chip. It runs the internal tests built into the chip. Prior to the test, the Intel 82586 Ethernet LAN co-processor chip disconnects itself from the Intel 82501 Serial Interface Unit Chip.

`e`

The *Encoder Loopback* command runs an internal loop back test of the Intel 82501 Serial Interface Unit Chip. The transmitter line, receiver line, noise filters and Manchester encoding/decoding logic are tested. Before executing this test, the transmitter and receiver lines of the chip are connected together.

`x`

The *External Loopback* command runs the external loop back test. This test checks the Intel 82586 Ethernet LAN Co-processor Chip, the Intel 82501 Serial Interface Unit Chip and the Ethernet transceiver and receiver lines. The test sends data out onto the Ethernet, then receives it back and compares the transmitted and received data. Before running this test, attach an Ethernet transceiver cable to the CPU board and to a properly terminated transceiver.

Memory Test Menu

When you choose the **me** option from the main Sun-2 Extended Test Menu, the Memory Menu comes up:

```
Memory Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low addr . 0x2000 [hi addr <0xXXXXXX] [hex pattern]

Cmd  -  Test

  a  -  Address Test
  c  -  Wr/Rd/Compare Test
  s  -  Scan Memory Test
  w  -  Write Pattern Test

Cmd=>
```

The Memory sub-menu test choices are described in the paragraphs that follow.

a`low high`

The **a** command executes the address test on a range of memory. Specifically, write-read-compare cycles is performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts a maximum of two arguments:

The *low* argument specifies the first address to test. By default, the value of *low* is `0x2000`. The *low* value should be a hexadecimal (base 16) number.

The *high* argument indicates the final address to test. The default high address is the highest memory address available. The *high* value should be a hexadecimal (base 16) number.

c *low high pattern*

The **c** command performs write-read-compare cycles on a range of addresses with a specified pattern. The test accepts three optional arguments and uses long-words only.

The *low* argument specifies the first address to test. By default, the value of *low* is `0x2000`. The *low* value should be a hexadecimal (base 16) number.

CAUTION **Addresses 0x0 to 0x2000 are reserved for use by the boot PROM. If this test overwrites those locations, the system may crash.**

The *high* argument represents the final address to test. The default high address is the highest memory address available. The *high* value should be a hexadecimal (base 16) number.

The *pattern* argument names the pattern expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *pattern* is `0xaaaaaaaa`. The *pattern* value should be a hexadecimal (base 16) number.

s *`low high pattern`*

The **s** command reads the specified range of addresses and compares the values to the value of *`pattern`*.

The *`low`* argument specifies the first address to test. By default, the value of *`low`* is `0x2000`. The *`low`* value should be a hexadecimal (base 16) number.

The *`high`* argument represents the final address to test. The default high address is the highest memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names the pattern expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

w *`low high pattern`*

The *w* command writes a pattern to a range of addresses. The test accepts three optional arguments.

The *`low`* argument specifies the first address to test. By default, the value of *`low`* is `0x2000`. The *`low`* value should be a hexadecimal (base 16) number.

The *`high`* argument represents the final address to test. The default high address is the highest memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

Mouse/Keyboard Menu

When you enter **mk** from the Sun-2 Extended Test Main Menu, the The Mouse/Keyboard Ports Tests menu is offered:

```
Mouse/Keyboard Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(M or K)] [Baud rate (decimal #1)] [hex byte
pattern]

Cmd  - Test

w  -  Wr/Rd SCC Reg 12 Test
x  -  Xmit Char Test
i  -  Internal Loopback test
e  -  External Loopback Test

Cmd=>
```

The following text describes the Mouse/Keyboard Ports Test menu choices.

w *channel baud pattern*

The *Write and Read SCC Test* command performs write-read-compare cycles to register 12 of the port under test. This command accepts as many three arguments. The *channel* argument determines which port is tested. Legal values for *channel* are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *baud* argument sets the baud rate at which the test is executed. The default baud rate is 1200. The *baud* argument should be a decimal (base 10) number.

The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *pattern* should be a hexadecimal (base 16) number.

x *channel baud pattern*

The *Xmit* command writes patterns to the port under test. This command accepts the same arguments as the **w** command.

`i` *channel baud pattern*

The *Internal* command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port are connected internally prior to the test.

This command accepts the same arguments described for the **w** command.

e *channel baud pattern*

The *External* command executes an external loop back test on a user-specified port. Write-read-compare cycles are performed on the port under test. To run this test, the within-port external loop back cable must be installed (see *Chapter 1*). This command accepts the same arguments as the **w** command.

Serial Ports Menu

When you choose **`rs`** from the Sun-2 Extended Test Main Menu, the Serial Ports Menu comes up:

```
Serial Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd:  Cmd [port(A or B)]  Baud rate(decimal #)] [hex byte pattern]


Cmd  - Test


w  -  Wr/Rd SCC Reg 12 Test
x  -  Xmit Char Test
i  -  Internal Loopback test
e  -  External Loopback Test


Cmd=>
```

The Serial Ports sub-menu choices are discussed in the paragraphs that follow.

`w` *`channel baud pattern`*

The `Wr/Rd SCC Reg` command performs write-read-compare cycles to register 12 of the port under test. This command accepts three arguments.

The *`channel`* argument determines which port is tested. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *`baud`* argument sets the baud rate at which the test is executed. The default baud rate is 1200. The *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

`x` *`channel baud pattern`*

The `Xmit` command writes patterns to the port under test. This command accepts the same arguments described for the **w** Serial Ports menu choice.

`i` *`channel baud pattern`*

The `Internal` command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port is connected internally prior to the test.

The arguments for this command are the same as those described for the **w** Serial Ports Menu choice.

e *channel baud pattern*

The `External` command executes an external loop back test on a the port you specify. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (see *Chapter 1*).

The arguments for this command are the same as those described for the **w** Serial Ports Menu choice.

Video Test Menu

When you choose **vi** from the Sun-2 Extended Test Main Menu, the Video Tests menu comes up:

```
Video Tests:  (Enter 'q' to return to Test Menu)

Enter Cmd [low addr > 0xXXXXXX [hi addr <0xXXXXXX] [hex pattern]


a  -  Address Test
c  -  Wr/Rd/Compare Test
s  -  Scan Memory Test
w  -  Write Pattern Test

Cmd=>
```

a*low high*

The **a** command executes the address test on a specified range of frame buffer memory addresses. Specifically, write-read-compare cycles are performed on long words. The data that is written to each memory "cell" is its own address. This command accepts two arguments.

The *low* argument specifies the first address to test. By default, the value of *low* is `0x2000`. The *low* value should be a hexadecimal (base 16) number.

The *high* argument represents the final address to test. The default high address is the highest frame buffer memory address available. The *high* value should be a hexadecimal (base 16) number.

c *low high pattern*

The **c** command performs write-read-compare cycles on a range of addresses with a specified pattern. The test accepts three optional arguments and uses long-words only.

The `low` argument specifies the first address to test. By default, the value of `low` is `0x2000`. The `low` value should be a hexadecimal (base 16) number.

The `high` argument represents the final address to test. The default high address is the highest frame buffer memory address available. The `high` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

s*low high pattern*

The **s** command reads the specified range of addresses and compares the values to the value of `pattern`. The test accepts three optional arguments.

The `low` argument specifies the first address to test. By default, the value of `low` is `0x2000`. The `low` value should be a hexadecimal (base 16) number.

The `high` argument represents the final address to test. The default high address is the highest frame buffer memory address available. The `high` value should be a hexadecimal (base 16) number.

The `pattern` argument names a pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w *low high pattern*

The **w** command writes a pattern to a range of addresses. The test accepts the same options as the **s** command, described above.

Options Menu

The Options Menu is offered when you select any Extended Menu Test. You may use the Options Menu to specify the sequence of testing. The default option is to execute the test once, then return to the current menu.

```
Test Options: (Enter 'q' to return to Test Menu)

Cmd - Option

f - Loop forever
h - Loop forever with halt on error
l - Loop once with loop on error
n - Loop forever with error messages inhibited
<cr> - Loop once


Cmd=>
```

f

The `Loop forever` command runs the test in an endless loop. Error messages are still reported.

h

The `Loop forever with halt on error` command runs the test in an endless loop until an error is detected. If an error occurs, testing halts.

l

The `Loop once with loop on Error` command runs the test once. If an error occurs, the test enters a scope loop.

n

The `Loop forever with error messages inhibited` command runs the test in an endless loop. No error messages are reported.

<cr>

The `Loop Once` command runs the test once, then returns to the current menu. All errors are reported. To select this option, press Return.

7

Sun-3 Self-tests and Initialization

Sun-3 Self-tests and Initialization **63**

7.1. The Power-Up Test Sequence 63
7.2. The Boot Sequence 64

7

Sun-3 Self-tests and Initialization

7.1. The Power-Up Test Sequence

In order to perform the power-up tests, two assumptions must be met. The MC68020 CPU must be functional and the ability to fetch instructions from the Boot PROM must be intact.

Powering up a Sun-3 workstation resets the CPU to *boot* state, which means that all instruction fetches are forced to the Boot PROMs. Execution of the minimum-confidence power-up tests begin immediately. These tests do not employ any memory until memory has been successfully checked.

The objective of the power-up test sequence is to determine whether or not the CPU board logic and main memory are functional. Following the successful completion of the power-up tests and subsequent workstation initialization, an attempt is made to boot the SunOS operating system, an EEPROM-specified program, or an operator-specified stand-alone program.

If the Diagnostic Switch at the rear of the system is in NORM position, you will not be able to interact with the self-tests. You may read the LEDs on the CPU board edge (described later in this chapter) to determine whether or not a test is failing, and you will see a rotating diagonal symbol after the

```
Testing _ megabytes of memory...
```

message on the console during the memory tests. The quantity of memory checked during a power-up with the diagnostic switch on NORM is dependent on EEPROM programming. The *EEPROM Layout* chapter explains how to set the parameter that controls the quantity of memory tested. If the workstation contains a large amount of main memory, self-tests may last three minutes or more.

If the Diagnostic Switch is in the NORM position, the power-up tests execute successfully and, if you do *not* terminate the default boot sequence, an attempt is made to down-load the SunOS operating system.

A display something like this appears on the *workstation's* screen to indicate that power-up tests are successful:

```
Selftest Completed Successfully.

         Sun Workstation, Model Sun-3/___ Series
         Type-3 keyboard
   ◆     ROM Rev __, _ MB memory installed, Serial #_______
         Ethernet address __:__:__:__:__:__

Testing __ megabytes of memory...Completed.

Auto-boot in progress...
```

7.2. The Boot Sequence

Following the initialization of the workstation, the default boot sequence is executed. There are two issues that must be considered here. One has to do with *what is to be down-loaded* while the other has to do with *where it is to be loaded from.*

Assuming *no* operator intervention, the position of the Diagnostic Switch will determine *what* is to be booted. If the Diagnostic Switch is in the NORM position, the SunOS operating system is booted. Otherwise, if the Diagnostic Switch is in the DIAG position, the EEPROM-specified program is booted. Be aware that the monitor is invoked if no EEPROM-specified program is available.

If you are in the PROM monitor mode, you may specify *what* is to be booted and *where* it is to be booted from. See command **b** (boot) in the chapter titled *Sun-3 PROM Monitor Commands* for a description of how to boot user-specified programs from user-specified devices.

If you are interacting with a Sun-3 workstation through the console to reach the monitor program, you may enter `L1-a` (or Break on a terminal) IMMEDIATELY after the

```
Testing __ megabytes of memory...Completed.
```

message.

CAUTION **Do not use `L1-A` procedure once the automatic boot has started; the file systems may be damaged if disks are powered-on and the operating system has started to run.**

If the operating system is already booted, use the procedures described in *Chapter 3* to start the monitor.

Once you are in the monitor mode (symbolized by the > prompt), you should do the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press `L1-a` or Break again when the message above finishes.

Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

The firmware also determines from what boot device the program will be loaded. If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is equal to 0x12 (an arbitrarily chosen value), Sun-3 firmware will attempt to boot the SunOS operating system from the boot path specified in the EEPROM, beginning at location 0x19. If the boot path is missing or contains an error, the monitor program is invoked.

If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is *not* equal to 0x12, Sun-3 firmware attempts to boot the SunOS operating system using the following boot device polling sequence:

1. Xylogics Disk.
2. SCSI Disk.
3. Ethernet.

If the Diagnostic Switch is in the DIAG position, the firmware assumes that both the path name of the file containing the to-be-loaded program and the boot device are specified in the EEPROM, beginning at EEPROM location 0x22. If either the file name or the boot device is not present or is in error, the monitor is invoked. If the Diagnostic Switch is in the DIAG position, you may connect a terminal to Serial A or B and interact with the self-tests and the Extended Test System, if required.

Diagnostic Power-Up

If the Diagnostic Switch is on DIAG, the self-test is executed as it is when the switch is "off" or on NORM, except that all of memory is tested. In addition, self-test status information is directed only to serial ports A and B, using the MMU (Memory Management Unit) bypass until all hardware required for the Video Monitor has been successfully tested.

Any hardware failures during the selftests will invoke scope loops to permit troubleshooting the failure. An RS-232 terminal with its characteristics set to 9600 Baud, 8 data bits, 1 stop bit and no parity should be connected to Serial Port A of the CPU board if you wish to view self-test status and interact with the system. If you use Serial Port B you must set the terminal baud rate to 1200.

Limited interaction with the self-test program is possible in this mode and the following characters will invoke the following actions when entered from the terminal connected to serial port A during self test. Each of these commands is documented below.

1. Press the `b` (a mnemonic for *burn-in*) key, prior to the display of the `...Completed` message, to execute the power-up test sequence

indefinitely. This option is useful during the manufacturing burn-in stage.

2. Press the **s** key prior to the display of the `...Completed` message to *re-start* the power-up test sequence.
3. If one of the power-up tests fails, it will continue to re-execute forever unless interrupted. Press the **space bar** to terminate the failed test and execute the next power-up test.
4. By default, an unsuccessful power-up test prints one error message before entering an infinite scope loop. Once inside of the scope loop, the failing test will *not* print any more messages. If, however, you would like to see test messages while in the scope loop, press the **p** key. Then, to turn the messages back off, press the **p** key a second time. In other words, the `p` command acts like a toggle switch to turn message mode on or off.

During a diagnostic boot, after self-test has completed successfully, you will be prompted to enter the extended tests, as shown below.

```
Selftest passed.

Optional Menu Tests

Type a character within 10 seconds to enter Menu Tests...(e for echo mode)
```

If you press a terminal key during this time the Extended Test Menu is displayed on the terminal screen. See the *Sun-3 Extended Test Sequence* chapter in this document for details.

If you do not assert control of the system by pressing a terminal key within the 10 seconds delay time the Boot PROM program will next display the Sun logo and message on the console.

Remote Testing

If you press **e** on a *dumb terminal* keyboard, during the ten second period, all subsequent output will appear on *both* the console video monitor *and* a terminal attached to Serial Port A or B.

The purpose of this feature is to enable an individual at a remote site, using a terminal attached to a non-local machine by way of a telephone line with modems at each end, and the individual on-site to observe the system's behavior simultaneously. Once a system is in this mode, the local and non-local parties can communicate by employing the # command. Specifically, if the person at the remote site would like to send a message to the person on-site, he or she would simply type the # before typing the actual message. In order to terminate the message, the person at the remote site would enter a second #. At this point, the on-site person would be able to respond by prefixing and terminating their response with the #.

The # option will be very useful in the situation where an on-site customer has contacted a remote repair depot regarding a potential hardware problem. The repair person at the remote repair depot could initiate diagnostics on the non-local machine and allow both parties to observe the output. Beyond that, the repair person and the customer could communicate by way of the # command.

Diagnostic Self-test Sequence

If the diagnostic switch is enabled, the name of each test appears on the terminal until all self-tests are complete. The tests differ slightly according to the system architecture. The examples that follow represent each Sun-3 workstation.

Figure 7-1 *Sun-3/75, 3/140, 3/150, 3/160, and 3/110 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  DVMA Reg Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  Parity Test
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type character within 10 seconds to enter menu tests... (e for echo mode)
```

Figure 7-2 *Sun-3/50 and Sun-3/60 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  Parity Tests
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type a character within 10 seconds to enter Menu Tests... (e for echo mode)
```

Figure 7-3 *Sun-3/260 and Sun-3/280 Diagnostic Boot Sequence*

```
Boot PROM Selftest

  PROM Checksum Test
  DVMA Reg Test
  Context Reg Test
  Segment Map Wr/Rd Test
  Segment Map Address Test
  Page Map Test
  Memory Path Data Test
  NXM Bus Error Test
  Interrupt Test
  TOD Clock Interrupt Test
  MMU Access Bit Test
  MMU Access/Modify Bit Test
  MMU Invalid Page Test
  MMU Protected Page Test
  ECC Error Tests
  Cache Data 3 Pat Test
  Cache Tags 3 Pat Test
  Memory Size = 0x00000xxx Megabytes
  Memory Test (testing xxxxxxxx MBytes)

Selftest passed

Optional Menu Tests

Type character in next 10 seconds to enter menu tests... (e for echo mode)
```

Test Descriptions

The following paragraphs describe each individual test, and the messages and indications generated if it fails. All tests are performed with the MC68020 cache enabled.

Diagnostic Register Test

The first test is indicated by a slow loop through the Diagnostic LEDs, intended to determine if the CPU is able to fetch instructions correctly from the Boot PROM and transfer data across the data bus.

NOTE *If all the LED s remain lighted, you could have a low voltage or board seating problem, or the system could contain the wrong Boot PROM.*

SCC (Z8530) Wr/Rd Test Through MMU

(Sun-3/50 and 3/60 only)

The ability of the CPU to communicate with the Zilog Z8530 Serial Communications Chip (SCC) is performed as the first test. Because the Sun-3/50 and the Sun-3/60 CPU board does not provide a means to communicate with the RS-232 Ports A and B except through the MMU, this test sets up the highest addresses in both the Segment MAP RAM and Page Map RAM to point to the base of the Z8530 SCC chip and performs a write/read test of WR 12/RR 12, an internal register within the SCC chip. This test is provided to get the CPU to SCC path tested so that all subsequent tests may display test name and error status to a terminal attached to Serial Port A; consequently no error message is displayed during this test.

NOTE *The diagrams that follow depict the LED states for each self-test. Note that the LEDs are shown here for convenience in reading as binary numbers; the least significant bit is on the right. In most systems, the LEDs are read with bit 0 on the left for desktop installations and with bit 0 on top for deskside installations. The LEDs on the Sun-3/60 CPU board are the exception; they are read as shown in the following diagrams.*

The tests are described here in the order that they are executed. Tests that do not apply to all systems are inserted in as close to the actual sequence as possible.

Boot PROM Checksum Test

The Boot PROM checksum is calculated and compared to the expected value. If this test fails the Boot PROM may need replacing.

Checksum compare errors will cause the following terminal display:

```
Err 16: PROM checksum error, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x01	Bit 7 ○○○○○○○● Bit 0	okay
0x81	●○○○○○○●	error

DVMA Register Test

This test is not performed on the Sun-3/50 or 3/60. Data from 0xff to 0x00 is written to this register and then read back and compared. The test enters a scope loop on data compare errors, and may display this message on the terminal:

```
Err 1: exp xxxxxxxx, obs xxxxxxxx, xor xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x02	**Bit 7 ○○○○○○●○ Bit 0**	**okay**
0x82	●○○○○○●○	**error**

Context Register Test

Data from 0x07 to 0x00 is written to this register and then read back and compared. The test enters a scope loop on data compare errors, and may display this message on the terminal:

```
Err 1: exp xxxxxxxx, obs xxxxxxxx, xor xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x03	**Bit 7 ○○○○○○●● Bit 0**	**okay**
0x83	●○○○○○●●	**error**

Segment Map RAM Wr/Rd Test

The Segment Map RAM Write/Read Test writes the Segment Map RAM with float bit patterns for each address of the Segment Map RAM. If a write/read compare error occurs during the test, the Segment Map RAM address is looped on after the program displays the RAM address and good and bad compare data.

The test will scope loop on data compare errors and may display this message:

```
Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x04	**Bit 7 ○○○○○●○○ Bit 0**	**okay**
0x84	●○○○○●○○	**error**

Segment Map Test

Three passes of write/read tests are performed on the Segment Map RAM. In the first pass, the consecutive patterns of hexadecimal 5A, 2C, 97 are written and read. In the second pass, the consecutive patterns 2C, 97, 5A are written and read. And, in the last pass, consecutive patterns of 97, 5A, 2C are written and read. The expected values are compared with the values read for each pass and data compare errors are reported.

The segment map test will scope loop on data compare errors with the following message:

```
Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x05	Bit 7 ○○○○○●○● Bit 0	okay
0x85	●○○○○●○●	error

Page Map Test

The page map test runs in context 0 only, with the segment maps set up to unity map, that is, address 0 will contain 0, address 1, 1, etc. Three passes of write/read tests are performed on the Segment Map RAM. In the first pass, the consecutive patterns 5A000C5A, 5A00072C, and 2C000A97 are written and read. In the second pass, the consecutive patterns 5A00072c, 2C000A97, and 5A000C5A are written and read. And, in the last pass, consecutive patterns of2C000A97, 5A000C5A, 5A00072C are written and read. The expected values are compared with the values read for each pass and data compare errors are reported. Data compare masking is appropriately provided for differences in Page Map RAM width for the three workstation models.

The page map test enters a scope loop on data compare errors and may display this message:

```
Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x06	Bit 7 ○○○○○●●○ Bit 0	okay
0x86	●○○○○●●○	error

Memory Path Data Test

The Memory Path Data Test is the first test of the ability to write/read access memory thru the Segment and Page Map RAM. This test writes and reads memory addresses from 0x400 to 0x1000 with a float bit pattern in order to test the CPU to memory data bus path and provide a tight scope loop in event of errors without testing all of memory.

Possible error messages are:

```
Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x07	Bit 7 ooooo••• Bit 0	okay
0x87	•oooo•••	error

NXM Bus Error Test

Bus errors are tested by writing to an invalid page, writing to a protected page, and by writing to non-existing memory. Upon error the test enters a scope loop after displaying an error message:

Possible error messages are:

```
Err 3: No NXM Bus Error exp xxxxxxxx, obs xxxxxxxx

Err 7: bus error reg  exp xxxxxxxx, obs xxxxxxxx, xor xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x08	Bit 7 oooo•ooo Bit 0	okay
0x88	•ooo•ooo	error

Interrupt Test

With the processor allowing all interrupts, the level 1 interrupt is enabled and verified to occur. The test enters a scope loop upon error with the following, and may display this message:

```
Err 4: No Level 1 interrupt!
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x09	Bit 7 oooo•oo• Bit 0	okay
0x89	•ooo•oo•	error

TOD Clock Interrupt Test

With the processor enabled for level 7 interrupt and real-time clock chip enabled for 100 HZ interrupts, the interrupt is verified to occur. The test enters a scope loop upon error and may display this error message:

```
Err 4: No TOD Clock interrupt!
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0a	Bit7 oooo•o•o Bit0	okay
0x8a	•ooo•o•o	error

MMU Protection/Status Tests

All MMU tests enter a scope loop upon error after displaying an error message.

Read accessing a page is verified to set the access status bit for the page.

Write accessing a page is verified to set the access and modified bits for that page.

A read access of an invalid page is verified to cause a bus error and set the invalid access status bit in the bus error register.

An attempt is made to write a write protected page and is verified to cause a bus error with the protect error bit set in the bus error register.

Possible error messages are:

```
Err 6: No bus error in accessing invalid page, addr xxxxxxxx

Err 5: Page Map status, exp xxxxxxxx, obs xxxxxxxx, xor xxxxx

Err 8: No bus error when writing a protected page
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0b	Bit7 oooo•o•• Bit0	okay
0x8b	•ooo•o••	error

Parity Error Tests (Not for Sun3/2xx)

This test verifies that writing and reading a known good memory address with parity checking enabled does not cause a parity error.

Possible error messages are:

```
Err 10: Parity error at Addr xxxxxxxx, Data xxxxxxxx, Status Exp xxxxxxxx, Obs xxxxx
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0e	Bit 7 ○○○○●●●○ Bit 0	okay
0x8e	●○○○●●●○	error

Parity Error Tests (Not for Sun-3/2xx)

The second parity error test verifies that forcing a parity error by writing bad parity causes a parity error NMI trap (autovector level 7). Correct parity error register status is also verified.

Possible error messages are:

```
Err 9: Bad parity should cause nmi.

Err 10: Parity error at Addr xxxxxxxx, Data xxxxxxxx, Status Exp xxxxxxxx, Obs xxxxxxxx
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0f	Bit 7 ○○○○●●●● Bit 0	okay
0x8f	●○○○●●●●	error

ECC Error Tests (Sun-3/2xx only)

By forcing a single bit error, the test verifies that the memory read data gets corrected. Then in a second pass of the test a two bit error is forced and the syndrome code is read to verify that a two bit error is detected but not corrected.

Upon failure the test enters a scope loop after displaying an error message. Possible error messages are:

```
Err 11: Syndrome error, exp xxxxxxxx obs xxxxxxxx

Err 20: Data correction error: exp xxxxxxxx obs xxxxxxxx
```

The diagram below summarizes the LED states:

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0c	Bit 7 ○○○○●●○○ Bit 0	okay
0x8c	●○○○●●○○	error

Cache Data 3-Pattern Test (Sun-3/2xx Only)

Three passes of pattern writes and reads of the entire cache data RAM address space are performed. The patterns used for the three passes are A5972C5A, 5AA5972C, and 2C5AA597 for the first pass, 5AA5972C, 2C5AA597, and 972C5AA5 for the second pass, and, 2C5AA597, A5972C5A, and 5AA5972C for the final pass.

Upon failure the test enters a scope loop after displaying an error message:

```
Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0d	Bit 7 oooo••o• Bit 0	okay
0x8d	•ooo••o•	error

Cache Tags 3-Pattern Test (Sun-3/2xx Only)

Three passes of pattern writes and reads of the entire cache tag RAM address space are performed. The patterns used for the three passes are A5972C5A, 5AA5972C, and 2C5AA597 for the first pass, 5AA5972C, 2C5AA597, and 972C5AA5 for the second pass, and, 2C5AA597, A5972C5A, and 5AA5972C for the third, and last pass.

A possible error message is:

Err 2: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0f	Bit 7 oooo•••• Bit 0	okay
0x8f	•ooo••••	error

Memory Sizing (Sun-3/2xx Only)

The Sun-3/2xx memory boards are read individually to determine the type of memory board (Old or New), the size of each board (8 or 32 Megabytes), a count of the total memory boards in the system, and to get the total size of main memory. The address of each board is set, relative to the number of megabytes previously read, and each board is enabled, allowing access to all available RAM. The following message is transmitted to the serial port after the memory sizing routine:

Memory Size = 0x00000xxx Megabytes

After sizing memory and if the diagnostic switch is in the diagnostic position, all of memory found will be tested. If the diagnostic switch is in the off NORM position, only the amount of memory specified in the EEPROM configuration information will be tested.

If the Esc key is pressed during any part of the Selftest sequence, the main memory is first sized and then fully initialized.

ECC Memory Initialization (Sun-3/2xx only)

The ECC main memory requires initialization by writing all of memory once with the pattern (0xa5a5a5a5). This will set the syndrome bits for all of memory to be tested. After writing the pattern, ECC is turned on for all memory boards found in the system, and left on for the duration of the ECC Modulo 3's Memory Test. At the beginning of ECC memory initialization, the following message is transmitted to the Serial port:

```
Initializing ECC Memory
```

Memory Modulo 3's Tests (Sun-3/2xx Only)

The Memory Modulo 3's Test first displays 0x0f on the diagnostic leds, to show the test is running. A modulo 3 pattern, residing in EPROM, is then read into memory at location 0x900. At this point the first Megabyte of RAM is mapped into a known location in the page table. Then the modulo 3's pattern is written to the entire Megabyte that was mapped in. When the Write is completed, a Read of each location in the Megabyte is done, comparing the value observed with the value expected. If no Correctable Errors (CE) are found, the test will update the display with the following message after testing each Megabyte:

```
Memory Test (Testing 0x00000xxx Mbytes)  Meg #0x00000xxx
```

The Memory Test will then attempt to map in the next Megabyte and test as before, continuing until all of main memory has been tested. This type of mapping has been implemented to allow the testing of memory sizes that are greater than 32 megabytes. The normal memory mapping is restored at the end of the Selftest sequence, before booting the operating system.

If a single bit Correctable Error (CE) is found during the memory test, a trap will occur, causing the test to enter a tight scope loop for trouble shooting, and displaying the following message:

```
Err 18: CE on Mem Bd 0x00000xxx,

Bd Addr = 0x00000000, Syn bits = 000000xx
```

If a double bit Uncorrectable Error (UE) is detected during the memory test, a trap will occur that causes the test to enter a tight scope loop for troubleshooting and display the following message:

```
Err 18: UE on Mem Bd 0x00000xxx,

Bd Addr = 0x00000000, Syn bits = 000000xx
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x0f	Bit 7 oooo•••• Bit 0	okay
0x8f	•ooo••••	error

Non Sun-3/2xx Memory Tests

Memory is initialized before testing, with the pattern 0xFFFFFFFF for each long word of memory above address 0x400, i.e., above trap/vector space. During initialization of memory it is sized by detecting the timeout bus error at the first nonexistent memory address.

The following message is displayed on the serial port terminal during memory sizing and initialization:

```
Sizing memory (size = xxxxxxxx MBytes)
```

After sizing memory and if the diagnostic switch is in the diagnostic position, all of memory found is tested. If the diagnostic switch is in the NORM position, only the amount of memory specified in the EEPROM configuration information is tested.

Memory Test (Non Sun-3/2xx CPU Boards)

The context register, segment maps, and page maps are all initialized with virtual addresses mapped to physical addresses in a one-to-one manner. Three passes of write/read tests are performed on all dynamic memory being tested.

In the first pass, the consecutive patterns 5A972C5A, 5A5A972C, 2C5A5A97 are written and read throughout the memory. In the second pass, the consecutive patterns 5A5A972C, 2C5A5A97, 72C5A5A are written and read. And, in the last pass, consecutive patterns of 2C5A5A97, 972C5A5A, 5A972C5A are written and read. The expected values are compared with the values read for each pass and data compare errors are reported. During the test, parity error interrupts (NMI) are enabled and serviced by the program.

The memory tests scope loop on data compare errors; however, the loop includes the pattern write for the entire memory space being tested.

The following message will be displayed during the Memory Test:

```
Memory Test (Testing 0x00000xxx MBytes)  Testing...
```

Possible error messages are:

```
Error: addr xxxxxxxx, exp xxxxxxxx, obs xxxxxxxx

Memory parity err, addr xxxxxxxx, exp xxxxxxxx

Err 12: NMI err, int with bad status, obs 0x00000000
```

Hexadecimal Value Of LEDs	Visual Representation	Condition
0x11	Bit 7 ○○○●○○○○ Bit 0	okay
0x91	●○○●○○○○	error

8

Sun-3 PROM Monitor Commands

Sun-3 PROM Monitor Commands **81**

8.1. Monitor Command Overview 81

8.2. The Monitor Commands 83

8

Sun-3 PROM Monitor Commands

This chapter describes the PROM monitor (sometimes called the ''system monitor'') commands available on Sun-3 workstation. Taken as a whole, these commands offer a low-level user interface to the Sun hardware. The commands control a variety of options, including booting from an alternate device, changing the console output, reading or altering registers or memory locations, and so on.

The effects of most monitor commands disappear when the power is turned off, with the exception of the `q` command, which programs the EEPROM. Parameters entered with the `q` command remain until deliberately altered with another PROM command. Programming the EEPROM *reconfigures* your workstation. The Boot PROM consults the EEPROM to determine whether or not to poll for a boot device, whether to boot a specified program, which device is the console, and so on. See the `q` command in this chapter, for details on how to do this.

Bringing up the PROM Monitor

Read *Chapter 3* for instructions on bringing up the PROM monitor.

8.1. Monitor Command Overview

The following example shows the menu that comes up when you enter **h** (help) at the monitor prompt. This section describes the general characteristics that all of the commands have in common. Detailed descriptions of each command, and its arguments, are listed in the next section. Some commands are available only for Sun-3/2xx workstations, and those will be identified.

NOTE The `i`, `j`, `n`, *and* `y` *commands are only available on Suns with on-board cache memory, such as the Sun-3/2xx.*

Figure 8-1 *Sun-3 Monitor Help Menu*

```
Boot PROM Monitor Commands
---------------------------------------------------------------
a [digit]                               |Open CPU Addr Reg (0-7)
b [dev([cntrl],[unit],[part])]          |Boot a file
c [addr]                                |Continue program at Addr
d [digit]                               |Open CPU Data Reg (0-7)
e [addr]                                |Open Addr as 16 bit word
f beg_addr end_addr pattn [size]        |Fill Memory
g [addr]                                |Go to Addr
h                                       |Help Menu
i [addr]                                |Open Cache Data
j [addr]                                |Open Cache Tags
k [number]                              |Reset (0)CPU, (1)MMU, (2)System
l [addr]                                |Open Addr as 32 bit long
m [addr]                                |Open Segment Map
n [i/e/d]                               |Cache Invalidate/Enable/Disable
o [addr]                                |Open Addr as 8 bit byte
p [addr]                                |Open Page Map
q [addr]                                |Open EEPROM
r                                       |Open CPU Regs (i.e. PC)
s [digit]                               |Set/Query Function Code (0-7)
t [y/n/c]                               |Trace: Yes/No/Continue
u [arg]                                 |Select Console Device
v beg_addr end_addr [size]              |Display Memory
w [addr] [string]                       |Vector
x                                       |Extended Diag Tests
y [c cxt] [s cxt sg_addr] [p cxt pg_adr]|Flush Cntxt/Seg/Page
z [addr]                                |Set Breakpoint
---------------------------------------------------------------
```

Executing a Command

In general, to execute a command you type the appropriate command letter, followed by any required command arguments. For example, if you wanted to execute the hypothetical command "θ" (which we will say needs two arguments `100` and `200`) you would type a line like this:

```
> θ 100 200
```

The command letter can be upper or lower case, and all commands and arguments are separated by white-space (tabs or spaces). Pressing the return key executes the command.

Default Values

Many of the monitor commands have built in, or *default* values, which the command uses if you do not supply arguments. The default values vary from command to command. Check the command descriptions for the default values of interest.

Word Sizes

Word sizes referred to in this chapter are defined as follows: A **byte** is eight bits long; a **word** is 16 bits long; a **long word** is 32 bits long.

8.2. The Monitor Commands

This section provides more detailed information on the commands listed in the help menu example. Both the commands and their arguments are described here.

Displaying and Modifying Memory

A number of the commands listed here can be used to display and/or modify the system's memory and registers. Regardless of the type of memory (RAM, EEPROM, etc.) or register, these commands have the same command syntax. This section describes the memory modification syntax used by the monitor commands. The example here references long words (32 bits) of memory, but the syntax is the same for bytes (8 bits) and words (16 bits).

The `address` argument specifies the initial memory location that is to be displayed and/or modified.

The second argument determines whether the current content of a memory location is to be displayed and/or modified. Entering *only* the address of a memory location after the command *displays* the content of that address.

```
>command FFE80000 Return
FFE80000: 12345678 ?
```

At this point, you can respond in one of *three* different ways.

1. Simply pressing the Return key displays the contents of the next memory location (in this case, 0xFFE80004) as shown below.

```
>command FFE80000
Return
FFE80000: 12345678 ?
Return
FFE80004: 00000001 ?
```

   Successively pressing the Return key displays the contents of successive memory locations. Assuming that you pressed Return four times, the contents of memory locations 0xFFE80004, 0xFFE80008, 0xFFE8000C and 0xFFE80010 would be displayed.

To exit the command, enter any *non-hexadecimal* character (`q` for quit, for example) before pressing Return Note that you will now return to the monitor's basic command level, with the > prompt.

```
>command FFE80000
Return
FFE80000: 12345678 ? q
Return
>
```

2. If you just want to modify one location, enter a command such as this:

```
>command FFE80000 00ABCDEF Return
```

3. The *third* response to the display of memory location contents is to *modify* those contents. You enter the *new* hexadecimal value immediately following the question mark ?, BEFORE pressing Return. The following display demonstrates how the value of memory location 0xFFE80000 can be changed from "12345678" to "00ABCDEF".

NOTE *Following the assignment of the new value to memory location 0xFFE80000, the new value will not be displayed; instead, the contents of the next memory location are shown. You will not be returned to the monitor's basic command level.*

```
>command FFE80000 Return

FFE80000: 12345678 ? 00ABCDEF Return

FFE80004: 00000001 ?
```

Entering a memory location's virtual address followed *only* by a non-hexadecimal character before pressing the Return key causes the monitor to *display* the contents of that location then exit to the monitor command level.

```
>command FFE80004 q Return
FFE80004: 00000001
>
```

Entering a non-hexadecimal character *and* a hexadecimal value after an address causes the monitor to display the original value at that virtual address, assign a new value, then display that value. For instance, assume that the original content at address 0xFFE80010 is "00000005". The command

```
>command FFE80010 ? 55555550 Return
```

displays the original value "00000005" then assigns the value "55555550" to address FFE80010 before returning to the monitor prompt:

```
>command FFE80010 ? 55555550 Return
FFE80010: 00000005 -> 55555550
>
```

You may also enter multiple display and/or modify commands for multiple memory locations on the same command line. If you enter this command line:

```
>command FFE80000 ? 00000000 ? ? 22222220 33333330 q Return
```

You will see this on the screen:

```
FFE80000: 12345678 -> 00000000
FFE80004: 00000001
FFE80008: 00000002 -> 22222220
FFE8000C -> 33333330
FFE80010: 00000004
>
```

The first part of the command line,

```
>command FFE80000 ? 00000000
```

displays the original contents of location FFE80000 before assigning the new value ''00000000'' to it.

The next question mark directs the monitor to display the contents of FFE80004. The next part of the command line, **`? 22222220`**, tells the monitor to display the original contents of FFE80008, before assigning the new value ''22222220'' to it. The **`33333330`** tells the monitor to assign the value ''33333330'' to memory location FFE8000C. Finally, the **`q`** causes the monitor to exit to the command level after the contents of FFE80010 are displayed.

Conventions

The paragraphs below describe each of the monitor commands in detail. Each paragraph starts with a line describing the command syntax. If a command has more than one distinct format, each one is shown on a separate line. Characters in **`bold typewriter`** font mean that you should enter them exactly as shown. Plain `typewriter` font represents what you should see on the screen. Words in *`typewriter italic`* show the type of information you are to enter. *Roman italic* or **boldfaced** font is also used for notes or emphasis within the text. Optional arguments and default values are listed in the descriptions.

Special Monitor Commands

Address Increment/Decrement Command

By preceding the command with a + or −, you can cause the address display to increment or decrement to the next location.

While traversing a range of addresses, type a + or − to change direction after the program displays the content and waits for input.
For example:

```
>l 0 Return

00000000 00000000 ?  Return

00000004 00000001 ?  Return

00000008 00000002 ? - (you enter a minus sign) Return

00000004 00000001 ?  Return (contents of previous location are displayed)

00000000 00000000 ? + Return (after decrement, you enter a plus sign)

00000004 00000001 ?  Return you increment again

00000008 00000002 ?  Return (you increment again)
```

The ^T Command

Entering the ^ character and then the `t` key, followed by a virtual address, displays the physical address to which that address is mapped, along with a detailed description of all the bits in the page table entry, the segment and page RAM addresses, and what space they are in.

For example, entering

```
>^t 1000 Return
```

results in this display:

```
Virtual Addr 1000 is mapped to Physical Addr 1000
Context = 0x0, Seg Map = 0x0, Page Map = 0xC0000000.
Page 0 has these attributes:

       Valid bit  = 0
       Permission = 1
       No Cache   = 0
       Type       = 0
       Access bit = 0
       Modify bit = 0
```

Entering `^t` with no arguments provides information with reference to virtual address 0x00.

The `^I` Command

Entering the `^` character and then the `i` key, followed with a command, displays compilation information for the system firmware. It includes the date, host name and build directory path. For example:

```
Compiled at 6/7/87 on hostname in /directory_name
```

The `^C` Command

`^C source destination n`

Entering the `^` character and then the `c` key, followed with the parameters shown, causes a block of `n` length to be copied from `source` to `destination` address, byte by byte. There is enough delay to copy to EEPROM also.

Regular Monitor Commands

Monitor `a` Command

`a register_number action`

The a command provides access to the CPU's address registers. `register_number` may be a value from 0 to 7 inclusive. The default value is 0. Register_number 7 accesses A7, the system stack pointer. To see the user stack pointer, use the `r` command.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`).
- You have manually "broken" into the monitor mode (by typing the `L1-a` sequence on a Sun keyboard or [Break] on a dumb terminal's keyboard).

Read the section entitled *Displaying and Modifying Memory* for details on how to use this command. Replace the word `command` in the description with the letter `a`.

Monitor `b` Command

`b` ? or `b! boot_device path argument_list`

The `boot` command loads and executes the SunOS operating system, an EEPROM-specified program, or a user-specified program. The boot program can be loaded from the default device, the device specified in the EEPROM, or the boot device specified in the command argument. A `boot_device` is a secondary storage device (disk, Ethernet or tape) that contains the program to be loaded and executed.

If the diagnostic switch on the back of the system is in the NORM position, the value in EEPROM address 0x18 is *not* equal to 0x12, and the boot command is entered without arguments, the system will boot the SunOS operating system, using the following default boot device polling sequence.

1. Xylogics Disk
2. SCSI Disk
3. Ethernet

If the EEPROM value at address 0x18 *is* equal to 0x12, the system will boot the SunOS operating system from an EEPROM-specified device. The boot device is specified in locations 0x19 through 0x1D, inclusive, of the EEPROM. Refer to the `q` command for information on how to open and modify these EEPROM locations.

When the diagnostic switch is in the DIAG position and command **b** is entered by itself, the system will boot an EEPROM-specified program from an EEPROM-specified device. In this case, the boot path is specified in locations 0x28 through 0x50, inclusive, of the EEPROM; the boot device is specified in locations 0x22 through 0x26, inclusive, of the EEPROM. If the boot attempt fails, the user is returned to the monitor's command level.

In order to boot from a specific device, the **b** command must be followed with a boot device abbreviation, such as those shown below. Enter **b ?** to view the boot device identifier arguments that your PROM monitor will accept.

```
b device(controller,unit,partition)path argument_list
```

You must surround `controller,` `unit,` and `partition` with parentheses, and place a comma after each entry. To invoke the default values, simply enter:

```
b device (,,)
```

device may be one of the following:

xy — Xylogics 450/451 disk
sd — SCSI disk
ie — Intel Ethernet
st — SCSI tape
xt — Xylogics 472 Tape
mt — Tape Master 9-Track Tape

controller stands for the Controller Number, referring to the tape or disk controller board. The default is 0.

unit refers to Unit Number, meaning disk number. The default is 0.

partition is the partition number of the boot device. The default is 0.

path is the path and filename of the program to boot.

argument_list is the list of arguments for the boot program. Up to seven optional arguments (which are passed to the boot program) may follow the path argument.

If you do not want the system to be reset prior to booting, you must insert ! before the device identifier argument.

```
b ie(0,0,1)/stand/video.diag -t
```

The boot command shown above would boot the video diagnostic from the `/stand` directory over the Ethernet. Because the ! argument does *not* precede the device identifier argument, the system is reset before the booting process begins. Note that one optional argument (`-t`) is passed on to program `video.diag`.

Monitor **c** Command

`c` *`virtual_address`*

The `continue` command resumes execution of an interrupted program. You can specify the virtual address of the instruction to be executed when the program restarts.

By default, the program resumes execution at the address pointed to by the program counter.

NOTE *This command is helpful if you should use the* `L1-A` *sequence and decide you did not want to abort the operating system.*

Monitor **d** Command

`d` *`register_number action`*

The `data register` command provides access to the CPU's data registers. *`register_number`* can be a value from 0 to 7 inclusive. The default value is 0.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`)
- You have manually "broken" into the monitor (by typing `L1-a` on the console's keyboard or **`break`** on the "dumb" terminal's keyboard).

Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **d**.

Monitor **e** Command

`e` *`virtual_address`*

The `display/modify memory` command displays and/or modifies the content of one or more virtual addresses in *word* mode (i.e. each virtual address will be treated as a 16-bit unit). That is, the content of one or more words can be *displayed*, *modified* or, both *displayed* and *modified.*

See the previous section *Displaying and Modifying Memory* for a description of this command's syntax. Use the letter **e** in place of the word *`command`* in the description.

Monitor **f** Command

f *`start_virtual_address end_virtual_address pattern size`*

The `block write` command writes the *`pattern`* you enter into each byte, word or long word in the range of virtual addresses you specify with the *`start_virtual_address`* and *`end_virtual_address`* arguments . By default, if the final, memory-cell-size argument is not present, bytes are written. Arguments *`start_virtual_address end_virtual_address`* and *`pattern`* are required while *`size`* is optional. The possible values for *`size`* are `b` (8-bit byte), `w` (16-bit word) or `l` (32-bit long word).

Monitor **g** Command

g *`vector argument`*
or

g *`virtual_address argument`*

When you use the `goto` command, you may go to (jump to) a *pre-determined, user-specified* or *default* routine. If a pre-determined or default routine is invoked, optional arguments `vector` (a hexadecimal number) and `argument` (a string) are passed to the routine to be executed.

In its other form, the argument `virtual_address` is used to indicate the virtual address of a *user-specified* routine, and the optional `argument` is passed along to that routine. If you do not supply the `vector/virtual_address` argument, the value in the Program Counter is used.

In order to set up a *pre-determined* routine, a user program has to set variable `*romp->v_vector_cmd` equal to the virtual address of the routine. Variable `*romp->v_vector_cmd` must be set prior to executing the `g` command. Pre-determined routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified `vector` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are `%x` and `%d`. Format `%x` prints argument `vector` as a hexadecimal number while format `%d` prints argument `vector` as a decimal number.

Monitor **h** Command

h

The `help` command brings up a Help display that describes the basic monitor commands and their argument(s). No arguments are accepted by the **h** command. An example of the help menu is shown at the beginning of this chapter.

Monitor **i** Command — Sun-3/2xx Only

i *`cache_data_offset`*

The `modify cache data RAM` command displays and/or modifies the contents of one or more of the cache data addresses. Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **i**.

Monitor `j` Command — Sun-3/2xx Only

`j` *`cache_tag_offset`*

The `modify cache tag` RAM command displays and/or modifies the contents of one or more of the cache tag addresses. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`j`**.

Monitor `k` Command

`k` *reset_level*

The `reset` command performs various levels of system resets. It can also be used to display the system's banner. This command accepts one optional argument. Entering **`k 0`** resets the VME-bus, interrupt register and video monitor (the Low-Level Reset).

Entering **`k 1`** invokes a Software Reset.

Entering **`k 2`** invokes a Power-On Reset (Hard Reset).

Finally, entering **`k b`** displays the banner on the video monitor. The default value of the argument is `0`.

Monitor `l` Command

`l` *`virtual_address`*

The `modify long words of memory` command displays and/or modifies the contents of one or more virtual addresses in *long* word mode. Each virtual address is treated as a 32-bit unit (long word). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`l`**.

Monitor `m` Command

`m` *virtual_address*

The `modify segment table` command displays and/or modifies the contents of one or more of the Segment Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`m`**.

Monitor `n` Command — Sun-3/2xx Only

`n` *`cache_command`*

The `control cache` command globally controls the cache. One argument is required. Entering **`n d`** *disables* the cache. Entering **`n e`** *enables* the cache. Finally, entering **`n i`** *invalidates* the cache.

Monitor `o` Command

`o` *virtual_address*

The `modify bytes of memory` command displays and/or modifies the content of one or more virtual addresses in *byte* mode. Each virtual address is treated as an 8-bit unit (byte). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`o`**.

Monitor **p** Command

p *`virtual_address`*

The `modify page table` command displays and/or modifies the contents of one or more of the Page Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **p**.

Monitor **q** Command

q *`eeprom_offset`* or **q ***

The `modify bytes of EEPROM` command displays and/or modifies the configuration information within the EEPROM in *byte* mode. This command works similarly to the memory commands discussed previously, except that the modified addresses are offset, and the changes you make remain when you power-down the workstation. Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **q**. Refer to the *Sun-3 EEPROM Layout* chapter for information on what parameters are stored at which EEPROM addresses.

q*—Version 2.4 and later Boot PROM Only

In the Version 2.4 Boot PROM, you may enter an asterisk instead of an EEPROM offset value as an argument, and the **q** command will erase the entire EEPROM.

Monitor **r** Command

r

The `miscellaneous register` command displays and/or modifies the contents of the CPU's miscellaneous registers. These registers are listed in the table below:

Table 8-1 *Miscellaneous Registers for the 68020*

68020 Miscellaneous Registers	
Symbol	*Name*
`IS`	Interrupt Stack Pointer
`MS`	Master Stack Pointer
`US`	User Stack Pointer
`SF`	Source Function Code
`DF`	Destination Function Code
`VB`	Vector Base
`CA`	Cache Address Register
`CC`	Cache Control Register
`CX`	Context Register
`SR`	Status Register
`PC`	Program Counter

It is important to note that it is *not* always possible to display the registers. They may be observed only after

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`) or
- You have manually "broken" into the monitor (by typing `L1-a` on the Sun keyboard or Break on a "dumb" terminal's keyboard).

Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *command* in the description with the letter **r**.

Monitor **s** Command

s *number*

The `modify` command sets or displays the address space to be used by subsequent memory access commands. The processor function codes decode the address spaces. Argument choices represent the function codes:

Table 8-2 *Function Code Values*

Value	Address Space
0	Reserved (don't use)
1	User Data
2	User Program
3	Control Space
4	Reserved (don't use)
5	Supervisor Data
6	Supervisor Program
7	CPU Space

If you do not enter a function code number, the current setting (either 1 or 5) is displayed, and entry of the monitor **o** command, for example, would cause the program to look for the specified address in either user or supervisor data space. Conversely, if you reset the function code number, you could query registers in the space represented by the number entered. For example, entering

```
>s 3
```

would allow you to read the bus error or system enable register, which are located in Control Space.

Monitor **u** Command

u *port options baud_rate*
or

u *echo*
or

u **u***virtual_address*

The *input/output* command configures the input and output devices and their characteristics or displays the current input and output device set-up.

The **u** command requires arguments to specify from which device(s) you want the system to expect input or which device(s) will display output.

If you do not enter an argument after the `u` command, the program will display the current settings. If no serial port is specified when changing baud rates, the baud rate of the current input device is changed. The default serial port baud rate is 9600 for both ports during a normal boot. During a diagnostic boot, defaults are 9600 baud for Port A and 1200 for Serial Port B.

Upon normal power-up (diag switch is in NORM position), the default console input device is the Sun keyboard, unless the EEPROM has specified another default input device. If the keyboard is unavailable, the system looks to serial port A for for input.

The default console output device is the Sun monitor (subject to change through EEPROM programming). If the workstation has a color monitor and a color board is unavailable, the program will look for a monochrome monitor as an output device.

You may alter the existing I/O settings while you are in the monitor mode, using the commands listed below; however, the default settings will be reinstated when the system is power cycled.

u Command Arguments:

The `port` argument can be one of the following:

Table 8-3 *Port Arguments*

Port Argument	Device
`a`	Serial Port A
`b`	Serial Port B
`k`	Keyboard
`s`	Screen

The `options` arguments are:

Table 8-4 *Option Arguments*

Option Argument	Meaning
`i`	input
`o`	output
`u`	UART
`e`	input echoed to output
`ne`	input *not* echoed to output
`r`	reset specified serial port

The `baud_rate` argument specifies baud rate of the serial port being discussed. The `virtual_address` argument specifies the virtual address of the UART (Universal Asynchronous Receiver/Transmitter).

Following are examples of port and options arguments:

- Enter `u aio` or `u bio` to select serial port A or B as the input and output device.
- Enter `u ai` or `u bi` to select serial port A or B for input only.
- Enter `u ao` or `u bo` to select serial port A or B for output only.
- Enter `u k` to select the keyboard for input.
- Enter `u ki` to select the keyboard for input.
- Enter `u s` to select the screen for output.
- Enter `u so` to select the screen for output.
- Enter `u ks, sk` to select the keyboard for input and the screen for output.
- Enter `u a`*baud rate* or `u b`*baud rate* to set the serial port speed.
- Enter `u e` to cause the output to echo the input.
- Enter `u ne` to cause the output **not** to echo the input.
- Enter `u` *address* to set the serial port virtual address.

A *previously mapped* UART can also be used for input and/or output. Command `u uvirtual_address`", where `virtual_address` is the virtual address of a previously mapped UART, changes the virtual address of the UART. Do not enter a space between the second `u` and the virtual address.

If the `u` command is not followed by an argument, the current settings are displayed. The current input device, output device, baud rate for serial ports A and B, UART's virtual address and input to output echo indicator are shown.

Monitor v Command

`v start_virtual_address end_virtual_address size`

The `display memory block` The `display memory block` command displays the contents of each byte, word or long word in the range of virtual addresses specified by `start_virtual_address` and `end_virtual_address`. The `word_size` argument is optional; the default is to display memory in byte format. In this format, sixteen consecutive bytes are displayed on each line. In word format, eight consecutive words appear on each line.

If long word format is enabled, four consecutive long words appear on each line. To the far right of each line, the character corresponding to each byte is also shown. All bytes that contain a non-ASCII code are shown as a period (.). Legal values for `size` are **b** (8-bit byte), **w** (16-bit word), or **l** (32-bit long word).

To terminate the **v** command, press the *space bar.* To "freeze" the display, press any key *except* the space bar. To restart the **v** command, press the space bar again.

Monitor w Command

`w virtual_address argument`

The `set execution vector` command allows you to vector to a *pre-determined* or *default* routine. Optional arguments `virtual_address` and `argument` are passed along to the to-be-executed routine.

In order to set up a *pre-determined* routine, a user program sets the variable `*romp->v_vector_cmd` equal to the virtual address of the *pre-determined* routine. Variable `*romp->v_vector_cmd` must be set prior to executing the

w command. *Pre-determined* routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified `virtual_address` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument `virtual_address` as a hexadecimal number; format "%d" prints it as a decimal number.

Monitor **x** Command

x

The `extended test system` command invokes the Extended Test Sequence or the Extended Test System, depending on what Boot PROM revision is on the CPU board. See *Chapter 9* for details.

Monitor **y** Command — Sun-3/2xx Only

y *`c number`*
or

y *`cache_section number virtual_address`*

The `flush cache` command performs a number of *flush* operations on the cache. Depending on what is to be flushed, two or three arguments are required. In order to flush a context, enter command **y c** *`number`*, where *`number`* is a valid context number.

Entering **y s** *`number virtual_address`* flushes segment *`virtual_address`* within context *`number`*. The argument *`number`* is a valid context number.

Entering **y p** *`number virtual_address`* flushes page *`virtual_address`* within context *`number`*. The argument *`number`* is a valid context number.

Monitor **z** Command

z *`number breakpoint_virtual_address type len`*

Command **z** sets or resets breakpoints for debugging purposes. Optional argument *`number`* can have values from 0 to 3, corresponding to the processor debug registers DR0 to DR3, respectively. Up to four distinct breakpoints can be specified. If argument *`number`* is not specified, the monitor will choose a breakpoint number.

The argument *`breakpoint_virtual_address`* specifies the breakpoint address to set.

The argument *`type`* can have three values: **x** for Instruction Execution breakpoint, **m** for Data Write only breakpoint, and **r** for Data Reads-and-Writes-only breakpoint. The default value for *`type`* is **x**.

The argument `len` can also have three values: **`b`**, **`w`**, and **`l`**, corresponding to the breakpoint field length of byte, word, and long-word, respectively. The default value for `len` is `b`. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks.

If the argument `number` is specified but the argument `breakpoint_virtual_address` is not specified, then the corresponding breakpoint will be reset.

This command without any arguments will display all the existing breakpoints.

9

Sun-3 Extended Test Sequence

Sun-3 Extended Test Sequence 101

9.1. Extended Menu Test Descriptions 101
9.2. Sun-3 Extended Test Sub-Menus 107

9

Sun-3 Extended Test Sequence

The Extended Menu Tests are a suite of test routines that provide detailed testing of the system hardware. They provide an additional level of testing beyond the power-up self-tests. You can invoke the Extended tests in two ways.

If the system is being booted with the diagnostic switch in DIAG position, this message is displayed on the terminal screen:

```
Type a character within 10 seconds to enter Menu Tests...
```

If you do type a character, the Extended tests start.

The other way to invoke the Extended test is to type the **x** command from the PROM monitor. You are in the monitor when you see the > prompt on the screen.

9.1. Extended Menu Test Descriptions

The Extended tests are organized into a series of menus. Each menu follows, along with descriptions of the command choices.

Main Menu

Due to the many varieties of Sun-3 workstations, there are five different main menus, each representing a sub-class of Sun-3 systems. The menus are similar to one another, in that they all have many of the same commands, and a command of a given name works the same for all main menus.

The menu for the Sun-3/150, 3/160, 3/180 and 3/75 workstations looks something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ie  - Intel Ethernet Test
kb  - Keyboard Input Test
me  - Memory Test
mk  - Mouse/Keyboard Ports Test
mt  - Tapemaster Bootpath
rs  - Serial Ports Test
sd  - SCSI Disk Bootpath Test
st  - SCSI Tape Bootpath Test
vi  - Video Test
xd  - Xylogics 7053 Disk Bootpath Test
xt  - Xylogics Tape Bootpath Test
xy  - Xylogics Disk Bootpath Test

Cmd=>
```

The main menu for the Sun-3/50 workstations is a subset of that for the the Sun-3/160 due to the fact that the Sun-3/50 has a more limited set of peripheral I/O devices. On the Sun-3/50 CPU Board, the AMD Ethernet Chip set is used instead of the Intel Ethernet Chip set. The Sun-3/50 Extended Test main menu looks something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ae  - AMD Ethernet Test
kb  - Keyboard Input Test
me  - Memory Test
mk  - Mouse/Keyboard Ports Test
rs  - Serial Ports Test
sd  - SCSI Disk Bootpath Test
si  - SCSI Interface Test
st  - SCSI Tape Bootpath Test
vi  - Video Test

Cmd=>
```

The main Extended Test Menu for the Sun-3/110 workstation looks something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ie  - Ethernet Test
kb  - Keyboard Input Test
me  - Memory Test
mk  - Mouse/Keyboard Test
mt  - Tapemaster Bootpath Test
rs  - Serial Ports Test
sd  - SCSI Disk Bootpath Test
st  - SCSI Tape Bootpath Test
vm  - Video (Monochrome) Test
vc  - Video (Color) Test
ve  - Video Enable Plane Test
cm  - Color Map Test
xt  - Xylogics Tape Bootpath Test
xy  - Xylogics Disk Bootpath Test

Cmd=>
```

The main Extended Test Menu for the Sun-3/260 and Sun-3/280 workstations looks something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ie  - Ethernet Test
kb  - Keyboard Input Test
me  - Memory Test
mk  - Mouse/Keyboard Test
mt  - Tapemaster Bootpath Test
rs  - Serial Ports Test
sd  - SCSI Disk Bootpath Test
st  - SCSI Tape Bootpath Test
vi  - Video Test
xd  - Xylogics 7053 Disk Bootpath Test
xt  - Xylogics Tape Bootpath Test
xy  - Xylogics Disk Bootpath Test

Cmd=>
```

The main Extended Test Menu for the Sun-3/60 workstation looks something like this:

```
Extended Test Menu: (Enter 'q' to exit)

Cmd - Test

ae  - AMD Ethernet Test
kb  - Keyboard Input Test
me  - Memory Test
mk  - Mouse/Keyboard Ports Test
rs  - Serial Ports Test
sd  - SCSI Disk Bootpath Test
si  - SCSI Interface Test
st  - SCSI Tape Bootpath Test
vm  - Video (Monochrome) Test
vc  - Video (Color) Test
ve  - Video Enable Plane Test
cm  - Color Map Test
vo  - Video Overlay Plane Test

Cmd=>
```

The following paragraphs briefly describe the Extended Test Menu choices, in alphabetical order.

ae

The *AMD Ethernet Test* checks the AMD Am7990 (Lance) Ethernet Chip, the Am7991 (SIA) Serial Interface Unit Chip, the AM7995 Transmitter/Receiver Chip, and the connection to the Ethernet network. This command brings up the AMD Ethernet Test Menu which is discussed later in this chapter.

cm

The *Color Map Test* command checks the color map table memory space only in Sun-3/110's. This command displays the Color Map Menu.

ie

The *Intel Ethernet Test* checks the Intel 82586 Ethernet Coprocessor chip, the Intel 82501 Serial Interface Unit chip, and the connection to the Ethernet network. This command brings up the Intel Ethernet Test Menu, which is discussed later in this chapter.

kb

The *Keyboard Test* command determines whether or not keyboard characters are correctly transmitted to the CPU. When this test begins, the ASCII code corresponding to a character, and the character itself, appear on the screen as you press keys on the keyboard. To exit the test, type an ESC character.

me

The *Memory Tests* command invokes the Memory Menu. This menu of tests is described later in this chapter.

`mk`

The *Mouse/Keyboard Ports Tests* command invokes a sub-menu that contains the keyboard and mouse tests, which are discussed later in this chapter.

`mt`

The *Tapemaster Bootpath Test* command tests the bootpath to the 1/2-inch Tapemaster tape drive. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior.

`rs`

The *Serial Ports Tests* command invokes the Serial Ports Menu. This menu of tests is discussed later in this chapter.

`sd`

The *SCSI Disk Bootpath Test* command tests the bootpath to the SCSI hard disk. It makes sure the machine can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior.

`si`

The *SCSI Interface Tests* command checks the VLSI chips that constitute the SCSI interface logic. They are: the 6X80 FIFO chip, the UDC DMA Controller chip, the NCR 5380 SCSI Controller chip, and the addressing and data paths connecting them. This command displays the SCSI Interface Menu, which is discussed later in this chapter.

`st`

The *SCSI Tape Bootpath Test* command tests the bootpath to the SCSI tape drive. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior.

`vc`

The *Video Color Tests* command checks the 1 Mbyte frame buffer used for color video display in the Sun-3/110 and Sun-3/60. This command displays the Video Color Menu, discussed later in this chapter.

`ve`

The *Video Enable Plane Tests* command checks the 128 Kbyte video enable plane, present only in Sun-3/110s and Sun-3/60s. This command displays the Enable Plane Menu, discussed later in this chapter.

`vi`

The *Video Menu Tests* command invokes the Video Menu. This menu contains all of the video tests, which are discussed later in this chapter.

`vo` The *Video Overlay Plane Test (Sun-3/60 Only)* provides a test of the 128 Kbyte video overlay plane.

`xd` The *Xylogics 7053 Disk Bootpath Test* tests the bootpath to the Xylogics 7053 Disk Controller. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code.

`xt`

The *Xylogics Tape Bootpath Test* command tests the bootpath to the Xylogics tape drive. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior.

`xy`

The *Xylogics 450/451 Disk Bootpath Test* command tests the bootpath to the Xylogics 450/451 Disk Controller. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. Plane Menu, discussed later in this chapter.

9.2. Sun-3 Extended Test Sub-Menus

The sub-menus that some of the Main Menu selections bring up are described here, in this order:

- Ethernet Menus
- Memory Menu
- Mouse/Keyboard Menu
- Serial Port Menu
- SCSI Menus
- Video Menus
- Option Menu

AMD Ethernet Test Menu

This menu contains all the tests that check the AMD Ethernet chips present on the Sun-3/50 CPU Board.

```
Cmd - Test

w - Wr/Rd CSR1
l - Local loopback
x - External loopback

Cmd=>
```

The following paragraphs describe the AMD Ethernet Test Menu.

w The *Write and Read CSR1 Test* command writes and reads a specified pattern to Control/Status Register 1, a write/read data register within the AM7990 Lance chip.

l The *Local Loopback Test* command makes the Am7990 chip disconnect itself from the Serial Interface Unit chip, then perform an internal loopback within the AM7990 chip. This test checks the Am7990 chip in isolation.

x The *External Loopback Test* command requires that you attach an external terminator to the Ethernet connector. This test checks all three chips and the external connector.

Intel Ethernet Test Menu

This menu comes up when you select **`ie`** from a Sun-3/75, 150, 160, 110, 180, 260 or 280. It contains all the tests that check Intel Ethernet circuitry.

```
Cmd - Test

l - Local loopback
e - Encoder loopback
x - External loopback

Cmd=>
```

The following paragraphs describe the Intel Ethernet Test Menu selections.

`l`

The *local* command tests the Intel 82586 Ethernet LAN co-processor chip. It runs the internal tests built into the chip. Prior to the test, the Intel 82586 Ethernet LAN co-processor chip disconnects itself from the Intel 82501 Serial Interface Unit Chip.

`e`

The *encoder* command runs an internal loop back test of the Intel 82501 Serial Interface Unit Chip. The transmitter line, receiver line, noise filters and Manchester encoding/decoding logic are tested. Before executing this test, the transmitter and receiver lines of the chip are connected together.

`x`

The *external* command runs the external loop back test. This test checks the Intel 82586 Ethernet LAN Co-processor Chip, the Intel 82501 Serial Interface Unit Chip and the Ethernet transceiver and receiver lines. The test sends data out onto the Ethernet, then receives it back and compares the transmitted and received data. Before running this test, attach an Ethernet transceiver cable to the CPU board and the terminator assemblies to the black transceiver box.

Memory Menu

The Memory Menu is offered when you select **me** from any Extended Test Main Menu.

```
Enter Cmd [low address >= 0] [high address <= 3FBFFF] [pattern]

Cmd - Test

 a  - Address Test
 c  - Data Compare Test
 s  - Scan Memory
 w  - Write Only Pattern

Cmd=>
```

a`low_address high_address`

The *Address Test* command executes the address test on a range of memory. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. When the test runs, write-read-compare cycles are performed on long words. The datum that is written to each memory ''cell'' is its own address. This command accepts a maximum of two arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x2000`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

c `low_address high_address pattern`

The *Data Compare Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts a maximum of three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x2000`. The `low_address` value should be a hexadecimal (base 16) number.

CAUTION **Addresses 0x0 to 0x2000 are reserved for use by the boot PROM. If this test overwrites those locations, the system may crash.**

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

s *`low_address high_address pattern`*

The *Scan Memory* command reads the specified range of addresses, and compares them to the value set by *`pattern`*. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts three optional arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x2000`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*

The *Write Only Pattern* command writes a pattern to a range of addresses. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts three optional arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x2000`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names pattern written throughout the range of addresses. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

Mouse/Keyboard Ports Test Menu

When you select **mk** from the Extended Test Main Menu on any system, these tests are offered for one of the CPU board's two Z8530 Serial Communications Controller (SCC) chips, which controls both the keyboard and the mouse. The Keyboard and Mouse Menu contains four options. These options are presented below:

```
Mouse/Keyboard Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(M or K)] [Baud rate (decimal #)] [hex byte pattern]

Cmd  - Test


w  -  Wr/Rd SCC Reg 12 Test
x  -  Xmit Char Test
i  -  Internal Loopback test
e  -  External Loopback Test


Cmd=>
```

The following paragraphs describe the Mouse/Keyboard Ports menu choices. For each menu selection, the *channel* argument may be:

Letter	*Device*
k	Keyboard (default)
m	Mouse

w `channel pattern`

The *Wr/Rd SCC Reg* command performs write-read-compare cycles to register 12 of the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments. Refer to the beginning of this section for *channel* argument choices. The `pattern` argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The `pattern` should be a hexadecimal (base 16) number.

x `channel baud pattern`

The *Xmit* command writes patterns to the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments.

Refer to the beginning of this section for *channel* argument choices.

The `baud` argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

`i` *`channel baud pattern`*
The *`Internal`* command performs internal loop back write-read-compare cycles on the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The transmitter and receiver lines of the requested port are connected internally prior to the test.

This command accepts three arguments.

Refer to the beginning of this section for *channel* argument choices.

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of `baud` should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The `pattern` should be a hexadecimal (base 16) number.

`e` *`channel baud pattern`*
The `External` command executes an external loop back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (see *Chapter 1*).

When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments.

Refer to the beginning of this section for *channel* argument choices.

The `baud` argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of `baud` should be a decimal (base 10) number.

The `pattern` argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The `pattern` should be a hexadecimal (base 16) number.

Serial Ports Menu

This menu is selected with the **rs** Extended Test Menu command and provides testing of the second Z8530 SCC chip in the system, which controls serial ports A and B. The Serial Ports Menu contains these options:

```
Serial Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(A or B)] [Baudrate(decimal #)] [hex byte pattern]

Cmd - Test

w - Write/Read SCC Reg 12
x - Xmit char
i - Internal loopback
e - External loopback

Cmd=>
```

w *channel baud pattern*

The *Wr/Rd SCC Reg* command performs write-read-compare cycles to register 12 of the port under test. When the test is selected, but before it runs, the Options Menu is displayed, allowing you to control the test's behavior. This command accepts three arguments.

The *channel* argument determines which port is to be tested. Legal values for *channel* are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *baud* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number.

The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *pattern* should be a hexadecimal (base 16) number.

`x` *`channel baud pattern`*

The *`Xmit`* command writes patterns to the port under test. This command accepts three arguments.

The *`channel`* argument determines which port the test is performed on. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

`i` *`channel baud pattern`*

The `Internal` command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port are connected internally prior to the test.

When the test is selected, but before it runs, the Options Menu is displayed, from which you may control the test's behavior. This command accepts three arguments.

The *`channel`* argument determines which port the test is performed on. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

`e` *`channel baud pattern`*

The `External` command executes an external loop-back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop-back cable must be installed (see chapter one).

When the test is selected, but before it runs, the Options Menu is displayed, allowing you to control the test's behavior. This command accepts three arguments. The *`channel`* argument determines which port the test is performed on. Legal values for channel are shown in the table below:

Letter	*Device*
`a`	Serial Port A (default)
`b`	Serial Port B

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

SCSI Interface Menu

This menu contains the commands that test the SCSI interface and is called up when you enter **`si`** from the Sun-3/50 or 3/60 Extended Test Menu. It contains six options, shown below:

```
Cmd - Test

b - SCSI Byte Ctr Wr/Rd Test
c - SCSI CS Register Wr/Rd Test
f - 8X60 FIFO/UDC DMA Test
s - 8350 SBC Chip Wr/Rd Test
u - 9516 UDC Chip Wr/Rd Test
x - SCSI Bus External Loopback Test

Cmd=>
```

`b`

The *SCSI Byte Control Test* command writes and reads data to and from the SCSI byte counter.

`c`

The *CS Register Test* command writes and reads data to and from the SCSI Control/Status Register.

`f`

The *SCSI FIFO Test* command checks the SCSI FIFO Static RAM.

`s`

The *SCSI SBC Test* command runs a write and read test of the registers within the NCR 5380 SCSI Bus Controller Chip.

`u`

The *SCSI UDC Test* command runs a write and read test of the registers within the AMD 9516 Universal DMA Controller Chip.

`x`

The *SCSI External Loopback Test* command checks the NCR 5380 chip. It also tests the SCSI bus pins on the SCSI Bus Connector at the handle edge of the CPU board. Before running this test, place a SCSI loopback connector on the SCSI bus connector.

Video Menu

The Video Menu is called up when you enter the **`vi`** command from any Sun-3 Extended Test Main Menu, with the exception of the Sun-3/110 and Sun-3/60. Each of the menu options is covered in the text that follows the example menu below.

```
Video Tests:  (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 1FFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read/Compare Test
s - Scan Memory
w - Write Data Pattern

Cmd=>
```

a *`low_address high_address`*

The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum which is written to each memory ''cell'' is its own address. This command accepts a maximum of two arguments. The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

c *`low high pattern`*

The `Write/Read/Compare Pattern Test` command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing. The test accepts three optional arguments. The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names the pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

s *`low_address high_address pattern`*

The *Scan Memory Test* command reads the specified range of addresses, and compares the values to *`pattern`*. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*
The *Write Data Pattern* command writes a pattern to a range of addresses. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern written throughout the range of addresses. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

Monochrome Video Menu

This test is provided to test the 128 Kbyte frame buffer space for the Sun-3/110 or 3/60 video display. It is called up when you enter **vm** from the Extended Test Main Menu. The following options are available for write/read and address tests.

```
Monochrome Video Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 1FFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test

Cmd=>
```

a `low_address high_address`

The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum which is written to each memory "cell" is its own address. This command accepts two arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s `low_address high_address pattern`

The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*

The `Write Only Test` command writes a pattern to a range of addresses. The test accepts three optional arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

Video Color Menu

The following options are available when you select **vc** from the Sun-3/60 or Sun-3/110 Extended Test Main Menu:

```
Color Frame Buffer Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0xFFFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test

Cmd=>
```

a `low_address high_address`

The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts two arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s `low_address high_address pattern`

The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*

The *Write Only Test* command writes a pattern to a range of addresses. The test accepts three optional arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

Enable Plane Menu

Entering **ve** from the Sun-3/60 or Sun-3/110 Extended Test Main Menu brings up this sub-menu, which is provided to test the 128 Kbyte video enable plane present on the CPU Board in those systems. The following options are available:

```
Enable Plane Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0x1FFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test

Cmd=>
```

a *low_address high_address*
The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts two arguments.

The *low_address* argument specifies the first address to test. By default, the value of *low_address* is `0x0`. The *low_address* value should be a hexadecimal (base 16) number. The *high_address* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *high_address* value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s *low_address high_address pattern*
The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors. The test accepts three optional arguments.

The *low_address* argument specifies the first address to test. By default, the value of *low_address* is `0x0`. The *low_address* value should be a hexadecimal (base 16) number.

The *high_address* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *high_address* value should be a hexadecimal (base 16) number.

The *pattern* argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of *pattern* is `0xaaaaaaaa`. The *pattern* value should be a hexadecimal (base 16) number.

w *low_address high_address pattern*
The *Write Only Test* command writes a pattern to a range of addresses. The

test accepts a maximum of three optional arguments. The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

Color Map Menu

This menu contains all the tests for the Sun-3/110 or 3/60 color map. It comes up when you enter **cm** from the Extended Test Main Menu for those systems.

```
Color Map Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0x2FF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test
f - Fill color maps with default pattern

Cmd=>
```

a `low_address high_address`
The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts two arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number. The `high_address` arguments indicate the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s `low_address high_address pattern`
The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w `low_address high_address pattern`

The *Write Only Test* command writes a pattern to a range of addresses. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names pattern expected throughout the range of addresses. The observed values is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

f

The *Fill Color Map* command fills the color map table with a default pattern. The pattern is a set of linearly increasing values. Starting with the first location in the red map, a linear set of values, starting with 0, is entered. The last location in the red map gets `0xFF`.

A similar pattern is entered for the green map, except that the 0 value is entered two-thirds of the way down in the map, not at the beginning. The values wrap around the green map, so the `0xFF` value is placed in the location just before 0 entry. The blue map has a similar pattern, except the 0 is entered at one third of the way down in the map.

Video Overlay Plane Test Menus

The *Video Overlay Plane Test (Sun-3/60 Only)* provides a test of the 128 Kbyte video overlay plane when you enter **vo** from a Sun-3/60 Extended Test Main Menu. The following options are available.

```
Enable Plane Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0x1FFFF] [pattern]


      Cmd - Test

        a  - Address Test
        c  - Write/Read Test
        s  - Scan Memory Test
        w  - Write Only Test

     Cmd=>(Operator enters command line)
```

The test options sub-menu (described next) is then presented.

Test Options Submenu

This Submenu is displayed after selecting almost any test. It controls the number of times a test is executed, and what happens when an error is discovered.

```
Test Options: (Enter 'q' to return to Test Menu)

Cmd - Option

f - Loop forever
h - Loop forever with Halt on error
l - Loop once with Loop on error
n - Loop forever with error messages inhibited
<cr> - Loop once

Cmd=>
```

`f`

The *Loop Forever* command runs all tests in an endless loop. Error messages are still reported.

`h`

The *Loop until Error* command makes all tests run in an endless loop until an error is detected. If an error occurs, testing halts.

`l`

The *Loop on Error* command runs the test once. If an error occurs, the test enters a scope loop.

`n`

The *No Error Messages* command runs all tests in an endless loop. No error messages are reported.

`<cr>`

The *Execute Once* command runs the test once, then returns to the current menu. All errors are reported. Select this option by pressing Return.

10

Sun-3 EEPROM Layout

Sun-3 EEPROM Layout **131**

10.1. EEPROM Introduction 131

10.2. Changing EEPROM Parameters 131

10.3. The EEPROM Layout 132

10

Sun-3 EEPROM Layout

10.1. EEPROM Introduction

The PROM monitor **q** command opens the EEPROM to allow examination or modification of configuration parameters. If you do not enter an address following the command, the content of the first address assigned to the EEPROM is presented. (EEPROM addresses are off-set, rather than complete addresses.)

EEPROM parameters set these functions:

- vary the quantity of memory tested during self-test;
- change the action that follows a watchdog reset;
- boot from a specified device with diagnostics switch on NORM, or poll the devices;
- recognize the specified device as the primary terminal or console;
- display the Sun banner or a custom banner during power-up;
- store and display a custom logo upon power-up;
- turn the keyboard "click" on or off;
- select special keyboard characters;
- boot a selected program from a specific device with diagnostics switch on DIAG;
- inhibit serial port DTR and RTS signals;
- select a serial port baud rate;
- store a system configuration record on EEPROM;
- erase EEPROM contents.

10.2. Changing EEPROM Parameters

The *Sun-3 PROM Monitor Commands* chapter contains **q** command syntax variations. The paragraphs that follow represent examples of one way to change or view EEPROM parameters. The layout section describes which parameters are stored at which EEPROM addresses.

To change the value of a specific EEPROM address, you must be in the monitor mode, signified by the > prompt. Now, enter the PROM monitor command followed by the offset EEPROM address of the parameter you wish to change, and [Return]:

```
>q offset_address
```

When the program displays the contents of that location, enter the new value followed with a non-hexadecimal character, such as a period, or a **q** for quit, and [Return]:

```
>q 1f [Return]
>EEPROM 01f: 10? 11 .  [Return]
>
```

To exit from the modify mode when you have *not* entered a new value, simply press the space bar and [Return] after the question mark.

To increment to the next EEPROM address instead of returning to the PROM monitor program, simply press [Return] after the question mark, or immediately following your entry.

10.3. The EEPROM Layout

This section has a detailed description of the EEPROM layout. The layout is divided into a diagnostic section, a reserved section, a ROM section, and a software section.

The table on the following page provides an example of default system configuration parameters that may be present at the various locations, and what they mean. Note that the amount of memory may vary greatly; the amount shown is the minimum found in a basic system. The ports referenced are those present on the CPU board. This is an example only; actual entries may differ slightly from this table.

Table 10-1 *Default System Configuration Parameters*

F/D = Factory Defined U/D = User Defined N/A = Not Applicable

		Default Entries for Various Configurations				
EEPROM Offset Address	***Function***	***3/50***	***3/60***	***3/75/140 3/160/180***	***3/110***	***3/260 3/280***
0x000-0x003	Test Area	F/D	F/D	F/D	F/D	F/D
0x004-0x00E	Write Count & Checksum	F/D	F/D	F/D	F/D	F/D
0x010-0x013	Last Hardware Update	F/D	F/D	F/D	F/D	F/D
0x014	Installed Memory	0x04	0x04	0x04	0x04	0x08
0x015	Memory Tested	0x04	0x04	0x04	0x04	0x08
0x016	Monitor Screen Size	0x00	0x00	0x00	0x00	0x13
0x017	Watchdog Action	0x00	0x00	0x00	0x00	0x00
0x018	Boot Device: Poll/EEPROM	0x00	0x00	0x00	0x00	0x00
0x019-0x1D	Alt.Boot Device	0x00	0x00	0x00	0x00	0x00
0x01E	Keyboard Type	0x00	0x00	0x00	0x00	0x00
0x01F	Primary Display	0x12	0x12	0x12	0x00	0x12
0x020	Custom/Sun Banner	0x00	0x00	0x00	0x00	0x00
0x021	Keyboard Click	0x00	0x00	0x00	0x00	0x00
0x022-0x026	Diag Boot Device	0x00	0x00	0x00	0x00	0x00
0x028-0x04F	Diag Boot Path	0x00	0x00	0x00	0x00	0x00
0x050	High Res Columns	N/A	0x50	N/A	N/A	0x50
0x051	High Res Rows	N/A	0x22	N/A	N/A	0x22
0x052-0x057	Reserved	N/A	N/A	N/A	N/A	N/A
0x58	U/D or Default Port A Baud Rate	0x00	0x00	0x00	0x00	0x00
0x059-0x05A	Alt Baud Rate	0x00	0x00	0x00	0x00	0x00
0x05B	Port A DTR/RTS	0x00	0x00	0x00	0x00	0x00
0x05C0x05F	Reserved	N/A	N/A	N/A	N/A	N/A
0x060	U/D or Default Port B Baud Rate	0x00	0x00	0x00	0x00	0x00
0x061-0x062	Alt Baud Rate	0x00	0x00	0x00	0x00	0x00
0x063	Port B DTR/RTS	0x00	0x00	0x00	0x00	0x00
0x064-0x067	Reserved	N/A	N/A	N/A	N/A	N/A
0x068-0x0B7	Custom Banner	0x00	0x00	0x00	0x00	0x00
0x0B8	Test Pattern	0x0AA	0x0AA	0x0AA	0x0AA	0x0AA
0x0B9	Test Pattern	0x55	0x55	0x55	0x55	0x55
0x0BC-0x18B	Configuration Blocks	F/D	F/D	F/D	F/D	F/D
0x018C	Key Table Selector	0x00	0x00	0x00	0x00	0x00

Table 10-1 *Default System Configuration Parameters— Continued*

		Default Entries for Various Configurations				
EEPROM Offset Address	*Function*	*3/50*	*3/60*	*3/75/140 3/160/180*	*3/110*	*3/260 3/280*
0x018D	Locale Specifier	F/D	F/D	F/D	F/D	F/D
0x018E	Keyboard ID	F/D	F/D	F/D	F/D	F/D
0x190-0x20F	Lower Case Key Table	0x00	0x00	0x00	0x00	0x00
0x210-0x28F	Upper Case Key Table	0x00	0x00	0x00	0x00	0x00
0x290-0x48F	Custom Logo	0x00	0x00	0x00	0x00	0x00
0x500-0x70A	Write Count & Checksum	F/D	F/D	F/D	F/D	F/D

In this text, the EEPROM locations are described first, in numerical order, with tables that illustrate the result of various parameter entries. At the end of each description, the offset addresses are shown with illustrations of the content of each byte in that range. If, for example, the illustration shows `size` as the content of the first byte, the previous text would contain a table of possible hexadecimal values that `size` represents.

Diagnostic EEPROM

Test Write Area

Four bytes are provided for the EEPROM portion for the CPU Diagnostic. The contents of these locations after the test are meaningless because these four locations are NOT part of the checksum data area. The Diagnostic area write count locations are updated each time these locations are written.

Address[0x000-0x003]:

Diag Test
Diag Test
Diag Test
Diag Test

Diagnostic Area Write Count

These write counters are for the Diagnostic area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is reliability of their correctness.

Address[0x004-0x009]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Diagnostic Area Checksum

Each EEPROM area maintains three identical 8 bit (byte) checksums. These separate checksums are intended to be the same.

Address[0x00C-0x00E]:

Checksum #1
Checksum #2
Checksum #3

Date of Last System Hardware Update

This four byte location contains the date of the last system update. This date is recorded in the same fashion as the Manufacturing date: total seconds since 1 January, 1970.

Address[0x010-0x013]:

Date (Bits 31-24)
Date (Bits 23-16)
Date (Bits 15-8)
Date (Bits 7-0)

Mbytes of installed Memory

This byte contains the total number (in hexadecimal) of Megabytes of memory installed in the system.

Address[0x014]:

Mbytes Installed

Mbytes of Memory to test on Normal Boot

This byte contains the total number (in hexadecimal) of Megabytes of memory that the firmware tests prior to the booting. The firmware ignores this value and tests all of memory if the diagnostic switch is in the DIAG position. All of memory is initialized even if not tested.

Address[0x015]:

Mbytes to Test

Monitor Screen Size

This byte selects the appropriate video screen sizes for the Monitor in the system. The following table illustrates the options:

Size	Definition
0x00	1152x900 Screen (Most Sun-3 Systems)
0x12	1024x1024 Screen
0x13	1600x1280 Screen (Hi Res — Sun-3/200 Series)
0x14	1440x1440 Screen

A hardware change on the CPU Board is necessary to complete a screen size change.

Address[0x016]:

Size

Watchdog Reset Action

This byte selects the appropriate action for the firmware after a Watchdog Reset. The following table illustrates the options:

Action	Definition
0x00	Watchdog Reset will fall into Boot PROM Monitor
0x12	Watchdog Reset will cause a Power-On-Reset

Address[0x017]:

Action

Operating System Boot-Up

This byte selects whether the Boot PROM polls for boot devices in the system or uses an EEPROM selectable boot device for loading the Sun Operating System (SunOS) during a normal boot. If this option is selected the boot device is specified in EEPROM address 0x019-0x01D. The following table illustrates the options:

Boot Device	Definition
0x00	Poll devices for the SunOS operating system (i.e. xy, sd, etc.)
0x12	Use EEPROM specified boot device

Address[0x018]:

Boot Device

Boot Device

These five bytes provide for installation of a command string that will boot the operating system from a specified device when EEPROM address 0x018 is set to 0x12, and the diagnostics switch is set to NORM. The locations are assigned as follows:

Address	Definition
0x019	Default boot device (1st character converted to hex)
0x01A	Default boot device (2nd character converted to hex)
0x01B	Controller number in Hex
0x01C	Unit number in Hex
0x01D	Partition number in Hex

Use the following table to convert these boot devices from ASCII to Hex:

Operating System Boot Device	**Address[0x019]**	**Address[0x01A]**
xy: Xylogics 450/451 Disk	0x78	0x79
xd: Xylogics Disk (7053)	0x78	0x64
sd: SCSI Disk	0x73	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address[0x019-0x01D]:

Device
Device
Controller
Unit
Partition

Keyboard Type

This byte is to signify a NON-SUN keyboard type. It is currently ignored by the Boot PROM.

Address[0x01E]:

Keyboard

Primary Terminal

This byte selects the appropriate device to use as the primary terminal or user interface. The following table illustrates the options:

Terminal	Definition
0x00	Use B/W Monitor (monochrome on-board frame buffer)
0x10	Use Serial Port A
0x11	Use Serial Port B
0x12	Use Color Monitor (CG2, CG3, CG5, or color daughter board)
0x20	Use single or first of multiheaded P4 board

Address[0x01F]:

Terminal

Display Sun Banner

This byte selects whether to display the Sun banner or a custom banner on the screen when booting. The custom banner is defined in a 80-byte character buffer at EEPROM addresses 0x068-0x0B8. See also the paragraphs *Custom Logo Selector* and *Custom Logo* near the end of this chapter for the location and selection of a bit-mapped image that replaces the Sun logo. The following table illustrates the banner options:

Value	Definition
0x00	Display the Sun Banner
0x12	Display Custom Banner

Address[0x020]:

Value

Keyboard Click

This byte selects whether the Sun-3 keyboard should be initialized with its key click option on or off. The following table illustrates the options:

Click	Definition
0x00	Turn click OFF
0x12	Turn click ON

Address[0x021]:

Click

Diagnostic Boot Device

These five bytes define the device that the Boot PROM will use when the Diagnostic switch is ON (in DIAG position). The following table illustrates the boot device specification:

Address	Definition
0x022	Default boot device (1st character in hex)
0x023	Default boot device (2nd character in hex)
0x024	Controller number in Hex
0x025	Unit number in Hex
0x026	Partition number in Hex

Use the following table to convert these boot devices from ASCII to Hex:

Diagnostic Boot Device	Address[0x022]	Address[0x023]
xy: Xylogics 450/451 Disk	0x78	0x79
xd: Xylogics Disk (7053)	0x78	0x64
sd: SCSI Disk	0x73	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address[0x022-0x026]:

Device
Device
Controller
Unit
Partition

Diagnostic Boot Path

These 40 bytes represent a character buffer for a user specified diagnostic path (/stand/diag, for example). These ASCII characters are represented by hex values, terminated with 0x00. You would first open address 0x028 and enter the hexadecimal equivalent of the first character in the selected path, and continue on in that manner, ending with 0x00. An ASCII to Hex conversion chart is included at the back of this manual for your convenience.

Address[0x028-0x04F]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	0x00		

High Res Screen Size

These 2 bytes allow the selection of the number of columns and number of rows for the high resolution monitor.

Address[0x050-0x051]:

# of columns
# of rows

The default entries are 50 for # of columns (the hexadecimal value for 80 columns), and 22 for # of rows (the hexadecimal value for 34 rows).

SCC Port A Default Baud Rate

This byte selects whether SCC Port A will use the default baud rate of 9600 baud or the user specified baud rate defined in the Port A Baud Rate EEPROM address [0x059-0x05A]. The following table illustrates the options:

Value	Definition
0x00	Use Default Baud rate of 9600 Baud
0x12	Use EEPROM defined Baud rate

Address[0x058]:

Value

SCC Port A Baud Rate

These two bytes define the baud rate at which SCC Port A is initialized if EEPROM address[0x058] has been set to 0x012. These bytes are the hexadecimal equivalent of the baud rate. The following table illustrates hexadecimal equivalents to the various baud rates.

Baud Rate	Hex Equivalent	Address[0x059]	Address[0x05A]
300	0x012C	0x01	0x2C
600	0x0258	0x02	0x58
1200	0x04B0	0x04	0xB0
2400	0x0960	0x09	0x60
4800	0x12C0	0x12	0xC0
9600	0x2580	0x25	0x80
19200	0x4B00	0x4B	0x00
38400	0x9600	0x96	0x00

Address[0x059-0x05A]:

Baud (High byte)
Baud (Low byte)

SCC Port A DTR/RTS

This byte selects whether SCC Port A will have the signals DTR and RTS asserted in the initialization process. The following table illustrates the options:

Value	Definition
0x00	Assert the DTR and RTS signals
0x12	Do NOT assert the DTR and RTS signals

Address[0x05B]:

Value

SCC Port B Default Baud Rate

This byte selects whether SCC Port B will use the default baud rate of 9600 baud or the user specified baud rate defined in the Port B Baud Rate EEPROM address[0x061-0x062]. The following table illustrates the options:

Value	Definition
0x00	Use Default Baud rate of 9600 Baud
0x12	Use EEPROM defined Baud rate

Address[0x060]:

Value (0x00 or 0x12)

SCC Port B Baud Rate

These two bytes define the baud rate at which SCC Port B is initialized if EEPROM address 0x060 has been set to 0x012. These bytes are the hexadecimal equivalent of the baud rate. The following table illustrates how to specify the baud rate:

Baud Rate	Hex Equivalent	Address[0x061]	Address[0x062]
300	0x012C	0x01	0x2C
600	0x0258	0x02	0x58
1200	0x04B0	0x04	0xB0
2400	0x0960	0x09	0x60
4800	0x12C0	0x12	0xC0
9600	0x2580	0x25	0x80
19200	0x4B00	0x4B	0x00
38400	0x9600	0x96	0x00

Address[0x061-0x062]:

Baud (High byte)
Baud (Low byte)

SCC Port B DTR/RTS

This byte selects whether SCC Port B will have the signals DTR and RTS asserted in the initialization process. The following table illustrates the options:

Signal	Definition
0x00	Assert the DTR and RTS signals
0x12	Do NOT assert the DTR and RTS signals

Address[0x063]:

Signal

Custom Banner

These 80 bytes represent a character buffer for a user specified custom banner to be displayed instead of the Sun banner, when the value of EEPROM location is 0x020 is 0x012. All locations up to the terminator (0x00) are displayed; each byte not filled with the hexadecimal equivalent of an ASCII character should contain zeroes.

Address[0x068-0x0B7]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	0x00				

Test Pattern

These two bytes are used to provide a known data test pattern to check the EEPROM data lines.

Address[0x0B8-0x0B9]:

0xAA	0x55

System Configuration

There are currently 208 bytes used to represent the hardware configuration of the system. The system configuration is divided into 12 ''slot-configuration'' blocks of 16 bytes each, and 1 sentinel block of 16 bytes to denote the end of the configuration table. The first 16-byte block represents the board specific information in slot 1 of the card cage. The second 16 byte block represents slot 2, and so on.

All empty slots are identified as Board Type Zero (the value 0x00 is in the first byte of that block). The sentinel block **(0xFF)** resides immediately after the configuration block for the last board in the system. The sentinel block should be the 2nd configuration block in a 1-slot system, 4th block for a 3-slot, and so on.

The first byte of each block identifies the type of board that occupies that slot. The information contained in the other bytes of each block may vary, as shown on the following pages.

The layout of the system configuration blocks is illustrated in the following table. B/S means that the contents of that byte are board specific. N/U means ''not used''.

The value in each byte depends on what type of board is in which slot. Each block always begins with the ''Board Type'' value. Logically, the first address, 0x0BC, would begin the CPU board block, because the CPU board is always in Slot 1, meaning that the ''Board Type'' value would be 0x01. ''Board Type'' values are shown in Table 10-3 . Possible configuration block values (to replace B/S) that identify the attributes of various Sun PC boards are presented in the next several pages.

This table shows each offset address in the configuration block, followed with a colon and the type of information represented in that byte. 0x0BC is the first byte of the block representing the board in Slot 1, and, in a 12-slot system, 0x16C is the first byte representing the board in Slot 12. The beginning address for each block is shown in boldfaced type: **0BC**.

"0BD:B/S" means that the value in location 0x0BD is board specific. If you turn to the CPU Board 16-byte representation, you will see that this byte would contain the hexadecimal value for the amount of memory on the CPU board, which will vary.

Table 10-2 *Configuration Block Layout Address[0x0BC-0x18B]*

0BC:Board Type	0BD:B/S	0BE:B/S	0BF:B/S	0C0:B/S	0C1:B/S	0C2:B/S	0C3:B/S
0C4:B/S	0C5:B/S	0C6:B/S	0C7:B/S	0C8:B/S	0C9:B/S	0CA:B/S	0CB:B/S
0CC:Board Type	0CD:B/S	0CE:B/S	0CF:B/S	0D0:B/S	0D1:B/S	0D2:B/S	0D3:B/S
0D4:B/S	0D5:B/S	0D6:B/S	0D7:B/S	0D8:B/S	0D9:B/S	0DA:B/S	0DB:B/S
0DC:Board Type	0DD:B/S	0DE:B/S	0DF:B/S	0E0:B/S	0E1:B/S	0E2:B/S	0E3:B/S
0E4:B/S	0E5:B/S	0E6:B/S	0E7:B/S	0E8:B/S	0E9:B/S	0EA:B/S	0EB:B/S
****0EC**:Board Type	0ED:B/S	0EE:B/S	0EF:B/S	0F0:B/S	0F1:B/S	0F2:B/S	0F3:B/S
0F4:B/S	0F5:B/S	0F6:B/S	0F7:B/S	0F8:B/S	0F9:B/S	0FA:B/S	0FB:B/S
0FC:Board Type	0FD:B/S	0FE:B/S	0FF:B/S	100:B/S	101:B/S	102:B/S	103:B/S
104:B/S	105:B/S	106:B/S	107:B/S	108:B/S	109:B/S	10A:B/S	10B:B/S
10C:Board Type	10D:B/S	10E:B/S	10F:B/S	110:B/S	111:B/S	112:B/S	113:B/S
114:B/S	115:B/S	116:B/S	117:B/S	118:B/S	119:B/S	11A:B/S	11B:B/S
11C:Board Type	11D:B/S	11E:B/S	11F:B/S	120:B/S	121:B/S	122:B/S	123:B/S
124:B/S	125:B/S	126:B/S	127:B/S	128:B/S	129:B/S	12A:B/S	12B:B/S
12C:Board Type	12D:B/S	12E:B/S	12F:B/S	130:B/S	131:B/S	132:B/S	133:B/S
134:B/S	135:B/S	136:B/S	137:B/S	138:B/S	139:B/S	13A:B/S	13B:B/S
13C:Board Type	13D:B/S	13E:B/S	13F:B/S	140:B/S	141:B/S	142:B/S	143:B/S
144:B/S	145:B/S	146:B/S	147:B/S	148:B/S	149:B/S	14A:B/S	14B:B/S
14C:Board Type	14D:B/S	14E:B/S	14F:B/S	150:B/S	151:B/S	152:B/S	153:B/S
154:B/S	155:B/S	156:B/S	157:B/S	158:B/S	159:B/S	15A:B/S	15B:B/S
15C:Board Type	15D:B/S	15E:B/S	15F:B/S	160:B/S	161:B/S	162:B/S	163:B/S
164:B/S	165:B/S	166:B/S	167:B/S	168:B/S	169:B/S	16A:B/S	16B:B/S
16C:Board Type	16D:B/S	16E:B/S	16F:B/S	170:B/S	171:B/S	172:B/S	173:B/S
174:B/S	175:B/S	176:B/S	177:B/S	178:B/S	179:B/S	17A:B/S	17B:B/S
***17C**:FF	17D:N/U	17E:N/U	17F:N/U	180:N/U	181:N/U	182:N/U	183:N/U
184:N/U	185:N/U	186:N/U	187:N/U	188:N/U	189:N/U	18A:N/U	18B:N/U

* This is the Sentinel Block ** Sentinel Block goes here for 3-slot system

Possible "Board Type" values for the first byte of the board configuration blocks are defined in the following table. Note that these are Board Type definitions and not system slot assignments, or addresses. For slot assignments, refer to the appropriate *Cardcage Slot Assignments and Backplane Configuration* document.

The value shown under ***Type*** in the table below is entered in the first byte of the configuration block that represents that board. The beginning EEPROM address of the configuration block depends on the slot assignment (refer to the representation on the previous page).

Table 10-3 *Board Type Values*

Type	***Board Definition***
0x00	None (Empty slot)
0x01	CPU
0x02	Memory
0x03	Color
0x04	Frame Buffer
0x05	FPA
0x06	SMD Disk Controller
0x07	Tape Controller
0x08	Ethernet Controller
0x09	MTI/ALM
0x0A	Graphics Processor (GP)
0x0B	SCP Controller
0x0C	SCSI Controller
0x0D	Integrated Personal Computer (SunIPC)
0x0E	Graphics Board (GB)
0x0F	3/75 SCSI with Memory
0x10	MAPKIT
0x11	Channel Adapter
0x12	ALM-2
0x13	MCP/SCP-2
0x14-0x7F	Reserved Sun Hardware
0x80-0xFE	Reserved Non-Sun Hardware
0xFF	Sentinel Block

Here is an example of what might be present in the configuration blocks for a Sun-3/75 workstation, which has only two slots:

0BC:01	0BD:04	0BE:01	0BF:00	0C0:00	0C1:00	0C2:00	0C3:00
0C4:00	0C5:00	0C6:00	0C7:00	0C8:00	0C9:00	0CA:00	0CB:00
0CC:02	0CD:04	0CE:00	0CF:00	0D0:00	0D1:00	0D2:00	0D3:00
0D4:00	0D5:00	0D6:00	0D7:00	0D8:00	0D9:00	0DA:00	0DB:00
DC:FF	0DD:00	0DE:00	0DF:00	0E0:00	0E1:00	0E2:00	0E3:00
0E4:00	0E5:00	0E6:00	0E7:00	0E8:00	0E9:00	0EA:00	0EB:00

In the example above, the first block begins at EEPROM offset address 0x0BC, and contains the "board type" value 0x01, which means that a CPU board resides in the first slot. The next byte, at location 0x0BD, indicates that the board has 4 Megabytes of main memory. The following byte, at location 0x0BE, contains 0x01 to indicate that the CPU board has an MC68881 Floating Point Coprocessor chip. Refer to the CPU board configuration block layout on the following page for other "Installed Option" values that could be present in location 0x0BE.

The second block, representing slot 2 and beginning with location 0x0CC, contains the "board type" value 0x02 in the first byte, which indicates that a memory board resides there. The 0x04 value in the second byte (location 0xCD) indicates that the memory board has 4 Megabytes of RAM.

The last block, at address 0x0DC, contains the value 0xFF in the first byte, which indicates that this is the sentinel block, meaning that the preceding block represented the last board in the system. The presence of the sentinel block also means that the values in the next 160 bytes of the configuration block are unused and should be ignored.

Configuration Block Examples

Each possible configuration block is graphically represented on the following pages to show the values that must be present in the sixteen bytes to represent the various Sun PC Boards. Each byte is shown as a box, filled with the type of information represented by the value in that byte.

Table 10-4 **No-Board Configuration Block**

1st Byte	0x00
2nd Byte	0x00
3rd Byte	0x00
4th Byte	0x00
5th Byte	0x00
6th Byte	0x00
7th Byte	0x00
8th Byte	0x00
9th Byte	0x00
10th Byte	0x00
11th Byte	0x00
12th Byte	0x00
13th Byte	0x00
14th Byte	0x00
16th Byte	0x00

Use this block to represent an empty slot. For example, assume that the block beginning with EEPROM address 0x0BC (the first block) contains the CPU board configuration values, because the CPU board is in Slot 1. Let us say that Slot 2 is empty. In order to represent this, the second configuration block, beginning with address 0x0CC, would contain 0x00 in each of the next sixteen bytes, up to and including address 0x0DB.

Table 10-5 **CPU Board Configuration Block**

1st Byte	Board Type: 0x01
2nd Byte	# Mbytes Memory in Hex
3rd Byte	Installed Options: Bit 0 = 1 (68881) Bit 1 = 1 (DCP/DES) Bit 2 = 0 Bit 3 = 0 Bit 4 = 0 Bit 5 = 0 Bit 6 = 0 Bit 7 = 0

4th Byte	# Kb of Cache (hex)
	0x0 = Sun-3/50,3/60,3/75,3/1xx,
	0x40 = Sun-3/2xx

5th Byte	Sun 3/60 Color Frame Buffer 0x00 = No Board 0x01 = Hi Res B/W 0x02 = Lo Res Color 0x03 = Hi Res Color
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

How to Program the CPU Configuration Block

Because the CPU board is always in Slot 1, this example will use the actual EEPROM offset addresses of each byte in the CPU board configuration block. If the CPU board were in another slot, the addresses shown would depend on the slot assignment, as shown in Table 10-2 .

However, it is likely that, if you open EEPROM location 0x0BC, the value in the first byte will be 0x01, to represent the CPU board:

```
>q 0bc Return

EEPROM 0BC: 01?  Return
```

The next location (Byte 2) should now be displayed. It represents how much memory is on the CPU board. Suppose the value is 0x04, for 4 Megabytes, and you want to change it to 32. You must convert the decimal number, 32, to hexadecimal and then enter it as follows:

```
EEPROM 0BD: 04 ? 20 Return

EEPROM 0BE:01 ?
```

The monitor program changes the value of of location 0x0BD and then shows you the value in the next location, which represents the third byte in the block, assigned to ''Installed Options''. The 0x01 value indicates that a MC68881 chip is on-board.

To fill in the ''Installed Options'' byte, you must convert the binary value — represented by a one or zero in Bits 0 through 7 — to a hexadecimal value. For example, if the CPU board contains both a MC68881 and a DCP chip, the binary value would be:

```
00000011
```

Converted to hexadecimal, the value in the third byte would be 0x03.

If only a MC68881 is present, the value would be 0x01. If only a DCP chip is present, the binary value would be

```
00000010
```

meaning that the hexadecimal value in the third byte would be 0x02. And, of course, if neither option is present, the value in the third byte would be 0x00.

If you were to change the ''Installed Options'' value, you would simply enter the new value after the question mark, and press Return to view the fourth byte, which represents the quantity of cache memory present on the CPU board.

If, at any time during this process, you want to return to the monitor prompt, enter a non-hexadecimal character after the new value you just entered, BEFORE pressing Return. Refer to the *PROM Monitor Commands* chapter for more information on procedures for modifying memory locations.

In this example, the CPU board did not have a MC68881 Floating Point Coprocessor, and has been upgraded to include that chip. No data encryption chip is present. You would enter:

```
EEPROM 0BE:00 ? 01 Return

EEPROM 0BF:00 ?
```

Now, suppose the CPU board contains cache memory. The fourth byte (location 0x0BF) would probably contain 0x40, meaning there are 64 Kilobytes of on-board cache memory.

As shown in the CPU board configuration example, the fifth byte (location 0x0C0) is reserved for Sun-3/60 configuration information. The values in this byte tell us whether or not a Color Frame Buffer board is attached to the 3/60 CPU board, and whether the board interfaces with a high or low resolution, black and white or color monitor. A 3/60 system with a High Resolution Black and White Monitor would show this value when you increment to the fifth byte:

```
EEPROM 0C0:01 ?
```

Table 10-6 **Memory Board Configuration Block**

1st Byte	0x02
2nd Byte	# Mbytes of Memory in hex
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-7 **Color Board Configuration Block**

1st Byte	0x03
2nd Byte	Type (2, 3 or 5) 0x2 = Sun2 0x3 = Sun3 0x5 = CG5
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-8 **Frame Buffer Board Configuration Block**

1st Byte	0x04
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-9 **FPA Board Configuration Block**

1st Byte	0x05
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-10 **SMD Disk Board Configuration Block**

1st Byte	0x06
2nd Byte	Manufacturer 1 = Xylogics 450 2 = Xylogics 451
3rd Byte	Controller #
4th Byte	# of Disks
5th Byte	Drive #0 Capacity 0 = No Disk 1 = 8" 130 Mb (450/451) 2 = 8" 280 Mb (451) 3 = 10.5" 380 Mb (450/451) 4 = 10.5" 575 Mb (451)
6th Byte	Drive #1 Capacity
7th Byte	Drive #2 Capacity
8th Byte	Drive #3 Capacity
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-11 **1/2 Inch Tape Controller Configuration Block**

1st Byte	0x07
2nd Byte	Manufacturer 1 = Xylogics (472) 2 = Ciprico (TM1000)
3rd Byte	Controller #
4th Byte	# of Tape Drives
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-12 **Second Ethernet Controller Board Configuration Block**

(For the LAN Gateway Interface)

1st Byte	0x08
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-13 **MTI/ALM Board Configuration Block**

1st Byte	0x09
2nd Byte	# Terminals
3rd Byte	Manufacturer 1= Systech 2= Sun Microsystems
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-14 **GP Board Configuration Block**

1st Byte	0x0A
2nd Byte	Type
	1 = GP+ 2 = GP2
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-15 **SCP Board Configuration Block**

1st Byte	0x0B
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-16 **SCSI Board Configuration Block**

1st Byte	0x0C
2nd Byte	Type (2 or 3) 0x2 = Sun2 0x3 = Sun3
3rd Byte	# of Tape Drives
4th Byte	# of Disk Drives
5th Byte	Tape Controller 0x1 = Sysgen 0x2 = MT02
6th Byte	Disk Controller 0x1 = MD21 0x2 = Adaptec
7th Byte	Disk Drive #0 Capacity 0x1 = 5.25" 71 Mb 0x2 = 5.25" 141 Mb 0x3 = 5.25" 327 Mb
8th Byte	Disk Drive #1 Capacity
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-17 **SunIPC Board Configuration Block**

1st Byte	0x0D
2nd Byte	Reserved
3rd Byte	Math Coprocessor Option: 0 = absent 1 = installed
4th Byte	Disk Drive Option * 0 = None 1 = Single 5 1/4" Floppy 2 = Dual 5 1/4" Floppy
5th Byte	Drive 0 Capacity 0 = None 1 = r/w 1.2 MB, read 360 KB 2 = r/w 1.2 MB
6th Byte	Drive 1 Capacity 0 = None 1 = r/w 360 KB
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

NOTE ** The disk drive option can readily be moved to other SunIPC boards and therefore this information may not always be correct.*

Table 10-18 **GB Board Configuration Block**

1st Byte	0x0E
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-19 **3/75 SCSI Memory Board Configuration Block**

1st Byte	0x0F
2nd Byte	# MB of Memory (Hex) 0x00, 0x02 or 0x04
3rd Byte	Type of SCSI 0x02 = Sun-2 0x03 = Sun-3
4th Byte	# of Tape Drives (hex)
5th Byte	# of Disk Drives (hex)
6th Byte	Tape Controller 0x01 = Sysgen 0x02 = MT02
7th Byte	Disk Controller 0x01 = MD21 0x02 = Adaptec
8th Byte	Disk Drive #0 Capacity 0x01 = 5.25" 71 MB 0x02 = 5.25" 141 MB
9th Byte	Disk Drive #2 Capacity
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-20 **MAPKIT Configuration Block**

1st Byte	0x10
2nd Byte	1=INI
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-21 **Channel Adapter Configuration Block**

1st Byte	0x11
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-22 **ALM-2 Configuration Block**

1st Byte	0x12
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-23 **MCP/SCP-2 Configuration Block**

1st Byte	0x13
2nd Byte	Reserved
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 10-24 **Sentinel Block**

1st Byte	0xFF
2nd Byte	Not Used
3rd Byte	Not Used
4th Byte	Not Used
5th Byte	Not Used
6th Byte	Not Used
7th Byte	Not Used
8th Byte	Not Used
9th Byte	Not Used
10th Byte	Not Used
11th Byte	Not Used
12th Byte	Not Used
13th Byte	Not Used
14th Byte	Not Used
15th Byte	Not Used
16th Byte	Not Used

Key Table Selector

The value of this byte is used to determine the appropriate key tables to be used based on the following table. EEPROM key tables refer to the tables entered in EEPROM addresses 0x190-0x20F (lower case) and 0x210-0x28F (upper case).

Key Table Selector	Definition
0x58	Use EEPROM key tables
Other than 0x58	Use PROM Key tables

Address [0x18C]:

Key table selector

EEPROM Locale Specifier

This byte contains the specifier for the locale (country, language, codeset) for which the system is configured. The specifier is provided by the SunOS operating system.

Address [0x18D]:

Locale Specifier

Keyboard ID

The Boot PROM checks the EEPROM Key Table Selector (address 0x18C), and if the value is 0x58, it then compares the value in the address 0x18E with the keyboard type it finds in the system. This byte contains the hard coded keyboard type.

Address [0x18E]:

Keyboard ID

Custom Logo Selector

The value of this byte is used to determine the appropriate Logo bit-map to be displayed upon power-up, based on the following table.

Custom Logo Selector	*Definition*
0x12	Use EEPROM logo bit-map (see Custom Logo, below)
Other than 0x12	Use Sun logo bit-map

Address [0x18F]:

Custom Logo Selector

EEPROM Lower Case Key table

An array of 128 bytes from address 0x190 can be used for a different lower case key table. This table is used by the firmware if location 0x18C is set to 0x58 and Keyboard ID matches the hard coded ID.

Address [0x190-0x20F]:

128 bytes for lower case key table

EEPROM Upper Case Key Table

An array of 128 bytes from address 0x210 can be used for a different upper case key table. This table is used by the firmware if location 0x18C is set to 0x58 and Keyboard ID matches the hard coded ID.

Address [0x210-0x28F]:

128 bytes for upper case key table

Custom Logo

This 64x8 matrix may contain a Custom Logo bit-map that can be selected by setting location 0x18F to 0x12.

Address [0x290-0x48F]:

64X8 bytes Custom Logo

Reserved Area

Write Count

These write counters are for the Reserved area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x500-0x505]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Reserved Area

Checksum

Each EEPROM area will maintain three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x508-0x50A]:

Checksum #1
Checksum #2
Checksum #3

ROM Area

Write Count

These write counters are for the ROM area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x600-0x605]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

ROM Area

Checksum

Each EEPROM area will maintain three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x608-0x60A]:

Checksum #1
Checksum #2
Checksum #3

Software Area

Write Count

These write counters are for the Software area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x700-0x705]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Software Area

Checksum

Each EEPROM area will maintain three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x708-0x70A]:

Checksum #1
Checksum #2
Checksum #3

11

Sun-4 Self-tests and Initialization

Sun-4 Self-tests and Initialization **173**

11.1. The Power-Up Test Sequence 173
11.2. Diagnostic LED Interpretation 180
11.3. Host System Initialization 201
11.4. The Boot Sequence 203

11

Sun-4 Self-tests and Initialization

11.1. The Power-Up Test Sequence

In order to perform the power-up tests, two assumptions must be met. The IU (Integer Unit or CPU) must be functional and the ability to fetch instructions from the EPROMs must be intact.

Powering up a Sun-4 workstation resets the IU to *boot* state, which means that all instruction fetches are forced to the Boot PROMs. Execution of the minimum-confidence power-up tests begin immediately. These tests do not employ any memory until memory has been successfully checked.

The quantity of memory checked during a power-up with the diagnostic switch on NORM is dependent on EEPROM programming. The *EEPROM Layout* chapter explains how to set the parameter that controls the quantity of memory tested. If the diagnostic switch is in the normal position and the workstation contains a large amount of main memory, self-tests may last eight minutes. You should see a banner on the console screen during memory tests, and a rotating diagonal symbol after the `Testing _ megabytes of memory...` message to indicate that the memory tests are in progress. LED five should blink on and off during the normal mode memory test.

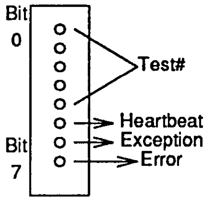
The objective of the power-up test sequence is to determine whether or not the CPU board logic and main memory are functional. Following the successful completion of the power-up tests and subsequent workstation initialization, an attempt is made to boot the SunOS operating system, an EEPROM-specified program or an operator-specified stand-alone program.

If the Diagnostic Switch is in the *NORM* position, the power-up tests execute successfully and you do *not* terminate the default boot sequence, an attempt is made to down-load SunOS software. A display similar to that shown on the following page appears on the *workstation's* screen to indicate that power-up tests were successful.

```
Selftest Completed Successfully.

        Sun Workstation, Model Sun-4/___ Series.
        Type-3 keyboard.
   ◆    ROM Rev __, _ MB memory installed, Serial #_______
        Ethernet address __:__:__:__:__:__

Testing __ megabytes of memory...Completed.

Auto-boot in progress...
Boot: sd(0,0,0)
Boot: sd(0,0,0)vmunix
```

One requirement of Sun-4 firmware is to assign a unique test number to most of the power-up tests and display that number in bits *zero* through *four* of the diagnostic LEDs as the test is running. Given that there are fewer test numbers than there are power-up tests, some power-up tests, which check the same part of hardware, will share a test number.

If one of these power-up tests should fail, bit *seven* of the diagnostic LEDs will also light up. Bit seven will serve as an indicator that there is a hardware problem. The LED display will enable the service person not only to conclude whether or not there is a problem, but he will also be able to determine which type of power-up test is failing.



For the sake of completeness, LED *five* is the *heart beat* LED. After the power-up tests have been completed, but prior to invocation of the SunOS operating system or an EEPROM-specified program, LED 5 will blink on and off to indicate that the IU is actually executing instructions. LED *six* indicates whether or not the failure is an exception (i.e. unexpected trap or unexpected interrupt). The diagram to the left summarizes the LEDs.

Additionally, if the Diagnostic Switch is in the DIAG position, status/error messages from the power-up tests are transmitted to serial ports A and B by way of the MMU by-pass so that they can be viewed on the screen of a "dumb" terminal. Therefore, even though some power-up tests will share a test number, the output on the terminal's screen tells you exactly which test is failing.

In order to view the messages, the terminal characteristics must be set up as follows. The baud rate for serial port A must be *9600* while the baud rate for serial port B must be *1200*. There must be *eight* data bits and *one* stop bit, and parity is turned *off*.

Upon detection of a failure, the unsuccessful test will enter an infinite scope loop. Continuously re-executing the failing test should enhance the service person's ability to study the problem with test equipment.

If the Diagnostic Switch is in the DIAG position, limited interaction with the power-up tests will be allowed. In particular, six commands are available. These commands will only be active prior to the display of the

```
Selftest Completed.
```

or the

```
Testing ...megabytes of memory...Completed.
```

message. Each of these commands is documented below.

1. Press the **`b`** (a mnemonic for *burn-in*) key, prior to the display of the `...memory...Completed` message, to execute the power-up test sequence indefinitely. This option is useful during the manufacturing burn-in stage.
2. Press the **`s`** key prior to the display of the `...Completed` message to *re-start* the power-up test sequence.
3. If one of the power-up tests fails, it will continue to re-execute forever unless interrupted. Press the **space bar** to terminate the failed test and execute the next power-up test.
4. By default, an unsuccessful power-up test will print one error message before entering an infinite scope loop. Once inside of the scope loop, the failing test will *not* print any more messages. If, however, you would like to see test messages while in the scope loop, the **`p`** key should be pressed. It is possible to turn the messages back off by pressing the `p` key a second time. In other words, the `p` command will act like a toggle switch. Pressing `p` will turn message mode on or off.
5. To skip the remaining power-up tests, press [Esc] prior to the `Selftest Completed` message.
6. To continuously re-execute the test that is currently running, press the **`l`** (loop) key.

If the content of location 0x17 of the EEPROM is equal to 0x12 (an arbitrarily chosen value), there is another way to re-start the power-up test sequence. Following the display of the banner, you may press the User Reset Switch to re-start the power-up test sequence. Pressing the User Reset Switch will generate a Watchdog Reset. However, rather than handling the Watchdog Reset in the normal manner, the power-up test sequence is initiated, assuming that location 0x17 of the EEPROM contains a 0x12. The power-up tests can be re-started in this manner independent of the position of the Diagnostic Switch.

If the Diagnostic Switch is in the DIAG position, the name of each power-up test will appear on the screen of a *terminal* attached to serial ports A or B. Following the self-test completion, you will have a *ten* second opportunity to enter the more comprehensive test system by pressing *any* key on a "dumb" terminal keyboard. The chapter titled *The Extended Test System* describes the additional tests.

If you do not press any key within the 10 seconds delay time the Boot PROM program will next display the Sun logo and message on the console:

```
Selftest Completed.

Type a key within 10 seconds to enter Extended Test System(e for echo mode).
```

Remote Testing

If you press **e** on a *dumb terminal* keyboard, during the ten second period, all subsequent output will appear on *both* the console video monitor *and* a terminal attached to Serial Port A or B.

The purpose of this feature is to enable an individual at a remote site, using a terminal attached to a non-local machine by way of a telephone line with modems at each end, and the individual on-site to observe the system's behavior simultaneously. Once a system is in this mode, the local and non-local parties can communicate by employing the # command. Specifically, if the person at the remote site would like to send a message to the person on-site, he or she would simply type the # before typing the actual message. In order to terminate the message, the person at the remote site would enter a second #. At this point, the on-site person would be able to respond by prefixing and terminating their response with the #.

The # option will be very useful in the situation where an on-site customer has contacted a remote repair depot regarding a potential hardware problem. The repair person at the remote repair depot could initiate diagnostics on the non-local machine and allow both parties to observe the output. Beyond that, the repair person and the customer could communicate by way of the # command.

The following two screens represent an example of the Sun-4 Diagnostic self-test display. The test sequence varies slightly between Sun-4/1xx and Sun-4/2xx systems. This example combines both versions, labeling the tests that are only present for a particular system.

```
Boot PROM Selftest.

EPROM Checksum Test.
UDVMA Enable Register Test. (Sun-4/2xx only)
UDVMA Map Write-Write-Read Test. (Sun-4/2xx only)
UDVMA Map Address Test. (Sun-4/2xx only)
UDVMA Map 3-Pattern Test. (Sun-4/2xx only)
Context Register Test.
Segment Map Write-Write-Read Test.
Segment Map Address Test.
Segment Map 3-Pattern Test.
Page Map Write-Write-Read Test.
Page Map Address Test.
Page Map 3-Pattern Test.
Software Traps Test.
Interrupt Register Test.
Software Interrupts Test.
TOD Interrupt Test.
Video Memory Write-Write-Read Test.
Video Memory Address Test.
Video Memory 3-Pattern Test.
P4 Color Map Test.  (Sun-4/1xx Only)
Limited Memory Test:Megabyte 0x0000000N. Tests Complete.
```

Continued on the following page

Sun-4 Diagnostic Self Tests, Continued

```
MMU Read Access/Modified Bits Test.
MMU Write Access/Modified Bits Test.
MMU Write To Protected Page Test.
MMU Read Not-Writable Invalid Page Test.
MMU Read Writable Invalid Page Test.
MMU Write Not-Writable Invalid Page Test.
MMU Write Writable Invalid Page Test.
Main Memory Timeout Test.
Control Space Timeout Test.
Range Error Test.
Size Error Test.
Parity No Fault Test. (Sun-4/1xx Only)
Parity Error Detection and Interrupt Test. (Sun-4/1xx Only)
Parity Error Detection without Interrupt. (Sun-4/1xx Only)
ECC Circuit Tests. (Sun-4/2xx Only)
Cache Tag Bank Test, Alternating Bit Pairs <31:11> (Sun-4/1xx Only)
Cache Tag Memory Write-Write-Read Test. (Sun-4/2xx Only)
Cache Tag Memory Address Test. (Sun-4/2xx Only)
Cache Tag Memory 3-Pattern Test. (Sun-4/2xx Only)
Cache Data Memory Write-Write-Read Test. (Sun-4/2xx Only)
Cache Data Memory Address Test. (Sun-4/2xx Only)
Cache Data Memory 3-Pattern Test. (Sun-4/2xx Only)
Cache Write/Read Hit/Miss Verify Test. (Sun-4/2xx Only)
Cache Write/Read/Flush/Verify Test. (Sun-4/2xx Only)
DMA Address Register Test. (Sun-4/1xx Only)
DMA Byte Counter Register Test. (Sun-4/1xx Only)
DMA Send and Byte Pack Register Test. (Sun-4/1xx Only)
DMA Conflict Interrupt Check. (Sun-4/1xx Only)
DMA From Invalid Page. (Sun-4/1xx Only)
Sizing memory (size = 8 Megabytes).
Main Memory Test:Megabyte 0x0000000N.Tests Complete.

Selftest Completed Successfully.

Type a key within ten seconds to enter Extended Test System(e for echo mode).
```

If the diagnostic switch is in the DIAG position, the power-up tests executed successfully and you did *not* enter a character on a *terminal* keyboard within the initial ten second period, the display below will appear on the *workstation* screen. Note that the you have *another ten-second* opportunity to invoke the extended tests. If you do not respond, the program tries to boot a program specified in locations 0x028-0x04F of the EEPROM. If this attempt fails, the monitor is invoked. The tables that follow this display summarize self-test LED interpretation.

```
Selftest Completed Successfully.

       Sun Workstation, Model Sun-4/xxx Series.
       Type-3 keyboard
       ROM Rev __, _ MB memory installed, Serial #____
       Ethernet address __:__:__:__:__:__

Type a key within ten seconds to enter Extended Test System.
```

Figure 11-1 *Sun-4/2xx LED Interpretation*

LEDs ● = ON, ○ = OFF 0 7	*What the System is Doing When These LEDs Are Cycling*	*What Might Be Bad If This Indication Stays On And LED 6/7 Lights*
● ● ● ● ● ● ● ●	A reset sets LEDs to this state	CPU or PROMs bad or +5VDC is low
● ○ ○ ○ ○ ○ ○ ○	Test 0x01 checks PROM checksum	CPU Board (Boot PROM)
○ ● ○ ○ ○ ○ ○ ○	Test 0x02 checks UDVMA enable register	CPU Board
● ● ○ ○ ○ ○ ○ ○	Test 0x03 checks UDVMA map	CPU Board
○ ○ ● ○ ○ ○ ○ ○	Test 0x04 checks the context register	CPU Board (MMU)
● ○ ● ○ ○ ○ ○ ○	Test 0x05, performs segment map tests	CPU Board (MMU)
○ ● ● ○ ○ ○ ○ ○	Test 0x06 checks page map RAM	CPU Board (MMU)
● ● ● ○ ○ ○ ○ ○	Test 0x07 performs software traps test	CPU Board (IU)
○ ○ ○ ● ○ ○ ○ ○	Test 0x08 performs interrupt register test	CPU Board
● ○ ○ ● ○ ○ ○ ○	Test 0x09 performs software interrupts test	CPU Board
○ ● ○ ● ○ ○ ○ ○	Test 0x0A performs TOD clock interrupt test	CPU Board
● ● ○ ● ○ ○ ○ ○	Test 0x0B checks video memory	CPU Board
○ ○ ● ● ○ ○ ○ ○	Test 0x0C performs limited main memory tests	CPU or Memory Board
● ○ ● ● ○ ○ ○ ○	Test 0x0D performs MMU read/write tests	CPU Board (MMU)
○ ● ● ● ○ ○ ○ ○	Test 0x0E performs MMU write to protected page test	CPU Board (MMU)
● ● ● ● ○ ○ ○ ○	Test 0x0F performs MMU read invalid page tests	CPU Board (MMU)
○ ○ ○ ○ ● ○ ○ ○	Test 0x10 performs MMU write invalid page test	CPU Board (MMU)
● ○ ○ ○ ● ○ ○ ○	Test 0x11 performs main memory timeout test	Memory Board
○ ● ○ ○ ● ○ ○ ○	Test 0x12 performs control space timeout test	CPU Board
● ● ○ ○ ● ○ ○ ○	Test 0x13 performs range error test	CPU Board
○ ○ ● ○ ● ○ ○ ○	Test 0x14 performs size error test	CPU Board
● ○ ● ○ ● ○ ○ ○	Test 0x15 tests ECC circuits	Memory Board
○ ● ● ○ ● ○ ○ ○	Test 0x16 tests cache tag memory	CPU Board
● ● ● ○ ● ○ ○ ○	Test 0x17 tests cache data memory	CPU Board
○ ○ ○ ● ● ○ ○ ○	Test 0x18 is cache write/read hit/miss verify test	CPU Board
● ○ ○ ● ● ○ ○ ○	Test 0x19 is cache write/read/flush verify test	CPU Board
○ ● ○ ● ● ○ ○ ○	Test 0x1A runs main memory tests	Memory Board
○ ○ ○ ○ ○ ○ ○ ●	Self-tests have found an error	CPU or Memory Board
○ ○ ○ ○ ○ ○ ● ○	An exception class error is found	CPU or Memory Board
○ ○ ○ ○ ○ ● ○ ○	Self-tests done, UNIX in boot state or monitor quiescent (LED is blinking)	CPU or Memory Board
●⇒ ○⇒ ○⇒ ○ ○⇐ ○⇐ ○⇐ ●	"converging/diverging" pattern	UNIX running okay

Figure 11-2 *Sun-4/1xx LED Interpretation*

LEDs ● = ON, ○ = OFF 0 7	*What the System is Doing When These LEDs Are Cycling*	*What Might Be Bad If This Indication Stays On And LED 6/7 Lights*
● ● ● ● ● ● ● ●	A reset sets LEDs to this state	CPU/PROMs bad or +5V low
● ○ ○ ○ ○ ○ ○ ○	Test 0x01 checks PROM checksum	CPU Board (Boot PROM)
○ ● ○ ○ ○ ○ ○ ○	Test 0x02 checks the context register	CPU Board
● ● ○ ○ ○ ○ ○ ○	Test 0x03 performs segment map tests	CPU Board
○ ○ ● ○ ○ ○ ○ ○	Test 0x04 checks page map RAM	CPU Board
● ○ ● ○ ○ ○ ○ ○	Test 0x05, performs software traps tests	CPU Board (MMU)
○ ● ● ○ ○ ○ ○ ○	Test 0x06 performs interrupt register test	CPU Board (MMU)
● ● ● ○ ○ ○ ○ ○	Test 0x07 performs software interrupts tests	CPU Board (IU)
○ ○ ○ ● ○ ○ ○ ○	Test 0x08 performs TOD clock interrupt test	CPU Board
● ○ ○ ● ○ ○ ○ ○	Test 0x09 checks video memory	CPU Board
○ ● ○ ● ○ ○ ○ ○	Test 0x0A performs P4 color map test	P4 Board
● ● ○ ● ○ ○ ○ ○	Test 0x0B runs limited main memory tests	CPU Board
○ ○ ● ● ○ ○ ○ ○	Test 0x0C performs MMU read/write tests	CPU Board
● ○ ● ● ○ ○ ○ ○	Test 0x0D = MMU write to protected page test	CPU Board (MMU)
○ ● ● ● ○ ○ ○ ○	Test 0x0E performs MMU read invalid page test	CPU Board (MMU)
● ● ● ● ○ ○ ○ ○	Test 0x0F performs MMU write invalid page tests	CPU Board (MMU)
○ ○ ○ ○ ● ○ ○ ○	Test 0x10 performs main memory and control space timeout tests	CPU Board (MMU)
● ○ ○ ○ ● ○ ○ ○	Test 0x11 performs range and size error tests	CPU Board
○ ● ○ ○ ● ○ ○ ○	Test 0x12 performs parity circuit test	CPU Board
● ● ○ ○ ● ○ ○ ○	Test 0x13 tests cache tag memory	CPU Board
○ ○ ● ○ ● ○ ○ ○	Test 0x14 performs SCSI/DMA tests	CPU Board
● ○ ● ○ ● ○ ○ ○	Test 0x15 runs main memory tests	CPU Board
○ ○ ○ ○ ○ ○ ○ ●	Self-tests have found an error	CPU Board
○ ○ ○ ○ ○ ○ ● ○	An exception class error is found	CPU Board
○ ○ ○ ○ ○ ● ○ ○	Self-tests done, UNIX in boot state or monitor quiescent (LED is blinking)	CPU Board
●⇒ ○⇒ ○⇒ ○ ○⇐ ○⇐ ○⇐ ●	"converging/diverging" pattern	UNIX running okay

11.2. Diagnostic LED Interpretation

The paragraphs that follow describe each power-up test. The LEDs would appear as shown in the *Sun-4 LED Interpretation* figures on the previous pages (bit 0 on the left) if the CPU board were installed horizontally, with the component side facing upward. They appear with Bit 0 at the top when the CPU board is installed vertically in the system. The LED read-outs shown on the following pages are represented as you would normally read binary or hexadecimal values, with the least significant bit on the right. If you watch the LEDs on the edge of the CPU board AS SOON AS YOU POWER-UP, you will notice that Bit 0 is the first to light. The Sun-4/100 and Sun-4/200 series workstations share many of the same power-up tests, but number them differently. These descriptions will show both readouts when the LED code differs between the two workstations for the same test.

Diagnostic Register Test

This test sequentially lights up each of the LEDs, beginning with the Bit 0 and ending with the Bit 7. It indirectly tests the IU's ability to fetch instructions from the Boot PROM and transfer data across the data bus. This test does *not* have a unique test number associated with it.

Boot PROM Checksum Test

Test number *1* calculates the checksum of each of the four Boot EPROMs and compares the calculated value with the expected value given in each Boot PROM. Should this test fail, one or more of the Boot PROMs will have to be replaced. The diagram below summarizes the Test 1 LED states.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
1	0x01	bit 7 ooooooo• bit 0	okay
1	0x81	•oooooo•	error

UDVMA Enable Register Test — Sun-4/2xx Only

Test number 2 performs write-read-compare cycles on the Sun-4/2xx User DVMA Enable Register, using patterns "0xff", "0xfe", ..., "0x00".

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
2	0x02	bit 7 oooooo•o bit 0	okay
2	0x82	•ooooo•o	error

UDVMA Map Write-Write-Read Test — Sun-4/2xx only

Test number *3* performs write-write-read-compare-read-compare cycles on each pair of addresses within the Sun-4/2xx User DVMA Map. For each pair of addresses within the User DVMA Map, pattern 0x5a is written to address *x-1*, pattern 0xa5 is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value. The next pair of User DVMA Map addresses that is tested is address *x-2* and address *x-1*. In other words, except for the first and last User DVMA Map addresses, all locations are tested twice.

The write-read-compare operation prevents the test from erroneously concluding that the memory capability of the User DVMA Map is intact when the chip is, in fact, *not* present. If the chip is not present and the write-read-compare operation is not performed, the initial value would remain on the bus and the comparison operation would pass.

If an error should be detected, the test continues to loop on the pair of User DVMA Map addresses currently under test.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
3	0x03	bit 7 ○○○○○○●● bit 0	okay
3	0x83	●○○○○○●●	error

UDVMA Map Address Test — Sun-4/2xx Only

The UDVMA Map Address Test writes the address of each Sun-4/2xx UDVMA Map location to that location before reading back the value and comparing the observed value with the expected value. The entire range of UDVMA Map addresses will be written prior to the read and compare operations.

Again, due to the lack of unique LED Test Numbers, the same LED display used by the UDVMA Map Write-Write-Read Test is used by the UDVMA Map Address Test.

If an error is detected, the test continues to loop on the entire range of UDVMA Map addresses.

UDVMA Map 3-Pattern Test — Sun-4/2xx Only

The UDVMA Map 3-Pattern Test performs three separate write-read-compare cycles on each address within the Sun-4/2xx User DVMA Map. In pass *one*, patterns "0x5a", "0x97" and "0x2c" are repeated throughout the User DVMA Map before each address is read back and the observed value is compared with the expected value. Pass *two* employs patterns "0x97", "0x2c" and "0x5a". Finally, pass *three* will use patterns "0x2c", "0x5a" and "0x97".

The same LED display used by the UDVMA Map Write-Write-Read Test is used by the UDVMA Map 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of UDVMA Map addresses.

Context Register Test

This is Test 4 for Sun-4/2xx and Test 2 for Sun-4/1xx. It performs write-read-compare cycles on the Context Register, using patterns "0xff", "0xfe", ..., "0x00".

Sun-4/2xx LED states:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
4	0x04	bit 7 ooooo•oo bit 0	okay
4	0x84	•oooo•oo	error

Sun-4/1xx LED states:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
2	0x02	bit 7 oooooo•o bit 0	okay
2	0x82	•ooooo•o	error

Segment Map Write-Write-Read Test

Test 5 for Sun-4/2xx and Test 3 for Sun-4/1xx performs write-write-read-compare-read-compare cycles on each pair of addresses within the Segment Map RAM. For each pair of addresses within the Segment Map RAM, pattern "0x12c5a" is written to address *x-1*, pattern "0x0d3a5" is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error should be detected, the test will continue to loop on the pair of Segment Map addresses currently under test.

Sun-4/2xx LED states:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
5	0x05	bit 7 ooooo•o• bit 0	okay
5	0x85	•oooo•o•	error

Sun-4/1xx LED states:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
3	0x03	bit 7 oooooo•• bit 0	okay
3	0x83	•ooooo••	error

Segment Map Address Test

The Segment Map Address Test will write the address of each Segment Map location to that location before reading back the value and comparing the observed value with the expected value. The entire range of Segment Map addresses is written prior to the read and compare operations.

The same LED displays used by the Segment Map Write-Write-Read Test are used by the Segment Map Address Test.

If an error should be detected, the test will continue to loop on the entire range of Segment Map addresses.

Segment Map 3-Pattern Test

The Segment Map 3-Pattern Test will perform three separate write-read-compare cycles on each address within the Segment Map. In pass *one*, patterns "0x12c5a", "0x05a97" and "0x0972c" is repeated throughout the Segment Map before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns "0x05a97", "0x0972c" and "0x12c5a". Finally, pass *three* will use patterns "0x0972c", "0x12c5a" and "0x05a97".

The same LED displays used by the Segment Map Write-Write-Read Test are used by the Segment Map 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Segment Map addresses.

Page Map Write-Write-Read Test

Test 6 on a Sun-4/2xx or Test 4 on a Sun-4/1xx performs write-write-read-compare-read-compare cycles on each pair of addresses within the Page Map RAM. For each pair of addresses within the Page Map RAM, pattern "0x5a972c5a" is written to address *x-1*, pattern "0xa568d3a5" is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error should be detected, the test will continue to loop on the pair of Page Map addresses currently under test.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
6	0x06	bit 7 ooooo●●o bit 0	okay
6	0x86	●oooo●●o	error

Sun-4/1xx LED States:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
4	0x04	bit 7 ooooo●oo bit 0	okay
4	0x84	●oooo●oo	error

Page Map Address Test

The Page Map Address Test will write the address of each Page Map location to that location before reading back the value and comparing the observed value with the expected value. The entire range of Page Map addresses will be written prior to the read and compare operations.

The same LED displays used by the Page Map Write-Write-Read Test are used by the Page Map Address Test.

If an error should be detected, the test will continue to loop on the entire range of Page Map addresses.

Page Map 3-Pattern Test

The Page Map 3-Pattern Test will perform three separate write-read-compare cycles on each address within the Page Map. In pass *one*, patterns ''0x5a972c5a'', ''0x972c5a97'' and ''0x2c5a972c'' is repeated throughout the Page Map before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns ''0x972c5a97'', ''0x2c5a972c'' and ''0x5a972c5a''. Finally, pass *three* will use patterns ''0x2c5a972c'', ''0x5a972c5a'' and ''0x972c5a97''.

The same LED displays used by the Page Map Write-Write-Read Test are used by the Page Map 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Page Map addresses.

Software Traps Test

Test 7 for a Sun-4/2xx or Test 5 for a Sun-4/1xx generates all possible software traps, levels ''0x80'' through ''0xd0'' inclusive, and verify that each trap was taken (occurred).

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
7	0x07	bit 7 ○○○○○●●● bit 0	okay
7	0x87	●○○○○●●●	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
5	0x05	bit 7 ○○○○○●○● bit 0	okay
5	0x85	●○○○○●○●	error

Interrupt Register Test

Test 8 on a Sun-4/2xx or Test 6 on a Sun-4/1xx performs write-read-compare cycles on the Interrupt Register, using patterns "0xff", "0xfe", ..., "0x00".

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
8	0x08	oooo●ooo	okay
8	0x88	●ooo●ooo	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
6	0x06	bit 7 ooooo●●o bit 0	okay
6	0x86	●oooo●●o	error

Software Interrupts Test

Test 9 on a Sun-4/2xx or Test 7 on a Sun-4/1xx generates all three software interrupts, levels "0x1", "0x4" and "0x6", an verify the occurrence of each interrupt.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
9	0x09	bit 7 oooo●oo● bit 0	okay
9	0x89	●ooo●oo●	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
7	0x07	bit 7 ooooo●●● bit 0	okay
7	0x87	●oooo●●●	error

TOD Interrupt Test

After the appropriate initialization, Test 10 on a Sun-4/2xx or Test 8 on a Sun-4/1xx determines whether or not TOD Clock interrupts are occurring.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
10	0x0a	oooo•o•o	okay
10	0x8a	•ooo•o•o	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
8	0x08	oooo•ooo	okay
8	0x88	•ooo•ooo	error

Video Memory Write-Write-Read Test

Test 11 on a Sun-4/2xx or Test 9 on a Sun-4/1xx performs write-write-read-compare-read-compare cycles on each pair of addresses within the Video Memory. For each pair of addresses within the Video Memory, pattern "0x5a972c5a" is written to address *x-1*, pattern "0xa568d3a5" is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error should be detected, the test will continue to loop on the pair of Video Memory addresses currently under test.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
11	0x0b	bit 7 oooo•o•• bit 0	okay
11	0x8b	•ooo•o••	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
9	0x09	bit 7 oooo•oo• bit 0	okay
9	0x89	•ooo•oo•	error

Video Memory Address Test

The Video Memory Address Test will write the address of each Video Memory location to that location before reading back the value and comparing the observed value with the expected value. The entire range of Video Memory addresses is written prior to the read and compare operations.

The same LED displays used by the Video Memory Write-Write-Read Test are used by the Video Memory Address Test.

If an error should be detected, the test will continue to loop on the entire range of Video Memory addresses.

Video Memory 3-Pattern Test

The Video Memory 3-Pattern Test will perform three separate write-read-compare cycles on each address within the Video Memory. In pass *one*, patterns "0x5a972c5a", "0x972c5a97" and "0x2c5a972c" is repeated throughout the Video Memory before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns "0x972c5a97", "0x2c5a972c" and "0x5a972c5a". Finally, pass *three* will use patterns "0x2c5a972c", "0x5a972c5a" and "0x972c5a97".

The same LED display used by the Video Memory Write-Write-Read Test are used by the Video Memory 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Video Memory addresses.

P4 Color Map Test — Sun-4/1xx only

Test 10 on a Sun-4/1xx performs a color map test on cards attached to the P4 bus. It checks the color map on the Brooktree DAC by writing a pattern to the DAC and reading it back to compare. The pattern written causes a shaded gray pattern to appear on the screen. In the event of an error the test loops, reading the color map entry that is in error. The LED states are as follows:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
10	0x0a	○○○○●○●○	okay
10	0x8a	●○○○●○●○	error

Limited Main Memory Write-Write-Read Test

Test 12 on a Sun-4/2xx or Test 11 on a Sun-4/1xx performs write-write-read-compare-read-compare cycles on each pair of addresses within the Main Memory reserved for the monitor. For each pair of addresses within the Main Memory required by the monitor, pattern "0x5a972c5a" is written to address *x-1*, pattern "0xa568d3a5" is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error should be detected, the test will continue to loop on the pair of Main Memory addresses currently under test.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
12	0x0c	bit 7 oooo●●oo bit 0	okay
12	0x8c	●ooo●●oo	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
11	0x0b	bit 7 oooo●o●● bit 0	okay
11	0x8b	●ooo●o●●	error

Limited Main Memory Address Test

The Limited Main Memory Address Test will write the address of each Main Memory location reserved by the monitor, to that location before reading back the value and comparing the observed value with the expected value. The entire range of Main Memory addresses required by the monitor is written prior to the read and compare operations.

The same LED displays used by the Limited Main Memory Write-Write-Read Test are used by the Limited Main Memory Address Test.

If an error should be detected, the test will continue to loop on the entire range of Main Memory addresses used by the monitor.

Limited Main Memory 3-Pattern Test

The Limited Main Memory 3-Pattern Test will perform three separate write-read-compare cycles on each address within the Main Memory reserved for the monitor. In pass *one*, patterns 0x5a972c5a, 0x972c5a97 and 0x2c5a972c are repeated throughout the Main Memory required by the monitor before each address is read back and the observed value is compared with the expected value. Pass *two* employs patterns 0x972c5a97, 0x2c5a972c and 0x5a972c5a. Finally, pass *three* will use patterns 0x2c5a972c, 0x5a972c5a and 0x972c5a97.

The same LED displays used by the Limited Main Memory Write-Write-Read Test are used by the Limited Main Memory 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Main Memory addresses used by the monitor.

MMU Read Access/Modified Bits Test

Test 13 on a Sun-4/2xx or Test 12 on a Sun-4/1xx verifies that a read access sets the Access Status Bit but, does not set the Modified Bit, of the referenced page.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
13	0x0d	bit 7 oooo••o• bit 0	okay
13	0x8d	•ooo••o•	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
12	0x0c	bit 7 oooo••oo bit 0	okay
12	0x8c	•ooo••oo	error

MMU Write Access/Modified Bits Test

The MMU Write Access/Modified Bits Test will verify that a write access sets both the Access Status Bit and the Modified Bit of the referenced page.

The MMU Write Access/Modified Bits Test uses the same LED display as the MMU Read Access/Modified Bits Test.

MMU Write To Write Protected Page Test

Test 14 on a Sun-4/2xx or Test 13 on a Sun-4/1xx determines whether or not an attempt to write to a write protected page will generate a Data Access Trap and set the Protect Error Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
14	0x0e	bit 7 oooo•••o bit 0	okay
14	0x8e	•ooo•••o	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
13	0x0d	bit 7 oooo••o• bit 0	okay
13	0x8d	•ooo••o•	error

MMU Read Not-Writable Invalid Page Test

Test 15 on a Sun-4/2xx or Test 14 on a Sun-4/1xx determines whether or not an attempt to read a write protected, invalid page will generate a Data Access Trap and set the Invalid Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
15	0x0f	bit 7 ○○○○●●●● bit 0	okay
15	0x8f	●○○○●●●●	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
14	0x0e	bit 7 ○○○○●●●○ bit 0	okay
14	0x8e	●○○○●●●○	error

MMU Read Writable Invalid Page Test

The MMU Read Writable Invalid Page Test will determine whether or not an attempt to read a writable, invalid page will generate a Data Access Trap and set the Invalid Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

The MMU Read Writable Invalid Page Test uses the same LED displays as the MMU Read Not-Writable Invalid Page Test.

MMU Write Not-Writable Invalid Page Test

Test 16 on a Sun-4/2xx or Test 15 on a Sun-4/1xx determines whether or not an attempt to write a write protected, invalid page will generate a Data Access Trap and set the Invalid Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
16	0x10	bit 7 ○○○●○○○○ bit 0	okay
16	0x90	●○○●○○○○	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
15	0x0f	bit 7 ○○○○●●●● bit 0	okay
15	0x8f	●○○○●●●●	error

MMU Write Writable Invalid Page Test

The MMU Write Writable Invalid Page Test will determine whether or not an attempt to write a writable, invalid page will generate a Data Access Trap and set the Invalid Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

The MMU Write Writable Invalid Page Test will use the same LED displays as the MMU Write Not-Writable Invalid Page Test.

Main Memory Timeout Test

Test 17 on a Sun-4/2xx or Test 16 on a Sun-4/1xx determines whether or not an attempt to access a non-existent physical address will generate a Data Access Trap and set the Timeout Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
17	0x11	bit 7 ooo•ooo• bit 0	okay
17	0x91	•oo•ooo•	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
16	0x10	bit 7 ooo•oooo bit 0	okay
16	0x90	•oo•oooo	error

Control Space Timeout Test

Test 18 on a Sun-4/2xx determines whether or not an attempt to access a non-existent device address within Control Space will generate a Data Access Trap and set the Timeout Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register. On a Sun-4/1xx this test shares the same LED states as the Main Memory Timeout Tests.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
18	0x12	bit 7 ooo•oo•o bit 0	okay
18	0x92	•oo•oo•o	error

Range Error Test

Test 19 on a Sun-2/xx or Test 17 on a Sun-4/1xx determines whether or not an attempt to access an invalid virtual address (0x88880000) will generate a Data Access Trap and set the Invalid Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
19	0x13	bit 7 ooo•oo•• bit 0	okay
19	0x93	•oo•oo••	error

Sun-4/1xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
17	0x11	bit 7 ooo•ooo• bit 0	okay
17	0x91	•oo•ooo•	error

Size Error Test

Test 20 on a Sun-4/2xx determines whether or not an attempt to access the 8-bit Bus Error Register as if it were a 32-bit register will generate a Data Access Trap and set the Size Error Bit in the Bus Error Register before verifying that the correct address was stored in the Memory Error Address Register. On a Sun-4/1xx, this test shares the same LED states as the Range Error Test.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
20	0x14	bit 7 ooo•o•oo bit 0	okay
20	0x94	•oo•o•oo	error

NOTE *When Sun-4/1xx systems detect a memory error during the power-up self tests, the faulty SIMM will be identified by its reference designator.*

Parity No Fault Test — Sun-4/1xx Only

Test 18 on a Sun-4/1xx performs the Parity No Fault Test. It ensures that parity errors are not generated randomly while accessing memory. With Parity enabled, Parity interrupts ON and Parity test bit OFF (in the Memory Error Control Register) a short section of memory is written and read. A check is made after each read that no parity error interrupt was generated and no error is detected. The LED states are as follows:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
18	0x12	bit 7 ooo•oo•o bit 0	okay
18	0x92	•oo•oo•o	error

Parity Error Detection and Interrupt Test — Sun-4/1xx Only

This Sun-4/1xx test shares LED states with the Parity No Fault Test. It ensures that when a parity error is generated an interrupt will occur. The parity test bit is enabled to force inverse parity bits. A short section of memory is written. That section is read and, after each read a check is made that the parity error is detected and that a parity error interrupt is generated. On an error, the test loops on the location that is not generating parity errors. The LED states are the same as those for the Parity No Fault Test.

Parity Error Detection Without Interrupt — Sun-4/1xx

This test ensures that Sun-4/1xx parity errors are detected without an interrupt having to be generated. With parity interrupts OFF, the parity test bit ON, and parity enabled, writes and reads are done and a check is done to ensure that parity errors are detected but no interrupt is generated. On an error the test loops on the location that is not generating parity errors. This Sun-4/1xx test shares LED states with the Parity No Fault Test.

Cache Tag Bank Test, Alternating Bit Pairs <31:11> — Sun-4/1xx Only

Test 19 on a Sun-4/1xx performs the Cache Tag Bank Test, alternating bit pairs 31:11. This test checks the functionality of the Cobra cache tag IC by accessing memory in pairs where the pairs differ in address by alternating bits 31:11. This should check the tag IC's ability to rapidly switch banks of memory and make hits and misses. Two values are written to a high and low address. Each is read back and compared to the original value written to that address. If a mis-match occurs, an error is indicated. If an error is detected the routine loops repeating the access. The LED states are as follows:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
19	0x13	bit 7 ooo•oo•• bit 0	okay
19	0x93	•oo•oo••	error

SCSI/DMA Tests — Sun-4/1xx Only

Test 20 on the Sun-4/1xx performs SCSI and DMA tests. These tests share these LED states:

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
20	0x14	bit 7 ooo•o•oo bit 0	okay
20	0x94	•oo•o•oo	error

DMA Address Register Test — Sun-4/1xx Only

This test checks the DMA address register in the array.SCSIgate It writes 256 different values to the DMA address register and reads them back. On each read it compares what was written with what was read back. On an error it loops, repeating the operation. The LED states are shown above under *SCSI/DMA Tests*.

DMA Byte Counter Register Test — Sun-4/1xx Only

This test checks the DMA byte count register in the SCSI gate array. It writes 256 different values to the DMA byte count register and reads them back. On each read it compares what was written with what was read back. On an error it loops repeating the operation. The LED states are shown above under *SCSI/DMA Tests*.

DMA Send and Byte Pack Register Test — Sun-4/1xx Only

This test checks DMA transfer into the byte pack register. It maps in a piece of memory and puts a test pattern into it. It then sets up and starts a DMA transfer to the SCSI which should put the test pattern bytes into the byte pack register. The byte pack register is then read and the results compared against the test pattern. The LED states are shown above under *SCSI/DMA Tests*.

DMA Conflict Interrupt Check — Sun-4/1xx Only

This test checks that a DMA conflict interrupt is generated when one of the SCSI registers is accessed during a DMA transfer. It first sets up a transfer with the mapping used in the previous test. While the transfer is in progress it touches the DMA count register and then checks to see that the conflict interrupt occurred. Possible error messages viewed on a terminal attached to a serial port are:

```
The shoebox is not on
```

which means that the cable is plugged into a mass storage subsystem that is not powered up.

```
No SCSI Error Interrupt when expected
```

This message comes up when no interrupt is seen.

```
Error: DMA_CONFLICT bit not set in DMA Status Register
```

This message comes up when no DMA conflict interrupt is seen.

DMA From Invalid Page — Sun-4/1xx Only

This test checks to see that DMA from an invalid page produces a bus error. This is accomplished by starting a transfer to an invalid page and checking to see that the bus error bit is set in the SCSI control/status register. If an error is detected, the routing loops, repeating the operation.

ECC Circuit Test — Sun-4/2xx Only

Test 21 on a Sun-4/2xx actually performs two ECC logic tests. The initial ECC Circuit Test determines whether or not single bit, Correctable Errors can be detected and corrected. The second ECC Circuit Test determines whether or not multiple bit, Uncorrectable Errors are detected and not corrected.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
21	0x15	bit 7 ooo●o●o●bit 0	okay
21	0x95	●oo●o●o●	error

Cache Tag Memory Write-Write-Read Test — Sun-4/2xx Only

Test 22 performs write-write-read-compare-read-compare cycles on each pair of addresses within the Sun-4/2xx Cache Tag RAM. For each pair of addresses within the Cache Tag RAM, pattern 0x5a972c5a is written to address *x-1*, pattern 0xa568d3a5 is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error should be detected, the test will continue to loop on the pair of Cache Tag addresses currently under test.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
22	0x16	bit 7 ooo●o●●obit 0	okay
22	0x96	●oo●o●●o	error

Cache Tag Memory Address Test — Sun-4/2xx Only

The Sun-4/2xx Cache Tag Memory Address Test writes the address of each Cache Tag Memory location to that location before reading back the value and comparing the observed value with the expected value. The entire range of Cache Tag Memory addresses is written prior to the read and compare operations.

The same LED display used by the Cache Tag Memory Write-Write-Read Test will be used by the Cache Tag Memory Address Test.

If an error should be detected, the test will continue to loop on the entire range of Cache Tag Memory addresses.

Cache Tag Memory 3-Pattern Test — Sun-4/2xx Only

The Cache Tag Memory 3-Pattern Test performs three separate write-read-compare cycles on each address within the Cache Tag Memory. In pass *one*, patterns 0x5a972c5a, 0x972c5a97 and 0x2c5a972c is repeated throughout the Cache Tag Memory before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns 0x972c5a97, 0x2c5a972c and 0x5a972c5a. Finally, pass *three* will use patterns 0x2c5a972c, 0x5a972c5a and 0x972c5a97.

The same LED display used by the Cache Tag Memory Write-Write-Read Test will be used by the Cache Tag Memory 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Cache Tag Memory addresses.

Cache Data Memory Write-Write-Read Test — Sun-4/2xx Only

Test 23 on a Sun-4/2xx will perform write-write-read-compare-read-compare cycles on each pair of addresses within the Cache Data RAM. For each pair of addresses within the Cache Data RAM, pattern 0x5a972c5a is written to address *x-1*, pattern 0xa568d3a5 is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error is detected, the test continues to loop on the pair of Cache Data addresses currently under test.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
23	0x17	bit 7 ○○○●○●●● bit 0	okay
23	0x97	●○○●○●●●	error

Cache Data Memory Address Test — Sun-4/2xx Only

The Cache Data Memory Address Test will write the address of each Cache Data Memory location to that location before reading back the value and comparing the observed value with the expected value. The entire range of Cache Data Memory addresses is written prior to the read and compare operations.

The same LED display used by the Cache Data Memory Write-Write-Read Test is used by the Cache Data Memory Address Test.

If an error should be detected, the test will continue to loop on the entire range of Cache Data Memory addresses.

Cache Data Memory 3-Pattern Test — Sun-4/2xx Only

The Cache Data Memory 3-Pattern Test performs three separate write-read-compare cycles on each address within the Cache Data Memory. In pass *one*, patterns 0x5a972c5a, 0x972c5a97 and 0x2c5a972c is repeated throughout the Cache Data Memory before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns 0x972c5a97, 0x2c5a972c and 0x5a972c5a. Finally, pass *three* will use patterns 0x2c5a972c, 0x5a972c5a and 0x972c5a97.

The same LED display used by the Cache Data Memory Write-Write-Read Test is used by the Cache Data Memory 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire range of Cache Data Memory addresses.

Cache Write/Read Hit/Miss Verify Test

Test 24 on a Sun-4/2xx verifies that the (1) write-hit, (2) write-miss, (3) read-hit, (4) read-miss and (5) write-back cache operations are functioning correctly.

In particular, the following steps are performed by the Cache Write/Read Hit/Miss Verify Test.

1. The cache is turned off.
2. The Cache Tag RAM is cleared.
3. An arbitrary pattern is written to a 64 long word block of main memory (twice the size of the cache). Filling this block of memory with a pattern enables the Cache Write/Read Hit/Miss Verify Test to determine whether or not subsequent cache write-back operations work as expected.
4. The cache is turned back on.
6. The address of each long word location, within the block of memory mentioned above, is written to that location. This operation should cause 8192 write-miss-without-write-back cycles, each of which is followed by three write-hit cycles, prior to causing 8192 write-miss-with-write-back cycles, each of which is followed by three write-hit cycles.
7. The content of each of the long words in memory is read back before the observed value is compared with the expected value. This read operation should cause 16384 read-miss-with-write-back cycles, each of which should be followed by three read-hit cycles.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
24	0x18	bit 7 ○○○●●○○○ bit 0	okay
24	0x98	●○○●●○○○	error

Cache Write/Read/Flush/Verify Test — Sun-4/2xx Only

Test 25 on a Sun-4/2xx verifies that the *page* flush cache operation is in good working order.

In particular, the following steps are performed by the Cache Write/Read/Flush/Verify Test.

1. The cache is turned off.
2. The Cache Tag RAM is cleared.
3. An arbitrary pattern is written to a 64 long word block of main memory (twice the size of the cache). Filling this block of memory with a pattern will enable the Cache Write/Read Hit/Miss Verify Test to determine whether or not subsequent cache write-back operations work as expected.
4. The cache is turned back on.
5. The value at each long word location, within the block of memory mentioned above, is read and written back to that same location. This operation should cause 16384 read-miss-with-write-back-write-hit-without-write-back cycles, each of which should be followed by three read-hit-without-write-back-write-hit-without-write-back cycles.
6. The *page* flush operation is repeated until the entire cache has been flushed.
7. The content of each of the long words in memory is read back before the observed value is compared with the expected value. This read operation should cause 16384 read-miss-without-write-back cycles, each of which should be followed by three read-hit cycles.

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
25	0x19	bit 7 ○○○●●○○● bit 0	okay
25	0x99	●○○●●○○●	error

Main Memory Tests

There are two different sets of Main Memory tests, one of which performs a more thorough check of Main Memory and, consequently requires more execution time. The more comprehensive tests are intended for use during manufacturing burn-in and at other times when a memory hardware failure is suspected. The other test provides a quick sanity check of the memory system under normal booting conditions.

If the Diagnostic Switch is in the NORM position, the Main Memory Address Test and the Main Memory Ones Complement Address Test are executed. The quantity of memory tested is controlled by the value in EEPROM location 0x15. Refer to the *EEPROM Layout* chapter for information on modifying this value.

However, if the Diagnostic Switch is in the DIAG position, the Main Memory Write-Write-Read Test, Main Memory Address Test and the Main Memory 3-Pattern Test are executed.

Main Memory Write-Write-Read Test — Diagnostic Mode Only

Test 26 on a Sun-4/2xx or Test 21 on a Sun-4/1xx performs write-write-read-compare-read-compare cycles on each pair of addresses within Main Memory, excluding the memory reserved for the monitor. For each pair of addresses within Main Memory, excluding memory required by the monitor, pattern 0x5a972c5a is written to address *x-1*, pattern 0xa568d3a5 is written to address *x*, address *x* is read, the observed value of address *x* is compared with the expected value, address *x-1* is read and the observed value of address *x-1* is compared with the expected value.

If an error is detected, the test continues to loop on the pair of Main Memory addresses currently under test.

Sun-4/2xx LED States

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
26	0x1a	bit 7 ○○○●●○●○bit 0	okay
26	0x9a	●○○●●○●○	error

Test Number	*Hexadecimal Value Of LEDs*	*Visual Representation*	*Condition*
21	0x15	bit 7 ○○○●○●○●bit 0	okay
21	0x95	●○○●○●○●	error

Main Memory Address Test

The amount of Main Memory to be tested is a function of the position of the Diagnostic Switch. See the previous discussion regarding the Main Memory Write-Write-Read Test for more detail.

The Main Memory Address Test will write the address of each Main Memory location, excluding those reserved by the monitor, to that location before reading back the value and comparing the observed value with the expected value. The entire range of Main Memory addresses not required by the monitor is written prior to the read and compare operations.

During Normal Mode, LED 5 will blink on and off during the Main Memory Address Test. The same LED displays used by the Main Memory Write-Write-Read Test are used by the Main Memory Address Test.

In contrast to the *diagnostic*
mode main memory tests, the *normal* mode Main Memory Address Test will be terminated if an error should be detected.

Main Memory 3-Pattern Test — Diagnostic Mode Only

The amount of Main Memory to be tested is a function of the position of the Diagnostic Switch. See the previous discussion regarding the Main Memory Write-Write-Read Test for more detail.

The Main Memory 3-Pattern Test will perform three separate write-read-compare cycles on each Main Memory location, excluding those reserved by the monitor. In pass *one*, patterns 0x5a972c5a, 0x972c5a97 and 0x2c5a972c is repeated throughout the Main Memory not required by the monitor before each address is read back and the observed value is compared with the expected value. Pass *two* will employ patterns 0x972c5a97, 0x2c5a972c and 0x5a972c5a. Finally, pass *three* will use patterns 0x2c5a972c, 0x5a972c5a and 0x972c5a97.

The same LED displays used by the Main Memory Write-Write-Read Test are used by the Main Memory 3-Pattern Test.

If an error should be detected, the test will continue to loop on the entire

Main Memory Ones Complement Address Test — Normal Mode Only

The value at location 0x15 of the EEPROM will determine the amount of Main Memory to be tested. Since the configuration information in the EEPROM may be modified, you will have the ability to control whether *none*, *part* or *all* of Main Memory will be tested. Instructions for modifying location 0x15 of the EEPROM are given in the *EEPROM Layout* chapter.

The Main Memory Ones Complement Address Test writes the ones complement of the address of each Main Memory location, excluding those reserved by the monitor, to that location before reading back the value and comparing the observed value with the expected value. The entire range of Main Memory addresses not required by the monitor will be written prior to the read and compare operations.

During the execution of the *normal* mode Main Memory Ones Complement Address Test, LED 5 will blink on and off.

Should an error should be detected, the *normal* mode Main Memory Ones Complement Address Test will terminate, rather than enter the scope loop that is characteristic of the *diagnostic* mode main memory tests.

11.3. Host System Initialization

Beyond the *error-free* completion of the power-up tests, but before the the execution of the default boot sequence, the CPU and memory require initialization. The initialization steps are listed below in an arbitrary order.

1. The System Enable Register is set to *normal* state.
2. The Context Register is set to context 0x0.

3. The Processor Status Register is initialized. Specifically, the CPU is in *supervisor* mode, window ''0x0'' is active, interrupts is *on* and traps is *enabled*. Furthermore, if the Floating Point Unit probe was successful, the FPU is *enabled*.
4. The virtual address of the Trap Table is written into the Trap Base Register.
5. The Window Invalid Mask register is initialized to 0x2.
6. The segment table entries for each context is initialized.
7. All page table entries is initialized.
8. Memory is sized.
    - A. A record of the total quantity of main memory that is physically present, both working and non-working, is stored in `*romp->v_memorysize`.
    - B. Variable `*romp->v_memoryavail` contains the amount of available memory, excluding non-working pages and pages reserved for the monitor itself.
9. All of the available main memory is initialized to 0x0 (an unimplemented instruction).
10. All available pages of *working* memory are mapped in ascending order, starting at location zero.
11. One page of RAM is reserved for a stack area and the Stack Pointer is initialized.
12. A page of RAM is reserved for the monitor's global variables.
13. An additional page of RAM is reserved for the monitor's Trap Vector Table and Font Table.
14. Assuming that they exist, the devices that are initialized and mapped are presented below.
    - A. ECC Memory Enable Register(s).
    - B. EEPROM.
    - C. EPROMs (Boot PROMs).
    - D. Ethernet.
    - E. Interrupt Register.
    - F. Keyboard.
    - G. Mouse.
    - H. Serial Ports A and B.
    - I. TOD Clock.
    - J. Video Memory.

15. The RAM copy of the monitor's Trap Vector Table, which contains the addresses of the trap and interrupt handling routines, is set up.
16. The entry points to all of the support routines provided by the monitor will be set up. In order to make use of these monitor support routines, programs must include the file `sunromvec.h`.
17. Cache initialization is performed.

11.4. The Boot Sequence

Following the initialization of the workstation, the default boot sequence is executed. There are two issues that must be considered here. One has to do with *what is to be down-loaded*; the other has to do with *from what device it is to be loaded.*

Assuming *no* operator intervention, the position of the Diagnostic Switch will determine *what* is to be booted. If the Diagnostic Switch is in the NORM position, the SunOS operating system is booted. Otherwise, if the Diagnostic Switch is in the DIAG position, the EEPROM-specified program, located in the Diagnostic Boot Path area of the EEPROM, is booted. Be aware that the monitor is invoked if no EEPROM-specified program is available.

If, however, you interrupt the automatic (default) boot sequence, the monitor program is invoked. At that point, you may specify *what* is to be booted and *where* it is to be booted from. See command **b** (boot) in the section titled *Basic Monitor Commands* for a description of how to boot user-specified programs from user-specified devices.

If you are interacting with a Sun-4 workstation through the Sun keyboard, you may interrupt the default boot sequence by typing **L1-a**. That is, hold down the L1 key while pressing the **a** key. On the other hand, if you are interacting with the work station through a "dumb" terminal, pressing the Break key terminates the automatic boot sequence.

CAUTION

When the system has a disk, do not use the `L1-a` sequence once the operating system is running. Doing so may result in damage to your file systems. Use `sync` and then `/etc/halt` if the operating system is running.

Once you are in the monitor mode (you will see the > prompt), you should reset the system to clear out all of the hardware settings. Do the following:

```
> g 0
panic: zero
Syncing disks... done
```

Press L1-a or Break again when the message above finishes. Next you may see this message:

```
dumping to dev somevalue, offset somevalue

Abort at somevalue
>
```

The firmware also determines from what boot device the program will be loaded. If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is equal to 0x12 (an arbitrarily chosen value), Sun-4 firmware will attempt to boot the operating system from the boot path specified in the EEPROM, beginning at location 0x19. If the boot path, which should begin at EEPROM location 0x19, is missing or contains an error, the monitor program is invoked.

If the Diagnostic Switch is in the NORM position and the content of EEPROM location 0x18 is *not* equal to 0x12, Sun-4 firmware attempts to boot the SunOS operating system, using the following boot device polling sequence:

1. Xylogics Disk.
2. SCSI Disk.
3. Ethernet.

If the Diagnostic Switch is in the DIAG position, the firmware assumes that both the path name of the file containing the program to be loaded and the boot device are specified in the EEPROM, beginning at EEPROM location 0x000022. If either the file name or the boot device is not present or is in error, the monitor is invoked.

12

Sun-4 PROM Monitor Commands

Sun-4 PROM Monitor Commands **207**

12.1. What's In This Chapter 207
12.2. Monitor Command Overview 207
12.3. The Monitor Commands 209

12

Sun-4 PROM Monitor Commands

12.1. What's In This Chapter

This chapter describes the PROM monitor (sometimes called the "system monitor") commands available to Sun-4 workstations. Taken as a whole, these commands offer a low-level user interface to the Sun hardware. The commands control a variety of options, including booting from an alternate device, changing the console output, reading or altering registers or memory locations, and so on.

Most of the monitor commands are transient (commands whose effects disappear when the power is turned off), with the exception of the `q` command, which programs the EEPROM and remains in effect until deliberately altered with another PROM command. The `q` command *reconfigures* your system. The Boot PROM consults the EEPROM to determine much of its configuration information, such as the booting sequence, primary input and output devices, and so on.

See the **q** command, in this chapter, for details on how to program the EEPROM and the *EEPROM Layout* chapter for parameter entry locations.

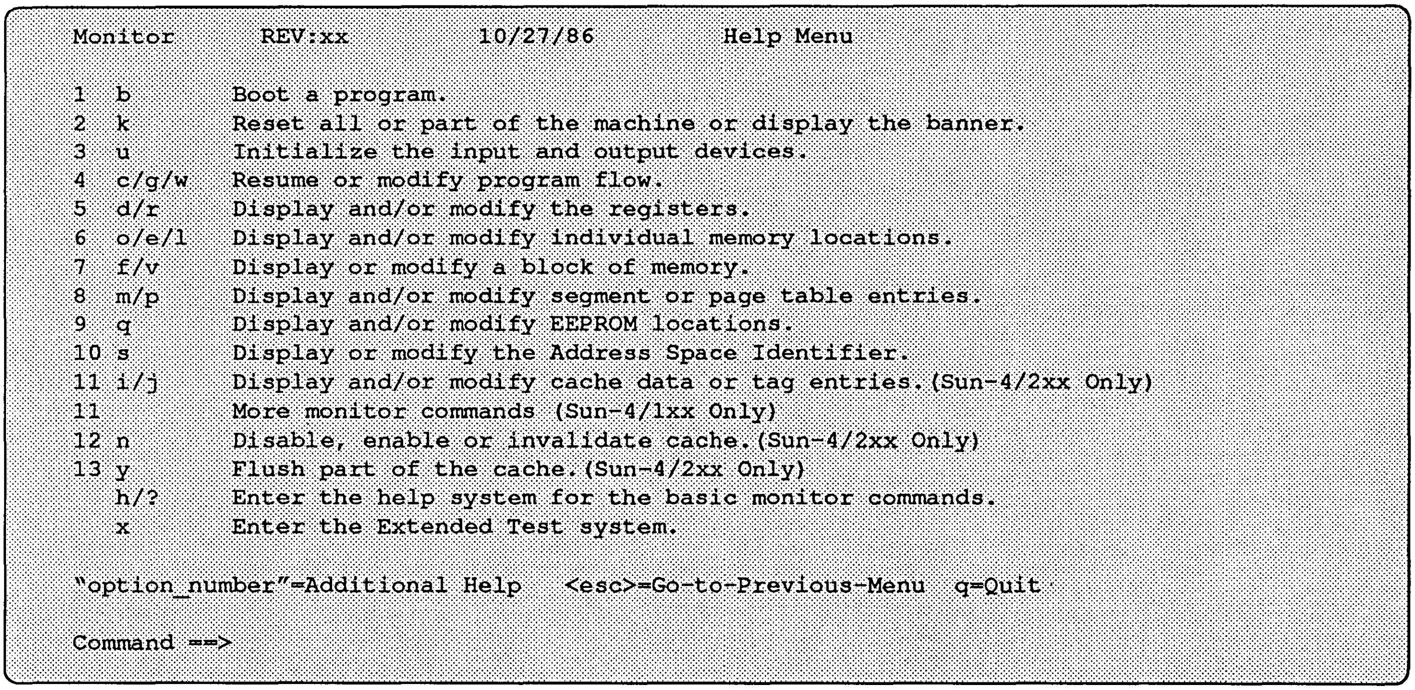
Bringing up the PROM Monitor

Read *Chapter 3* for instructions on bringing up the PROM monitor.

12.2. Monitor Command Overview

The example that follows shows the menu that comes up when you enter **h** (help) at the monitor prompt. This section describes the general characteristics that all of the commands have in common. Detailed descriptions of each command and its arguments are listed in the next section. For on-line help concerning one of the commands, after bringing up the help menu, enter the number that appears to the left of the command and then press Return.

Figure 12-1 *Monitor Help Menu*



```
Monitor      REV:xx        10/27/86        Help Menu

1  b       Boot a program.
2  k       Reset all or part of the machine or display the banner.
3  u       Initialize the input and output devices.
4  c/g/w   Resume or modify program flow.
5  d/r     Display and/or modify the registers.
6  o/e/l   Display and/or modify individual memory locations.
7  f/v     Display or modify a block of memory.
8  m/p     Display and/or modify segment or page table entries.
9  q       Display and/or modify EEPROM locations.
10 s       Display or modify the Address Space Identifier.
11 i/j     Display and/or modify cache data or tag entries.(Sun-4/2xx Only)
11         More monitor commands (Sun-4/1xx Only)
12 n       Disable, enable or invalidate cache.(Sun-4/2xx Only)
13 y       Flush part of the cache.(Sun-4/2xx Only)
   h/?     Enter the help system for the basic monitor commands.
   x       Enter the Extended Test system.

"option_number"=Additional Help   <esc>=Go-to-Previous-Menu  q=Quit

Command ==>
```

Getting Help

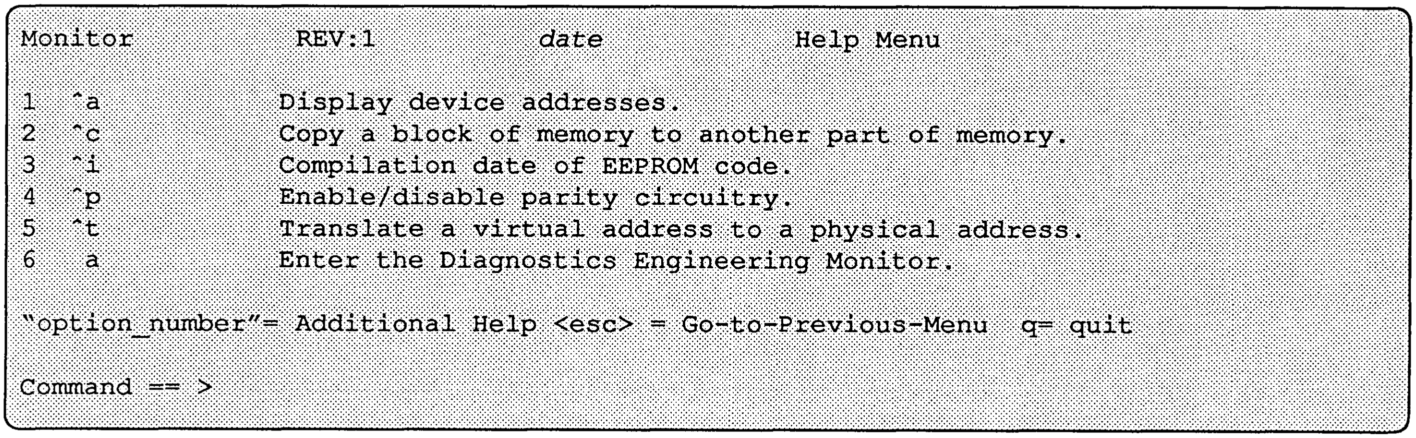
If you want further help with one of the commands shown above, enter the number to the left of the command, like this:

```
Command== >1
```

If you entered `1`, more detailed information about the `Boot` command would be displayed.

Sun-4/1xx Additional Menu

If you have a Sun-4/100 series system and you did not find the topic you need help with in the help menu, enter "11" (More monitor commands) on the command line at the bottom of the menu. This will take you to an additional menu that is similar to the following.



```
Monitor          REV:1          date            Help Menu

1  ^a           Display device addresses.
2  ^c           Copy a block of memory to another part of memory.
3  ^i           Compilation date of EEPROM code.
4  ^p           Enable/disable parity circuitry.
5  ^t           Translate a virtual address to a physical address.
6   a           Enter the Diagnostics Engineering Monitor.

"option_number"= Additional Help <esc> = Go-to-Previous-Menu  q= quit

Command == >
```

Executing a Command

The monitor command syntax in this chapter shows the command in **`bold typewriter font`** and the optional arguments in *`italic typewriter font`*. The text that follows the command line tells you the possible arguments you may use in place of the *`italic_descriptions`*.

In general, to execute a command, you type the appropriate command letter, followed by any required command arguments. For example, if you wanted to execute the hypothetical command θ, which we will say needs the arguments ''100'' and ''200'', you would type a line like this:

```
> θ 100 200
```

The command letter can be upper or lower case, and all commands and arguments are separated by white-space (tabs or spaces). Pressing the return key executes the command.

When typing commands, the [BackSpace] or [Delete] keys erase the last character typed.

Default Values

Many of the monitor commands have built in, or *default* values, which the command uses if you do not supply arguments. The default values vary from command to command. Check the command descriptions for the default values of interest.

Word Sizes

Word sizes referred to in this chapter are defined as follows: A **byte** is eight bits long; a **word** is 16 bits long; a **long word** is 32 bits long.

12.3. The Monitor Commands

This section provides more detailed information on the commands listed in the previous section. Both the commands and their arguments are described here. Note that any changes made with these commands, with the exception of the **`q`** command for EEPROM programming, are valid only until the system is powered down. Parameters changed with the **`q`** command take effect when you power cycle or do a **`k2`** reset to the system, and remain in effect until you use the **`q`** command again to alter the contents of the EEPROM.

Displaying and Modifying Memory

A number of the commands described here can be used to display and/or modify sections of the workstation memory. Regardless of the type of memory (RAM, EEPROM, etc.), these commands have the same command syntax. This section describes the memory modification syntax used by the monitor commands. The example here is referencing long words (32 bits) of memory, but the syntax is the same for bytes (8 bits) and words (16 bits).

Memory modification commands accept two arguments. The *`address`* argument specifies the initial memory location that is to be displayed and/or modified.

The second argument determines whether the current content of a memory location is to be displayed and/or modified. Entering *only* the address of a memory location after the command *displays* the content of that address.

```
>command ffe80000 Return
ffe80000: 1234 ?
```

At this point, you can respond in one of *four* different ways.

1. Simply pressing the Return key displays the contents of the next memory location (in this case, 0xFFE80004) as shown below.

```
>command ffe80000 Return

ffe80000: 1234 ?  Return

FFE80002: 0001 ?
```

Successively pressing the Return key displays the contents of successive memory locations. Assuming that you pressed Return four times, the contents of memory locations 0xffe80002, 0xffe80004, 0xffe10006 and 0xffe0008 would be displayed.

2. The second way to respond to the

   `> ffe80000: 1234?`

   display returns you to the PROM monitor command. To exit the display/modify mode, enter any *non-hexadecimal* character other than the plus or minus signs (enter `q` for quit, for example) before pressing Return. Note that you will now return to the monitor's basic command level, with the > prompt.

```
>command FFE80000 Return

FFE80000: 1234 ? q Return
>
```

3. The *third* response to the display of memory location contents is to change the direction in which the addresses are displayed. For information on using plus and minus characters to do this, please refer to the section titled *Special Monitor Commands.*

Following the assignment of the new value to memory location 0xffe80000, the new value will *not* be displayed; instead, the contents of the next memory location are shown. You will *not* be returned to the monitor's basic command level.

4. The fourth method of response to the

   `> ffe80000: 1234?`

   display is to *modify* those contents. To do so, you enter the *new* hexadecimal value immediately following the question mark `?`, BEFORE pressing Return. The following display demonstrates how the value of memory location 0xFFE80000 can be changed from "1234" to "00abcd".

```
    >command ffe80000 Return

    ffe80000: 1234 ? 00abcd Return

    FFE80004: 00000001 ?
```

Entering a memory location's virtual address followed *only* by a non-hexadecimal character before pressing the Return key causes the monitor to *display* the contents of that location then exit to the monitor command level.

```
>command ffe80004 q Return

FFE80004: 00000001

>
```

Entering a non-hexadecimal character *and* a hexadecimal value after an address causes the monitor to display the original value at that virtual address, assign a new value, then display that value. For instance, assume that the original contents at address 0xffe8000a is "0005". The command

```
>command FFE8000a ? 5550 Return
```

displays the original value "0005" then assigns the value "5550" to word address 0xffe80000a before displaying the contents of the next address, as shown below:

```
>command ffe8000a ? 5550 Return
ffe8000a: 0005 -> 5550
>
```

Entering a hexadecimal value after the virtual address corresponding to a word assigns the new value to that word, displays the newly-assigned value, then returns to the monitor. If you enter:

```
>command ffe8000c 6660 Return
```

The program will answer back:

```
ffe8000c -> 6660
>
```

You may also enter multiple display and/or modify commands for multiple memory locations on the same command line. If you enter this command line:

```
>command ffe80000 ? 0000 ? ? 2220 3330 q Return
```

You will see this on the screen:

```
ffe80000: 1234 -> 0000
ffe80002: 0001
ffe80004: 0002 -> 2220
ffe80006 -> 3330
ffe80008: 0004
>
```

The first part of the command line,

>*command* **ffe80000 ? 0000**

displays the original contents of location ffe80000 before assigning and displaying the new value "0000" to.

The next question mark directs the monitor to display the contents of the word ffe80002. The next part of the command line, **? 2220**, tells the monitor to display the original contents of ffe80004, before assigning and displaying the new value "2220" to it. The **3330** tells the monitor to assign the value "3330" to word ffe80006 before displaying the new value. Finally, the q causes the monitor to exit to the command level after the contents of ffe80008 are displayed.

Special Monitor Commands

Address Increment/Decrement Command

By preceding the command with a **+** or **-**, you can cause the address display to increment or decrement to the next location.

While traversing a range of addresses, type a + or - to change direction after the program displays the content and waits for input. For example:

```
>l 0 Return

00000000 00000000 ?  Return

00000004 00000001 ?  Return

00000008 00000002 ? - Return

00000004 00000001 ?  Return (contents of previous location are displayed)

00000000 00000000 ? + Return (after decrement, you enter a plus sign)

00000004 00000001 ?  Return (you increment again)

00000008 00000002 ?  Return (you increment again)
```

The ^T Command

Entering the ^ character and then the **t** key, followed by a virtual address, displays the physical address to which that address is mapped, along with a detailed description of all the bits in the page table entry, the segment and page RAM addresses, and what space they are in.

For example, entering

```
>^t 1000 Return
```

results in this display:

```
Virtual Addr 1000 is mapped to Physical Addr 1000
Context = 0x0, Seg Map = 0x0, Page Map = 0xC0000000.
Page 0 has these attributes:

       Valid bit  = 0
       Permission = 1
       No Cache   = 0
       Type       = 0
       Access bit = 0
       Modify bit = 0
```

Entering **`^t`** with no argument provides information with reference to virtual address 0x00.

The **`^I`** Command

Entering the ^ character and then the **`i`** key, followed with a command, displays compilation information concerning the system firmware. It includes the date, host name and build directory path. For example:

`Compiled at 6/7/87 on` *`hostname`* `in` *`/directory_name`*

The **`^C`** Command

`^C` *`source destination n`*

Entering the ^ character and then the **`c`** key, followed with the parameters shown, causes a block of *`n`* length to be copied from *`source`* to *`destination`* address, byte by byte. There is enough delay to copy to EEPROM also.

List of Sun-4 Monitor Commands

The paragraphs below describe each of the monitor commands in detail. Each paragraph starts with a line describing the command syntax. If a command has more than one distinct format, each one is shown on a separate line. Letters in **`bold typewriter`** font mean you should enter them exactly as shown. Plain `typewriter` font represents what you should see on the screen, or the name of a program or file. Words in *`typewriter italic`* show the type of information you are to enter. *Roman italic* or **boldfaced** font is also used for emphasis within the text. Optional arguments and default values are listed in the descriptions.

Monitor **`b`** Command

`b` ? or **`b`** *`! boot_device path argument_list`*

The `boot` command loads and executes the operating system, an EEPROM-specified program, or a user-specified program. The boot program can be loaded from the default device, the device specified in the EEPROM, or the boot device specified in the command argument. A *`boot_device`* is a secondary storage device (disk, Ethernet or tape) that contains the program to be loaded and executed.

If the Diagnostic Switch on the back of the system is in the NORM position, the value in EEPROM address 0x18 is *not* equal to 0x12, and the boot command is entered without arguments, the system will boot the SunOS operating system, using the following default boot device polling sequence.

1. Xylogics Disk
2. SCSI Disk
3. Ethernet

If the EEPROM value at address 0x18 *is* equal to 0x12, the system will boot the SunOS operating system from an EEPROM-specified device. The boot device is specified in locations 0x19 through 0x1D, inclusive, of the EEPROM. Refer to the `q` command for information on how to open and modify these EEPROM locations.

When the Diagnostic Switch is in the DIAG position and command **`b`** is entered by itself, the system will boot an EEPROM-specified program from an EEPROM-specified device. In this case, the boot path is specified in locations 0x28 through 0x50, inclusive, of the EEPROM; the boot device is specified in locations 0x22 through 0x26, inclusive, of the EEPROM. If the boot attempt fails, the user is returned to the monitor's command level.

In order to boot from a specific device, the **`b`** command must be followed with a boot device abbreviation, such as those shown below. Enter **`b ?`** to view the boot device identifier arguments that your PROM monitor will accept.

```
>b device (controller,unit,partition)path argument_list
```

`device` may be one of the following:

xy — Xylogics 7053/450/451 disk
sd — SCSI disk
ie — Intel Ethernet
st — SCSI tape
xt — Xylogics 472 Tape
mt — Tape Master 9-Track Tape

You may enter `()` or `(,,)` after *`device`* and the default values for the controller, disk and partition will be used. If you enter values, you must type in the parentheses and commas.

`controller` stands for the Controller Number, referring to the tape or disk controller board. The default is 0.

`unit` refers to Unit Number, meaning disk number. The default is 0.

`partition` is the partition number of the boot device. The default is 0.

`path` is the path and filename of the program to boot.

`argument_list` is the list of arguments for the boot program. Up to seven optional arguments (which are passed to the boot program) may follow the path argument.

If you do not want the system to be reset prior to booting, you must insert ! before the device identifier argument.

The boot command given below would boot the video diagnostic from the `/stand` directory over the Ethernet. Because the ! argument does *not* precede the device identifier argument, the system is reset before the booting process begins. Note that one optional argument (`-t`) is passed on to program `video.diag`.

```
>b ie(0,0,1)/stand/video.diag -t
```

Monitor c Command

c *virtual_address*

The `continue` command resumes execution of an interrupted program. You can specify the virtual address of the instruction to be executed when the program restarts.

The default `virtual address` is equal to the content of the Program Counter.

NOTE *This command is helpful if you should use the* `L1-A` *sequence and then wish to restart the operating system.*

Monitor d Command

d *`window_number`* or **d** *`register_type`*

The *dump listing* command displays (dumps) the state of the processor. The processor state display consists of the six Special registers (PSR, PC, nPC, TBR, WIM, and Y), followed by the eight Global registers, then the 24 Window registers (eight *in*, eight *local* and eight *out* registers) corresponding to one of seven windows. If a Floating Point Unit is on board, the Floating Point Status Register and the 32 Floating Point Registers are also displayed.

By specifying optional argument *`window_number`* (0, 1, 2, 3, 4, 5, 6), you can display the registers within a particular window along with the Special, Global and Floating Point registers. As illustrated in the table below, entering `g`, `f` or `s` as `register_type` specifies the register type. If no window number is specified and the PSR's Current Window Pointer field contains a valid window number, the *previous* window's registers are displayed. The *previous* window will be defined as the window which was active immediately *before* the monitor program was entered. If the PSR's Current Window Pointer is invalid, window zero is displayed as the default.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`)
- You have manually "broken" into the monitor (by typing `L1-a` on the console's keyboard or break on the "dumb" terminal's keyboard).

The table below illustrates the results of various command/parameter entries:

Command	*Result*
dg	displays only Global Registers
df	displays only FPU Registers
ds	displays only Special Registers
dw	displays only the previous window
dwc	displays only the previous window
dw0	displays only window 0
dw1	displays only window 1
dw2	displays only window 2
dw3	displays only window 3
dw4	displays only window 4
dw5	displays only window 5
dw6	displays only window 6

Monitor **e** Command

e *`virtual_address`*

The `display/modify memory` command displays and/or modifies the content of one or more virtual addresses in *word* mode (i.e. each virtual address will be treated as a 16-bit unit). That is, the content of one or more words can be *displayed, modified* or, both *displayed* and *modified.*

See the previous section *Displaying and Modifying Memory* for a description of this command's syntax. Use the letter **e** in place of the word *`command`* in the description.

Monitor **f** Command

f *`start_virtual_address end_virtual_address pattern size`*

The *block write* command writes the *`pattern`* you enter into each byte, word or long word in the range of virtual addresses you specify with the *`start_virtual_address`* and *`end_virtual_address`* arguments . By default, if the final, memory-cell-size argument is not present, bytes are written. Arguments *`start_virtual_address end_virtual_address`* and *`pattern`* are required while *`size`* is optional. The possible values for *`size`* are `b` (8-bit byte), `w` (16-bit word) or `l` (32-bit long word).

Monitor **g** Command

g *vector argument*
or

g *virtual_address argument*

When you use the `goto` command, you may go to (jump to) a *pre-determined, user-specified* or *default* routine. If a pre-determined or default routine is invoked, optional arguments `vector` (a hexadecimal number) and `argument` (a string) are passed to the routine to be executed.

In its other form, the argument `virtual_address` is used to indicate the virtual address of a *user-specified* routine, and the optional `argument` is passed along to that routine. If you do not supply the `vector/virtual_address` argument, the value in the Program Counter is used.

In order to set up a *pre-determined* routine, a user program has to set variable `*romp->v_vector_cmd` equal to the virtual address of the routine. Variable `*romp->v_vector_cmd` must be set prior to executing the `g` command. Pre-determined routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified `vector` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are `%x` and `%d`. Format `%x` prints argument `vector` as a hexadecimal number while format `%d` prints argument `vector` as a decimal number.

Monitor **h** Command

`h`

The `help` command invokes the Help System for the basic monitor commands. Each of the basic monitor commands and its argument(s) are described in this help system. No arguments are accepted by the **`h`** command, but after bringing up the help screen you may enter the number that appears to the left of a command for more on-line information about that command. For example, if you wanted to read more about the **`s`** command, you would enter the number **`10`**. Then, to return to the monitor's basic command level, press the [Esc] key or the **`q`** key before pressing [Return].

The initial help menu of the Help System is shown at the beginning of this chapter.

Monitor **i** Command

`i` *`cache_data_offset`*

The `modify cache data` RAM command displays and/or modifies the contents of one or more of the Sun-4 cache data addresses. Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`i`**.

Monitor **j** Command

`j` *`cache_tag_offset`*

The `modify cache tag` RAM command displays and/or modifies the contents of one or more of the Sun-4 cache tag addresses. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`j`**.

Monitor **k** Command

`k` *`reset_level`*

The `reset` command performs various levels of system resets. It can also be used to display the system banner. This command accepts one optional argument.

Entering **`k 0`** resets the VME-bus, interrupt register and video monitor (the Low-Level Reset).

Entering **`k 1`** invokes a Software Reset.

Entering **`k 2`** invokes a Power-On Reset (Hard Reset).

Finally, entering **`k b`** displays the banner on the video monitor. The default value of the argument is `0`.

Monitor **l** Command

l *`virtual_address`*

The `modify long words of memory` command displays and/or modifies the contents of one or more virtual addresses in *long* word mode. Each virtual address is treated as a 32-bit unit (long word). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **l**.

Monitor **m** Command

m *`virtual_address`*

The `modify segment table` command displays and/or modifies the contents of one or more of the Segment Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **m**.

Monitor **n** Command

n *`cache_command`*

The `control cache` command globally controls the cache. One argument is required. Entering **n d** *disables* the cache. Entering **n e** *enables* the cache. Finally, entering **n i** *invalidates* the cache.

Monitor **o** Command

o *`virtual_address`*

The `modify bytes of memory` command displays and/or modifies the content of one or more virtual addresses in *byte* mode. Each virtual address is treated as an 8-bit unit (byte). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter `o`.

Monitor **p** Command

p *`virtual_address`*

The `modify page table` command displays and/or modifies the contents of one or more of the Page Table entries. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **p**.

Monitor **q** Command

q *`eeprom_offset`* or **q ***

The `modify bytes of EEPROM` command displays and/or modifies the configuration information within the EEPROM in *byte* mode. This command works similarly to the memory commands discussed previously, except that the modified addresses are offset, and the changes you make remain when you power-down the workstation. Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **q**. Refer to the *Sun-4 EEPROM Layout* chapter for information on what parameters are stored at which EEPROM addresses.

You may enter an asterisk instead of an EEPROM offset value as an argument, and the **q** command will erase the entire EEPROM.

Monitor `r` Command

`r w` *`window_number`*
or

`r` *`register_type`*
or

`r` *`register_number`*

The `modify register` command displays and/or modifies the contents of one or more of the IU (Integer Unit or CPU) and/or FPU (Floating Point Unit) registers. If the *`register_type`* argument is **`f`**, **`g`** or **`s`**, the first Floating Point, Global or Special register will be displayed, respectively.

If **`w`** and a *`window_number`* (0, 1, 2, 3, 4, 5, 6) is entered, the first *in* register within the user-specified window will be displayed. If no *`window_number`* is specified and the PSR's Current Window Pointer field contains a valid window number, the *previous* window's first *in* register is displayed. The *previous* window will be defined as the window which was active immediately *before* the monitor was entered. If the PSR's Current Window Pointer is invalid, the first *in* register within window zero will be displayed.

If a *`register_number`* (a *hexadecimal* number) is provided, that specific register will be displayed. The default value of argument *`register_number`* is "0". The name of each register and its corresponding *`register_number`* is shown below.

Table 12-1 *Register Numbers*

Processor Registers	
Register Number	*Register Name*
0x00 - 0x0f	window(0,i0), ..., window(0,i7), window(0,l0), ..., window(0,l7)
0x16 - 0x1f	window(1,i0), ..., window(1,i7), window(1,l0), ..., window(1,l7)
0x20 - 0x2f	window(2,i0), ..., window(2,i7), window(2,l0), ..., window(2,l7)
0x30 - 0x3f	window(3,i0), ..., window(3,i7), window(3,l0), ..., window(3,l7)
0x40 - 0x4f	window(4,i0), ..., window(4,i7), window(4,l0), ..., window(4,l7)
0x50 - 0x5f	window(5,i0), ..., window(5,i7), window(5,l0), ..., window(5,l7)
0x60 - 0x6f	window(6,i0), ..., window(6,i7), window(6,l0), ..., window(6,l7)
0x70 - 0x77	g0, g1, g2, g3, g4, g5, g6, g7
0x78 - 0x7d	PSR, PC, nPC, WIM, TBR, Y
0x7e - 0x9e	FSR, f0, ..., f31

It is important to note that it is *not* always possible to display the registers. They may be observed only after

- ☐ An unexpected trap
- ☐ A user program has "dropped" into the monitor (by calling monitor function `abortent`) or
- ☐ You have manually "broken" into the monitor (by typing **`L1-a`** on the console's keyboard or `Break` on the "dumb" terminal's keyboard).

Read the previous section *Displaying and Modifying Memory* for details on how to use this command. Replace the word `command` in the description with the letter **r**.

Monitor **s** Command

s *asi_value*

The `modify ASI` command sets or displays the ASI (Address Space Identifier). The value of the ASI determines which Address Identifier Space is accessed by the memory display and/or modify commands. If given no argument, the command prints the current value of of the ASI. The acceptable ASI argument values are shown in the following table.

Table 12-2 *ASI Values*

Name	*ASI*
Control Space	0x2
Segment Table	0x3
Page Table	0x4
User Instruction	0x8
Supervisor Instruction	0x9
User Data	0xa
Supervisor Data	0xb
Flush Segment	0xc
Flush Page	0xd
Flush Context	0xe
Cache Data	0xf

Monitor **u** Command

```
u port options baud_rate
   or
u echo
   or
u uvirtual_address
```

The *input/output* command configures the input and output devices and their characteristics or displays the current input and output device set-up.

The **u** command requires arguments to specify from which device(s) you want the system to expect input or which device(s) will display output.

If you do not enter an argument after the `u` command, the program will display the current settings. If no serial port is specified when changing baud rates, the baud rate of the current input device is changed. The default serial port baud rate is 9600 for both ports during a normal boot. During a diagnostic boot, defaults are 9600 baud for Port A and 1200 for Serial Port B.

Upon normal power-up (diag switch is in NORM position), the default console input device is the Sun keyboard, unless the EEPROM has specified another default input device. If the keyboard is unavailable, the system looks to serial port A for for input.

The default console output device is the Sun monitor (subject to change through EEPROM programming). If the workstation has a color monitor and a color board is unavailable, the program will look for a monochrome monitor as an output device.

You may alter the existing I/O settings while you are in the monitor mode, using the commands listed below; however, the default settings will be reinstated when the system is power cycled.

The `port` argument can be one of the following:

Table 12-3 *Port Arguments*

Port Argument	*Device*
a	Serial Port A
b	Serial Port B
k	Keyboard
s	Screen

The `options` arguments are:

Table 12-4 *Option Arguments*

Option Argument	*Meaning*
i	input
o	output
u	UART (Universal Asynchronous Receiver/Transmitter)
e	input echoed to output
ne	input *not* echoed to output
r	reset specified serial port

The `baud_rate` argument specifies baud rate of the serial port being discussed.

The `virtual_address` argument specifies the virtual address of the UART

Following are examples of port and options arguments:

- Enter **ua** or **ub** to select serial port A or B as the input and output device.
- Enter **u aio** or **u bio** to select serial port A or B as the input and output device.
- Enter **u ai** or **u bi** to select serial port A or B for input only.
- Enter **u ao** or **u bo** to select serial port A or B for output only.
- Enter **u k** to select the keyboard for input.
- Enter **u ki** to select the keyboard for input.
- Enter **u s** to select the screen for output.

- ☐ Enter **u so** to select the screen for output.
- ☐ Enter **u ks, sk** to select the keyboard for input and the screen for output.
- ☐ Enter **u a***baud rate* or **u b***baud rate* to set the serial port speed.
- ☐ Enter **u e** to cause the output to echo the input.
- ☐ Enter **u ne** to cause the output **not** to echo the input.
- ☐ Enter **u** *address* to set the serial port virtual address.
- ☐ A *previously mapped* UART can also be used for input and/or output. Command **u u***`virtual_address`*, where *`virtual_address`* is the virtual address of a previously mapped UART, changes the virtual address of the UART. Do not leave a space between the second **u** and the virtual address.

If the **u** command is not followed by an argument, the current settings are displayed. The current input device, output device, baud rate for serial ports A and B, UART's virtual address and input to output echo indicator are shown.

Monitor v Command

v *`start_virtual_address end_virtual_address size`*

The `display memory block` command displays the contents of each byte, word or long word in the range of virtual addresses specified by *`start_virtual_address`* and *`end_virtual_address`*. The *`word_size`* argument is optional; the default is to display memory in byte format. In this format, sixteen consecutive bytes are displayed on each line. In word format, eight consecutive words appear on each line.

If long word format is enabled, four consecutive long words appear on each line. To the far right of each line, the character corresponding to each byte is also shown. All bytes that contain a non-ASCII code are shown as a period (.). Legal values for *`size`* are **b** (8-bit byte), **w** (16-bit word), or **l** (32-bit long word).

To terminate the **v** command, press the *space bar*. To "freeze" the display, press any key *except* the space bar. To restart the **v** command, press the space bar again.

Monitor w Command

w *`virtual_address argument`*

The `set execution vector` command allows you to vector to a *pre-determined* or *default* routine. Optional arguments *`virtual_address`* and *`argument`* are passed along to the to-be-executed routine.

In order to set up a *pre-determined* routine, a user program sets the variable `*romp->v_vector_cmd` equal to the virtual address of the *pre-determined* routine. Variable `*romp->v_vector_cmd` must be set prior to executing the **w** command. *Pre-determined* routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified *`virtual_address`* argument according to the user-specified format (given in *`argument`*) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument *`virtual_address`* as a hexadecimal number; format "%d" prints it as a decimal number.

Monitor **x** Command

The `extended test system` command invokes the Extended Test System. See *Chapter 13* for details.

Monitor **y** Command

y *c number*
or

y *cache_section number virtual_address*

The `flush cache` command performs a number of *flush* operations on the cache. Depending on what is to be flushed, two or three arguments are required. In order to flush a context, enter command **y c** *number*, where *number* is a valid context number.

Entering **y s** *number virtual_address* flushes segment *virtual_address* within context *number*. The argument *number* is a valid context number.

Entering **y p** *number virtual_address* flushes page *virtual_address* within context *number*. The argument *number* is a valid context number.

13

Sun-4 Extended Test System

Sun-4 Extended Test System **227**

13.1. The User Interface 227
13.2. Extended Test Descriptions 229

13

Sun-4 Extended Test System

The Extended Test System is a suite of test routines that provide detailed testing of the system hardware. They provide an additional level of testing beyond the power-up self-tests. You can invoke the Extended Tests in two ways.

If the system is being booted with the diagnostic switch in DIAG position, this message is displayed on the workstation screen:

```
Type any character within 10 seconds to enter Extended Test System.
```

If you do type a character, the Extended Test System is invoked.

The other way to invoke the Extended Test System is to type the **`x`** command from the PROM monitor. You are in the monitor when you see the > prompt on the screen.

13.1. The User Interface

The user interface of the Extended Test System consists of a hierarchy of menus.

In addition to the commands listed in each menu, there are a set of commands available in all of the menus. These commands are not tests themselves, but make it easier to execute tests. To execute one of these special commands, simply enter it on the command line, as you would for a regular command. The special commands are listed below:

? Command

The `Help` command displays a descriptive paragraph, explaining the commands in the currently displayed menu.

`q` Command

The `Quit` command exits the current command, and returns to the PROM monitor program.

`Escape` Command

The Esc command returns to the previous non-help menu.

`Loop` Command

The `Loop` command repeats a command line a specified number of times. To use the loop command, enter the command followed by **`loop=`***n* where *n* indicates the number of times to repeat the command. Entering an asterisk (*) for *n* makes the command repeat forever. The value can be decimal or hexadecimal. If the value is in hexadecimal, preface the value with the letters `%h`, or `%x`. If the value is a decimal, preface it with %d. Entering no value resets `loop` to its default, which is one.

Control-C Command

`^C` stops a command that is looping because the loop command was executed.

! Command

The `History` command displays the last five command lines entered. To re-execute any of those commands, type the number printed along side the appropriate command.

The Command Line

While viewing the one menu you may enter commands from another menu without actually bringing up that menu, provided that you know the command and its arguments, and you are familiar with the menu structure. You may use the [Esc] command in the command line to signify that the following commands belong in the menu above the present one.

All commands and arguments are displayed in the menus with one or more letters capitalized. For example, the argument `pattern` is shown like this:

```
PATtern
```

In order to enter the argument `pattern`, you need only enter the letters shown in upper case, followed by the "equals" sign, when it is shown:

```
pat=0xa55aa55a
```

You may type in the entire word, if you wish, but it is only necessary that you enter the letters shown in upper case so that the diagnostic interpreter understands what you want to do. You may enter the commands and arguments in upper or lower case. It is important that you *do not* leave off the = sign when it is specified.

When you string commands together, you must separate each sequence with a semicolon (;). In each command string, you must put a space between the command and each parameter, as shown in the example command line below.

The following example command line, executed while viewing a hypothetical Main Menu, would execute two memory tests twice and both SCC tests five times before returning to the main menu.

```
m;ad p=2;c pat=1 pass=2; [Esc] ; sc;int baud=300 pa=5;ex b=600 pas=5; [Esc]
```

In the example, we are entering the command line from a hypothetical Main Menu that contains a Memory Menu and an SCC Menu. In the command line, entering `m` selects the Memory Test Menu from the main menu. The semicolon that follows separates the next command and its arguments.

The hypothetical Memory Test Menu contains and Address test and a Constant test. Therefore, `ad` stands for the Address Test. The address test allows you to specify the number of times it will run. `p=2` specifies "two passes" of the address test. The semicolon acts as a separator between the command strings.

The `c` following the second semicolon selects the Constant test from the Memory Test menu. `pat=1` specifies that the Constant memory test is to use "1" for the pattern, and `pass=2` tells the Constant memory test to run twice. Again, a semicolon separates the command string.

The Esc command tells the interpreter that the next higher level menu (in this case, the Main Menu) contains the next command choice. After the escape command and another semicolon, `sc` selects the hypothetical SCC menu. Let us say that menu contains Internal and External tests. `int` selects the Internal test, and the `baud` rate is set to 300, and the number of passes is set to 5 (`pa=5`). After another semicolon, `ex` selects the External test, and `b` selects a baud rate of 600. `pas=5` sets the number of External test passes to five. Because the menu selection for *number of passes* is shown as `PASs=`, we have shown that argument in all its variations, from the terse `p=` to the verbose `pass=`.

The final Esc in the command line brings you back to the Main Menu, which is just above the hypothetical SCC menu (`sc`).

13.2. Extended Test Descriptions

The Extended Test System is organized into a series of menus. Each menu is shown below, followed by descriptions of each of its commands.

To execute a command, you need only enter the capital letters from the menu. For example, if you want to enter the `All` test sequence, simply enter:

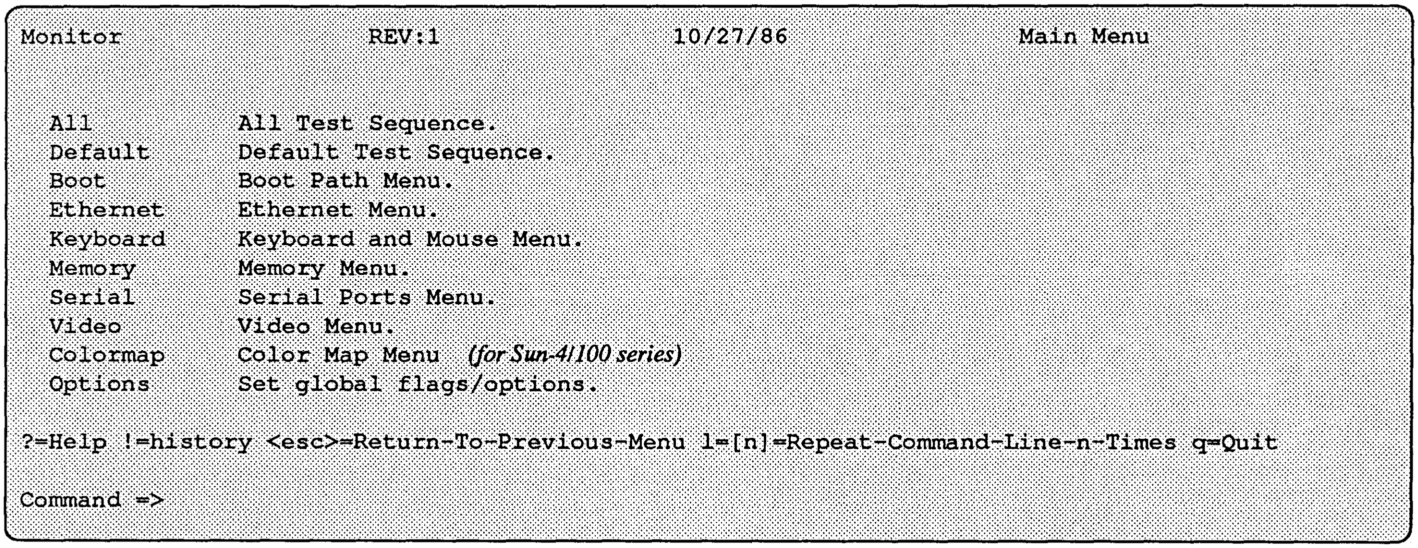
```
Command > a
```

Once a command completes, a Test Status Message appears on the screen and remains there for about five seconds.

To ''freeze'' the Test Status Message, press any key except `q`. In order to continue after ''freezing'' the state of the screen, press any key.

Main Menu

The Main Menu, shown below, has nine options. These options are discussed following this example.



```
Monitor                  REV:1                   10/27/86                Main Menu


  All          All Test Sequence.
  Default      Default Test Sequence.
  Boot         Boot Path Menu.
  Ethernet     Ethernet Menu.
  Keyboard     Keyboard and Mouse Menu.
  Memory       Memory Menu.
  Serial       Serial Ports Menu.
  Video        Video Menu.
  Colormap     Color Map Menu  (for Sun-4/100 series)
  Options      Set global flags/options.

?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit

Command =>
```

`All`

The following pages briefly describe each of the Main Menu commands. Commands that bring up a sub-menu are described in detail later in the chapter. `All Test Sequence` command runs the tests listed on the following page, in the order shown. Before running these tests, connect an Ethernet transceiver cable and within-channel external loop back connectors.

If you are using Port A as the input device, attach within-channel external loop back connectors to the keyboard/mouse and Port B connectors.

If you are using Port B as the input device, attach within-channel external loop back connectors to the keyboard/mouse and Port A connectors.

If you are using the keyboard for input, attach the within-channel external loop back connectors on Ports A and B.

The `All` test sequence runs these tests:

`Ethernet Local Loop Back Test`
`Ethernet Encoder Loop Back Test`
`Ethernet External Loop Back Test`
`Keyboard Register 12 Test`
`Mouse Register 12 Test`
`Keyboard Transmit Test`
`Mouse Transmit Test`
`Keyboard Internal Loop Back Test`
`Mouse Internal Loop Back Test`
`Keyboard External Loop Back Test` *(not run if keyboard is input device)*
`Mouse External Loop Back Test` *(not run if keyboard is input device)*
`Main Memory Address Test/Byte Mode`
`Main Memory Constant Pattern Test/Word Mode`
`Main Memory Fill Utility/Long Mode`
`Main Memory Check Utility/Long Mode`
`Serial Port A Register 12 Test`
`Serial Port B Register 12 Test`
`Serial Port A Transmit Test`
`Serial Port B Transmit Test`
`Serial Port A Internal Loop Back Test`
`Serial Port B Internal Loop Back Test`
`Serial Port A External Loop Back Test` *(not run if Port A is input device)*
`Serial Port B External Loop Back Test` *(not run if Port B is input device)*
`Video Address Test/Byte Mode`
`Video Constant Pattern Test/Word Mode`
`Video Fill Utility/Long Mode`
`Video Check Utility/Long Mode`

`Default`

The *Default Test Sequence* command runs the tests listed below in order.

`Ethernet Local Loop Back Test`
`Ethernet Encoder Loop Back Test`
`Keyboard Register 12 Test`
`Mouse Register 12 Test`
`Keyboard Internal Loop Back Test`
`Mouse Internal Loop Back Test`
`Main Memory Address Test/Long Mode`
`Serial Port A Register 12 Test`
`Serial Port B Register 12 Test`
`Serial Port A Internal Loop Back Test`
`Serial Port B Internal Loop Back Test`
`Video Address Test/Long Mode`

`Boot`

The `Boot` command displays the Boot Paths Menu, which contains all of the boot path tests, described later in this chapter.

`Ethernet`	The `Ethernet` command displays the Ethernet Menu, which contains all of the Ethernet tests described later in this chapter.
`Keyboard`	The `Keyboard and Mouse` command displays the Keyboard and Mouse Menu, which contains all of the keyboard and mouse tests described later in this chapter.
`Memory`	The `Memory` command displays the Memory Tests menu, which contains all of the memory tests described later in this chapter.
`Serial`	The `Serial` command displays the Serial Ports Menu, which contains all of the serial port tests described later in this chapter.
`Video`	The `Video` command displays the Video Menu, which contains all of the video tests described later in this chapter.
`Colormap`	The `Color Map` command appears only on extended test main menus for Sun-4/1xx systems. The test menu is the same as those described under *Video Menu*, with the addition of one more test. The additional test is the `default pattern fill` test, which fills the color map with a default pattern. The Color Map Menu presents this option as `DFill`. For this test, specifying a pattern has no effect. Refer to the section on running the video tests for more information.
`Options`	The `Options` command displays the Options Menu, which controls the global flags and parameters that control the behavior of the Extended Tests. These options are described later in this chapter. The Options Menu is also available from each sub-menu.

Boot Paths Menu

The Boot Paths Menu contains four tests. They are shown below:

```
Monitor              REV:1              10/27/86              Boot Paths Menu


  Sd          SCSI disk boot path test.

  ST          SCSI tape boot path test.

  Xy          Xylogics disk boot path test.

  XT          Xylogics tape boot path test.

  Options     Set global flags/options.

?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`Sd` Bootpath Test

The SCSI `Disk Bootpath Test` command tests the bootpath to the SCSI hard disk. It makes sure the machine can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code.

The command syntax is:

```
Sd Pass=
```

The program expects a base 10 number after *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter

```
s p=*
```

`ST` Bootpath Test

The SCSI `Tape Bootpath Test` command tests the bootpath to the SCSI tape. It makes sure the machine can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it will read the boot blocks into system memory without actually executing the boot code.

The command syntax is:

```
ST Pass=
```

The program expects a base 10 number after *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
st p=*
```

`Xy` Bootpath Test

The `Xylogics Disk Bootpath Test` command tests the bootpath to the Xylogics hard disk controller board. It makes sure the machine can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code.

The command syntax is:

```
Xy Pass=
```

The program expects a base 10 number after `pass=`. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
x p=*
```

`XT` Bootpath Test

The `Xylogics Tape Bootpath Test` command tests the bootpath to the Xylogics tape controller board. It makes sure the machine can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code.

The command syntax is:

```
XT Pass=
```

The program expects a base 10 number after `pass=`. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

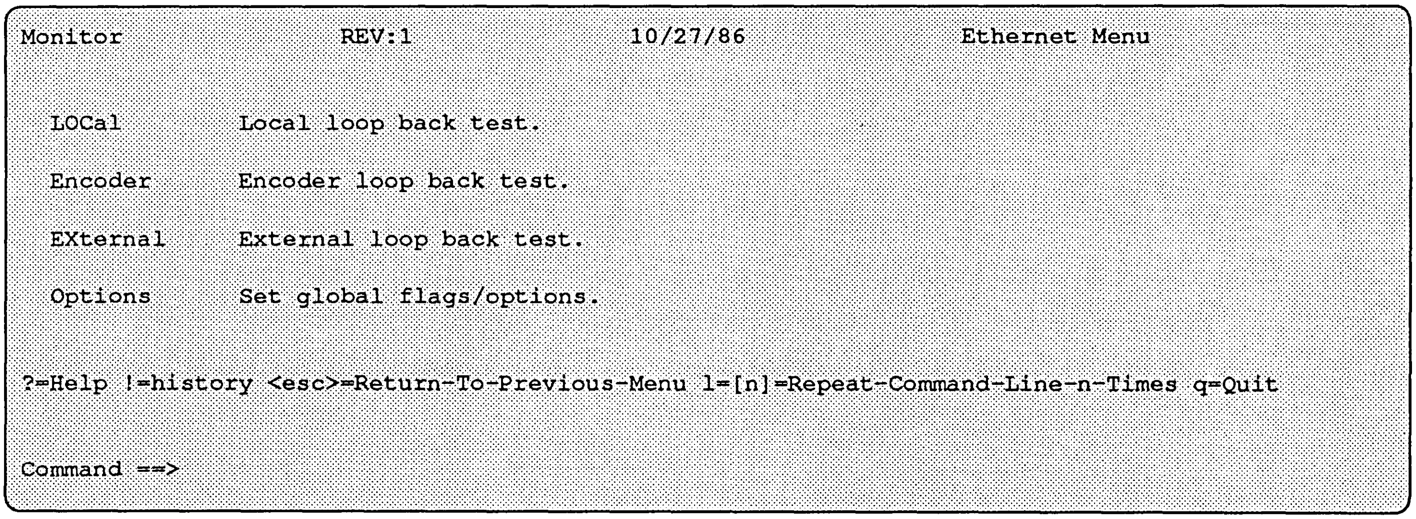
```
xt p=*
```

`Options` Command

The `Options` command displays the Options Menu, which contains all of the global flags and parameters that control the test commands and is discussed later in this chapter.

Ethernet Menu

The Ethernet Menu contains three tests. They check the functionality of the Intel 82501 Serial Interface Unit Chip, the Intel 82586 Ethernet LAN Co-Processor Chip and the connection to the Ethernet network. The Ethernet tests build upon each other. The second test will fail if the first test fails, and the third test will fail if either the first or second test fails.



```
Monitor              REV:1              10/27/86              Ethernet Menu


  LOCal        Local loop back test.

  Encoder      Encoder loop back test.

  EXternal     External loop back test.

  Options      Set global flags/options.


?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`LOCal` Command

The `local` command tests the Intel 82586 Ethernet LAN co-processor chip. It runs the internal tests built into the chip. Prior to the test, the Intel 82586 Ethernet LAN co-processor chip disconnects itself from the Intel 82501 Serial Interface Unit Chip.

The command syntax is:

```
LOCal Pass=
```

The program expects a base 10 number after *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
loc p=*
```

`Encoder` Command

The `encoder` command runs an internal loop back test of the Intel 82501 Serial Interface Unit Chip. The transmitter line, receiver line, noise filters and Manchester encoding/decoding logic are tested. Before executing this test, the transmitter and receiver lines of the chip are connected together.

The command syntax is:

```
Encoder Pass=
```

The program expects a base 10 number after *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
e p=*
```

`EXternal` Command

The `external` command runs the external loop back test. This test checks the Intel 82586 Ethernet LAN Co-processor Chip, the Intel 82501 Serial Interface Unit Chip and the Ethernet transceiver and receiver lines. The test sends data out onto the Ethernet, then receives it back and compares the transmitted and received data. Before running this test, attach an Ethernet transceiver cable to the CPU board and the terminator assemblies of the transceiver box.

The command syntax is:

```
EXternal Pass=
```

The program expects a base 10 number after *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
ex p=*
```

`Options` Menu

The `Options` command displays the Options Menu, which contains all of the global flags and parameters that control the test commands. This menu is discussed in a later section.

Keyboard and Mouse Menu

The Keyboard and Mouse Menu contains five tests. They are listed below:

```
Monitor          REV:1           10/27/86          Keyboard and Mouse Menu


  Register     Register 12 test.

  Transmit     Transmit character.

  Internal     Internal loop back test.

  External     Within-channel external loop back test.

  Keyboard     Keyboard input test.

  Options      Set global flags/options.


?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`Register` Command

`Register Channel= PATtern= Pass=`

The `Register` command performs write-read-compare cycles to register 12 of the port under test. This command accepts three arguments.

The *`channel=`* argument determines which port the test is performed on. Legal values for channel are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *`pattern=`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. *`pattern`* should be a hexadecimal (base 16) number. If you enter a base 10 number, precede it with `%d`.

The program expects a base 10 number after *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
r p=*
```

`Transmit` Command

`Transmit Channel= Baud= PATtern= Pass=`

The `Transmit` command writes patterns to the port under test. This command accepts four arguments.

The *`channel=`* argument determines on which port the test is performed. Legal values for *`channel`* are shown in the table that follows:

Argument	*Device*
k	Keyboard (default)
m	Mouse

The *baud=* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of *baud* should be a decimal (base 10) number.

The *pattern=* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. The *pattern* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
t p=*
```

Internal Command

Internal Channel= Baud= PATtern= Pass=

The `Internal` command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port are connected internally prior to the test.

This command accepts four arguments. The *channel=* argument determines on which port the test is performed. Legal values for *channel* are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *baud=* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table that follows:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern=`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. *`pattern`* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
i p=*
```

`External` Command

`External Channel= Baud= PATtern= Pass=`

The `External` command executes an external loop back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (see *Chapter 1*). This command accepts four arguments.

The *`channel=`* argument determines on which port the test is performed. Legal values for *`channel`* are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *`baud=`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern=`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. *`pattern`* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
e p=*
```

`Keyboard` Command

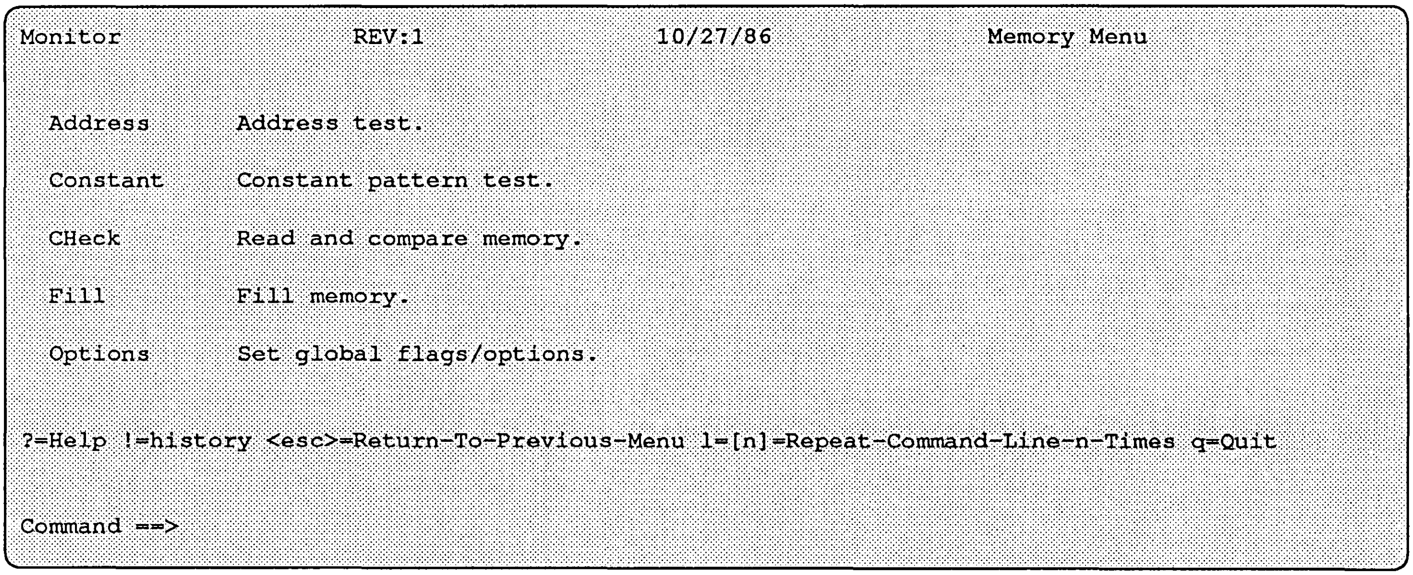
This command determines whether or not keyboard characters are correctly transmitted to the CPU. After you enter **`k`** form the `Keyboard and Mouse` menu, the ASCII code corresponding to a character and the character itself appear on the screen as you press keys on the keyboard. To exit the test, type an Esc character.

`Options` Command

The `Options` command displays the Options Menu, which contains all of the global flags and parameters that control the test commands. This menu is discussed later.

Memory Menu

The Memory Menu contains four options. They are described below.



```
Monitor                REV:1                10/27/86                Memory Menu


  Address      Address test.

  Constant     Constant pattern test.

  CHeck        Read and compare memory.

  Fill         Fill memory.

  Options      Set global flags/options.


?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

Address Test

Address Low= High= Cell= Pass=

The `Address Test` command executes the address test on a range of memory. When the test runs, write-read-compare cycles are performed on bytes, words or long-words, depending on the `cell=` argument (bytes are tested by default). The datum that is written to each memory ''cell'' is its own address. This command accepts four arguments.

The *low=* argument specifies the first physical address to test. By default, the value of *low* is `0x0`. The *low* value should be a hexadecimal (base 16) number.

The *high=* argument indicates the final physical address to test. The default high address is the highest memory address available. The *high* value should be a hexadecimal (base 16) number.

Legal values for `cell=` are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
a p=*
```

Constant Test

`Constant Low= High= PATtern= Cell= Pass=`

The `Constant` command performs write-read-compare cycles on a range of physical addresses with a specified pattern. The test accepts five optional arguments.

The *`low=`* argument specifies the first address to test. By default, the value of *`low`* is `0x0`. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final physical address to test. The default high address is the highest memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names patterns expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern`* is `0x0`. The *`pattern`* value should be a hexadecimal (base 16) number.

Legal values for *`cell`* are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
c p=*
```

CHeck Command

`CHeck Low= High= PATtern= Cell= Pass=`

The `Check` command reads the specified range of physical addresses, and checks for a pattern. The test accepts five optional arguments.

The *`low=`* argument specifies the first physical address to test. By default, the value of *`low`* is `0x0`. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final physical address to test. The default high address is the highest memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names patterns expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern`* is `0x0`. The *`pattern`* value should be a hexadecimal (base 16) number.

Legal values for `cell` are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
ch p=*
```

`Fill` **Command**

`Fill Low= High= PATtern= Cell= Pass=`

The `Fill` command writes a pattern to a range of addresses. The test accepts four optional arguments.

The *`low=`* argument specifies the first address to test. By default, the value of *`low`* is `0x0`. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final address to test. The default high address is the highest memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names patterns written throughout the range of addresses. The default value of *`pattern`* is `0x0`. The *`pattern`* value should be a hexadecimal (base 16) number.

Legal values for `cell` are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
f p=*
```

`Options` **Menu**

The `Options` command displays the Options Menu, which contains all of the global flags and parameters controlling the test commands. This menu is discussed later.

Serial Ports Menu

The Serial Ports Menu contains four tests. They are described in the text that follows the example menu:

```
Monitor            REV:1             10/27/86            Serial Ports Menu


  Register     Register 12 test.

  Transmit     Transmit character.

  Internal     Internal loop back test.

  External     Within-channel external loop back test.

  Options      Set global flags/options.


?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`Register` Command

`Register Channel= PATtern= Pass=`

The `Register` command performs write-read-compare cycles to Register 12 of the port under test. This command accepts three arguments.

The *`channel=`* argument determines on which port the test is performed. Legal values for *`channel`* are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *`pattern=`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. The *`pattern`* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
r p=*
```

`Transmit` Command

`Transmit Channel= Baud= PATtern= Pass=`

The `Transmit` command writes patterns to the port under test. This command accepts four arguments.

The *`channel=`* argument determines on which port the test is performed. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *baud=* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number.

The *pattern=* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. The *pattern* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *pass=*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
t p=*
```

`Internal` Command

Internal Channel= Baud= PATtern= Pass=

The `Internal` command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port are connected internally prior to the test. This command accepts four arguments.

The *channel=* argument determines on which port the test is performed. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *baud=* argument sets the baud rate at which the test is executed.

Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`.

The *`pattern=`* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
i p=*
```

`External` Command

E*`xternal Channel= Baud= PATtern= Pass=`*

The `External` command executes an external loop back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (see *Chapter 1*). This command accepts four arguments.

The *`channel=`* argument determines on which port the test is performed. Legal values for channel are shown in the table below:

Letter	*Device*
a	Serial Port A (default)
b	Serial Port B

The *`baud=`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0x0`. *`pattern`* should be a hexadecimal (base 16) number.

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

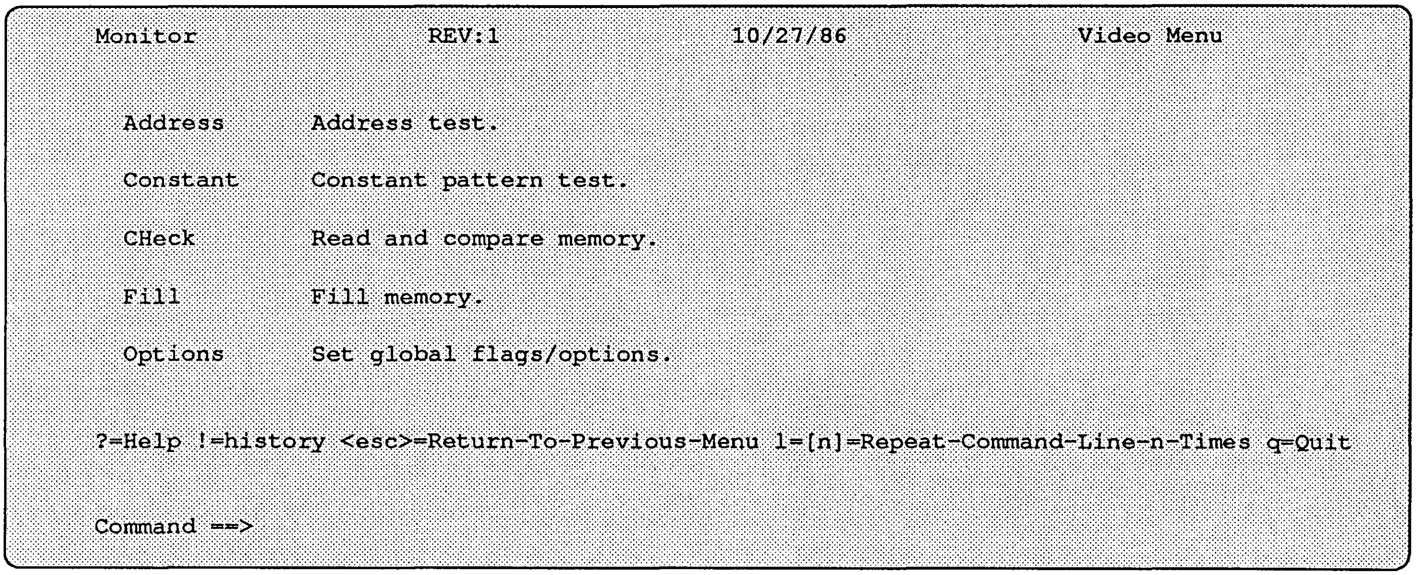
```
e p=*
```

`Options` Menu

The `Options` command displays the Options Menu, which contains all of the global flags and parameters controlling the test commands. This menu is discussed later.

Video Menu

The Video Menu contains four options. They are described below:



```
Monitor                REV:1                 10/27/86                 Video Menu


  Address       Address test.

  Constant      Constant pattern test.

  CHeck         Read and compare memory.

  Fill          Fill memory.

  Options       Set global flags/options.


?=Help !=history <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`Address` Test Command

A*ddress Low= High= Cell= Pass=*

The `Address Test` command executes the address test on a range of frame buffer memory. When the test runs, write-read-compare cycles are performed on bytes, words or long-words, depending on the `cell=` argument. The test are performed on bytes if no `cell` argument is entered. The datum that is written to each memory "cell" is its own address. This command accepts four arguments.

The *`low=`* argument specifies the first address to test. By default, the value of *`low`* is the lowest address in frame buffer memory. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high`* value should be a hexadecimal (base 16) number.

Legal values for *`cell=`* are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
a p=*
```

`Constant` Command

`Constant Low= High= PATtern= Cell= Pass=`

The `Constant` command performs write-read-compare cycles on a range of addresses with a specified pattern. The test accepts up to five optional arguments.

The *`low=`* argument specifies the first address to test. By default, the value of *`low`* is the lowest address in frame buffer memory. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names patterns expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern`* is `0x0`. The *`pattern`* value should be a hexadecimal (base 16) number.

Legal values for *`cell=`* are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
c p=*
```

`CHeck` Command

`CHeck Low= High= PATtern= Cell= Pass=`

The `Check` command reads the specified range of physical addresses, and checks for a pattern. The test accepts five optional arguments. The *`low`* argument specifies the first physical address to test.

By default, the value of *`low=`* is the lowest address in frame buffer memory. The *`low=`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final physical address to test. The default high address is the highest frame buffer memory address available. The *`high=`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names patterns expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern=`* is `0x0`. The *`pattern=`* value should be a hexadecimal (base 16) number.

Legal values for `cell=` are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
ch p=*
```

`Fill` Command

`Fill Low= High= PATtern= Cell= Pass=`

The `Fill` command writes a pattern to a range of physical addresses. The test accepts five optional arguments.

The *`low=`* argument specifies the first physical address to test. By default, the value of *`low`* is the lowest address in frame buffer memory. The *`low`* value should be a hexadecimal (base 16) number.

The *`high=`* argument indicates the final physical address to test. The default high address is the highest frame buffer memory address available. The *`high`* value should be a hexadecimal (base 16) number.

The *`pattern=`* argument names the pattern written throughout the range of addresses. The default value of *`pattern`* is `0x0`. The *`pattern`* value should be a hexadecimal (base 16) number. Legal values for

`cell=` are shown in the table below:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

The program expects a base 10 number after the argument *`pass=`*. If you want to enter a hexadecimal value, precede the value with %h or %x. If you want the test to continue until you interrupt it, enter:

```
f p=*
```

`Options`

Selecting `Options` displays the Options Menu, which contains all of the global flags and parameters controlling the test commands. This menu is discussed next.

Color Map Menu

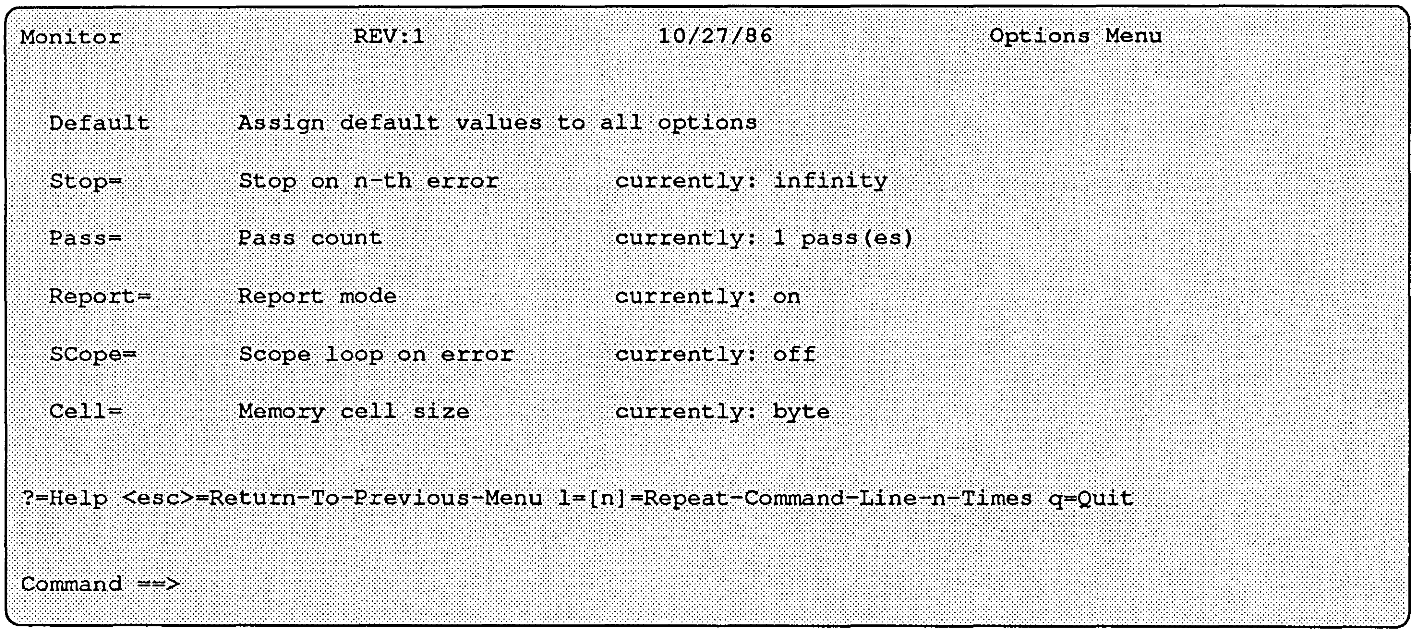
The Color Map Menu is available for Sun-4/xx only and is identical to the Video Menu, with the addition of this command:

```
DFill        Default pattern fill
```

This command fills the color map with a default pattern. Specifying a pattern has no effect.

Options Menu

The Options Menu sets the global flags and parameters that control the behavior of the tests under certain conditions. The flags and their effects are listed below.



```
Monitor                  REV:1                 10/27/86               Options Menu


  Default       Assign default values to all options

  Stop=         Stop on n-th error        currently: infinity

  Pass=         Pass count                currently: 1 pass(es)

  Report=       Report mode               currently: on

  SCope=        Scope loop on error       currently: off

  Cell=         Memory cell size          currently: byte


?=Help <esc>=Return-To-Previous-Menu l=[n]=Repeat-Command-Line-n-Times q=Quit


Command ==>
```

`Default`

The `Default` command sets all of the global flags to their default values. The default values are shown in the menu above.

`Stop=`

The `Stop on nth error` flag sets the *number of errors* that must occur before the test stops. Entering **`stop=*`** on the command line means tests will never stop for an error.

If the test stops for an error, it will wait until the you press a key. If you press **`q`**, the currently running test is terminated. If any other key is pressed, the test resumes.

`Pass=`

The `pass count` flag sets the number of times tests run. If you enter **`pass=*`** on the command line, the next test will run forever.

You can override the global pass count flag by specifying a *local* pass count value in the command line when a test is run.

Report=

The `report mode` flag controls whether status or error reports are displayed on the screen. This flag is useful for executing scope loops.

SCope=

The `scope loop on error` flag determines whether a scope loop starts when a test finds an error. Once a scope loop has been entered, you may press **q** to terminate the scope loop and the test. Pressing the **n** key terminates the scope loop on the current address and moves on to the next address. If the stop on nth error flag is set to some value other than infinity (*), the scope loop will stop after `n` errors occur and wait for user input. To continue the scope loop and test, press any key except **q** or **n**.

Cell=

The `memory cell size` flag determines the size of memory cells the tests use. The options are:

Cell	*Value*
`b or byte`	8 bits
`w or word`	16 bits
`l or long`	32 bits

Byte is the default cell size. You can override the global cell size flag by specifying a local cell size value in the command line when you invoke the test.

14

Sun-4 EEPROM Layout

Sun-4 EEPROM Layout **255**

14.1. EEPROM Introduction 255
14.2. Changing EEPROM Parameters 255
14.3. The EEPROM Layout 256

14

Sun-4 EEPROM Layout

14.1. EEPROM Introduction

The PROM monitor **q** command opens the EEPROM to allow examination or modification of configuration parameters. If you do not enter an address following the command, the content of the first address assigned to the EEPROM is presented. (EEPROM addresses are off-set, rather than complete addresses.)

EEPROM parameters set these functions:

- vary the quantity of memory tested during self-test;
- change the action that follows a watchdog reset;
- boot from a specified device with diagnostics switch on NORM, or poll the devices;
- recognize the specified device as the primary terminal or console;
- display the Sun banner or a custom banner during power-up;
- store and display a custom logo upon power-up;
- select special keyboard characters;
- turn the keyboard "click" on or off;
- boot a selected program from a specific device with diagnostics switch on DIAG;
- inhibit serial port DTR and RTS signals;
- select a serial port baud rate;
- store a system configuration record on EEPROM;
- erase EEPROM contents.

14.2. Changing EEPROM Parameters

The *Sun-4 PROM Monitor Commands* chapter contains **q** command syntax variations. The paragraphs that follow represent examples of one way to change or view EEPROM parameters. The layout section describes which parameters are stored at which EEPROM addresses.

To change the value of a specific EEPROM address, you must be in the monitor mode, (you'll see the > prompt). Now, enter the PROM monitor command **q**, the offset EEPROM address of the parameter you wish to change, and [Return].

When the program displays the contents of that location, enter the new value followed with a non-hexadecimal character, such as a period, or a **q** for quit, and [Return]:

```
>q 1f Return

>EEPROM 01f: 10? 11 q Return >
```

To exit from the modify mode when you have *not* entered a new value, simply press the space bar and Return after the question mark.

To increment to the next EEPROM address instead of returning to the PROM monitor program, simply press Return after the question mark, or immediately following your entry.

14.3. The EEPROM Layout

This section has a detailed description of the EEPROM layout. The layout is divided into a diagnostic section, a reserved section, a ROM section, and a software section.

The table on the following page provides an example of default system configuration parameters that may be present at the various locations, and what they mean. Note that the amount of memory may vary greatly; the amount shown is the minimum found in a basic system. The ports referenced are those present on the CPU board. This is an example only, actual entries may vary slightly from those shown on this table.

In this text, the EEPROM locations are described first, in numerical order, with tables that illustrate the result of various parameter entries. At the end of each description, the offset addresses are shown with illustrations of the content of each byte in that range. If, for example, the illustration shows `size` as the content of the first byte, the previous text would contain a table of possible hexadecimal values that `size` represents.

Table 14-1 *Default System Configuration Parameters for Sun-4 Systems*
F/D = Factory Defined U/D = User Defined N/A = Not Applicable

EEPROM Offset Address	*Function*	*Default Entry*	*EEPROM Offset Address*	*Function*	*Default Entry*
0x004-0x00E	Write Count & Checksum	F/D	0x58	U/D or Default Port A Baud Rate	0x00
0x010-0x013	Last Hardware Update	F/D	0x059-0x05A	Port A Alt Baud Rate	0x00
0x014	Installed Memory	0x08	0x05B	Port A DTR/RTS	0x00
0x015	Memory Tested	0x08	0x05C-0x05F	Reserved	N/A
0x016	Monitor Screen Size	0x13	0x060	U/D or Default Port B Baud Rate	0x00
0x017	Watchdog Action	0x00	0x061-0x062	Port B Alt Baud Rate	0x00
0x018	Boot Device:Poll/EEPROM	0x00	0x063	Port B DTR/RTS	0x00
0x019-0x1D	Alt.Boot Device	0x00	0x064-0x067	Reserved	N/A
0x01E	Keyboard Type	0x00	0x068-0x0B7	Custom Banner	0x00
0x01F	Primary Display	0x12	0x0B8	Test Pattern	0x0AA
0x020	Custom/Sun Banner	0x00	0x0B9	Test Pattern	0x55
0x021	Keyboard Click	0x00	0x0BC-0x18B	Configuration Blocks	F/D
0x022-0x026	Diag Boot Device	0x00	0x018C	Key Table Selector	0x00
0x028-0x04F	Diag Boot Path	0x00	0x018D	Locale Specifier	F/D
0x050	High Res Columns	0x50	0x018E	Keyboard ID	F/D
0x051	High Res Rows	0x22	0x190-0x20F	Lower Case Key Table	0x00
0x052-0x057	Reserved	N/A	0x210-0x28F	Upper Case Key Table	0x00
0x290-0x48F	Custom Logo	0x00	0x500-0x70A	Write Count & Checksum	F/D

Diagnostic EEPROM Test Write Area

Four bytes are provided for the EEPROM portion for the CPU Diagnostic. The contents of these locations after the test are meaningless because these four locations are NOT part of the checksum data area. The Diagnostic area write count locations are updated each time these locations are written.

Address[0x000-0x003]:

Diag Test
Diag Test
Diag Test
Diag Test

Diagnostic Area Write Count

These write counters are for the Diagnostic area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is reliability of their correctness.

Address[0x004-0x009]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Diagnostic Area Checksum

Each EEPROM area maintains three identical 8 bit (byte) checksums. These separate checksums are intended to be the same.

Address[0x00C-0x00E]:

Checksum #1
Checksum #2
Checksum #3

Date of Last System Hardware Update

This four byte location contains the date of the last system update. This date is recorded in the same fashion as the Manufacturing Date: total seconds since 1 January, 1970.

Address[0x010-0x013]:

Date (Bits 31-24)
Date (Bits 23-16)
Date (Bits 15-8)
Date (Bits 7-0)

Mbytes of installed Memory

This byte contains the total number (in hexadecimal) of Megabytes of memory installed in the system.

Address[0x014]:

Mbytes Installed

Mbytes of Memory to Test on Normal Boot

This byte contains the total number (in hexadecimal) of Megabytes of memory that the firmware tests prior to booting the SunOS operating system. The firmware ignores this value and tests all of memory if the diagnostic switch is in the DIAG position. All of memory is initialized even if not tested.

Address[0x015]:

Mbytes to Test

Monitor Screen Size

This byte selects the appropriate video screen sizes for the Monitor in the system. The following table illustrates the options:

Size	Definition
0x00	1152x900 Screen
0x12	1024x1024 Screen
0x13	1600x1280 Screen
0x14	1440x1440 Screen

Address[0x016]:

Size

A hardware change on the CPU Board is necessary to complete a screen size change.

Watchdog Reset Action

This byte selects the appropriate action for the firmware after a Watchdog Reset. The following table illustrates the options:

Action	Definition
0x00	Watchdog Reset will fall into Boot PROM Monitor
0x12	Watchdog Reset will cause a Power-On-Reset

Address[0x017]:

Action

Operating System Boot-up

This byte selects whether the Boot PROM polls for boot devices in the system or uses an EEPROM selectable boot device for loading the Sun Operating System (SunOS) during a normal boot. If this option is selected the boot device is specified in EEPROM address 0x019-0x01D. The following table illustrates the options:

Boot Device	Definition
0x00	Poll devices for the SunOS operating system (i.e. xy, sd, etc.)
0x12	Use EEPROM specified boot device

Address[0x018]:

Boot Device

Boot Device

These five bytes provide for installation of a command string that will boot the operating system from a specified device when EEPROM address 0x018 is set to 0x12, and the diagnostics switch is set to NORM. The locations are assigned as follows:

Address	Definition
0x019	Default boot device (1st character converted to hex)
0x01A	Default boot device (2nd character converted to hex)
0x01B	Controller number in Hex
0x01C	Unit number in Hex
0x01D	Partition number in Hex

The following converts the current boot devices from ASCII to Hex:

Boot Device	Address[0x019]	Address[0x01A]
xy: Xylogics 450/451 Disk	0x78	0x79
sd: SCSI Disk	0x73	0x64
xd: Xylogics Disk (7053)	0x78	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address[0x019-0x01D]:

Device
Device
Controller
Unit
Partition

Keyboard Type

This byte is to signify a NON-SUN keyboard type. It is currently ignored by the Boot PROM.

Address[0x01E]:

Keyboard

Primary Terminal

This byte selects the appropriate device to use as the primary terminal or user interface. The following table illustrates the options:

Terminal	Definition
0x00	Use B/W Monitor (monochrome on-board frame buffer)
0x10	Use Serial Port A
0x11	Use Serial Port B
0x12	Use Color Monitor (CG2, CG3, CG5, or color daughter board)
0x20	Use "first" head of multi-headed P4 card

Address[0x01F]:

Terminal

Display Sun Banner

This byte selects whether to display the Sun banner or custom banner on the screen when booting. The custom banner is defined in a 80-byte character buffer at EEPROM addresses [0x068-0x0B8]. See the paragraphs *Custom Logo Selector* and *Custom Logo* near the end of this chapter for the location and selection of a bit-mapped image that replace the Sun logo. The following table illustrates the banner options:

Value	Definition
0x00	Display the Sun Banner
0x12	Display Custom Banner

Address[0x020]:

Value

Keyboard Click

This byte selects whether the Sun-3 keyboard should be initialized with its key click option on or off. The following table illustrates the options:

Click	Definition
0x00	Turn click OFF
0x12	Turn click ON

Address[0x021]:

Click

Diagnostic Boot Device

These five bytes define the device that the Boot PROM will use when the Diagnostic switch is ON (in DIAG position). The following table illustrates the boot device specification:

Address	Definition
0x022	Default boot device (1st character in hex)
0x023	Default boot device (2nd character in hex)
0x024	Controller number in Hex
0x025	Unit number in Hex
0x026	Partition number in Hex

The following converts the boot devices from ASCII to Hex:

Diagnostic Boot Device	Address[0x022]	Address[0x023]
xy: Xylogics 450/451 Disk	0x78	0x79
xd: Xylogics Disk (7053)	0x78	0x64
sd: SCSI Disk	0x73	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address[0x022-0x026]:

Device
Device
Controller
Unit
Partition

Diagnostic Boot Path

These 40 bytes represent a character buffer for a user specified diagnostic path (/stand/diag, for example). These ASCII characters are represented by Hex values, You would first open address 0x028 and enter the hexadecimal equivalent of the first character in the selected path, and continue on in that manner, ending with 0x00. An ASCII to Hex conversion chart is included at the back of this manual for your convenience.

Address[0x028-0x04F]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	0x00		

High Res Screen Size

These 2 bytes allow the selection of the number of columns and number of rows for the high resolution monitor.

Address[0x050-0x051]:

# of columns
# of rows

The default entries are 50 for # of columns (the hexadecimal value for 80 columns), and 22 for # of rows (the hexadecimal value for 34 rows).

SCC Port A Default Baud Rate

This byte selects whether SCC Port A will use the default baud rate of 9600 Baud or the user specified baud rate defined in the Port A Baud Rate EEPROM address[0x059-0x05A]. The following table illustrates the options:

Value	Definition
0x00	Use Default Baud rate of 9600 Baud
0x12	Use EEPROM defined Baud rate

Address[0x058]:

Value

SCC Port A Baud Rate

These two bytes define the baud rate at which SCC Port A is initialized if EEPROM address[0x058] has been set to 0x012. These bytes are the hexadecimal equivalent of the baud rate. The following table illustrates hexadecimal equivalents to the various baud rates.

Baud Rate	Hex Equivalent	Address[0x059]	Address[0x05A]
300	0x012C	0x01	0x2C
600	0x0258	0x02	0x58
1200	0x04B0	0x04	0xB0
2400	0x0960	0x09	0x60
4800	0x12C0	0x12	0xC0
9600	0x2580	0x25	0x80
19200	0x4B00	0x4B	0x00
38400	0x9600	0x96	0x00

Address[0x059-0x05A]:

Baud (High byte)
Baud (Low byte)

SCC Port A DTR/RTS

This byte selects whether SCC Port A will have the signals DTR and RTS asserted in the initialization process. The following table illustrates the options:

Value	Definition
0x00	Assert the DTR and RTS signals
0x12	Do NOT assert the DTR and RTS signals

Address[0x05B]:

Value

SCC Port B Default Baud Rate

This byte selects whether SCC Port B will use the default baud rate of 9600 Baud or the user specified baud rate defined in the Port B Baud Rate EEPROM address[0x061-0x062]. The following table illustrates the options:

Value	Definition
0x00	Use Default Baud rate of 9600 Baud
0x12	Use EEPROM defined Baud rate

Address[0x060]:

Value (0x00 or 0x12)

SCC Port B Baud Rate

These two bytes define the Baud rate at which SCC Port B is initialized if EEPROM address 0x060 has been set to 0x012. These bytes are the Hexadecimal equivalent of the Baud rate. The following table illustrates how to specify the Baud rate:

Baud Rate	Hex Equivalent	Address[0x061]	Address[0x062]
300	0x012C	0x01	0x2C
600	0x0258	0x02	0x58
1200	0x04B0	0x04	0xB0
2400	0x0960	0x09	0x60
4800	0x12C0	0x12	0xC0
9600	0x2580	0x25	0x80
19200	0x4B00	0x4B	0x00
38400	0x9600	0x96	0x00

Address[0x061-0x062]:

Baud (High byte)
Baud (Low byte)

SCC Port B DTR/RTS

This byte selects whether SCC Port B will have the signals DTR and RTS asserted in the initialization process. The following table illustrates the options:

Value	Definition
0x00	Assert the DTR and RTS signals
0x12	Do NOT assert the DTR and RTS signals

Address[0x063]:

Value

Custom Banner

These 80 bytes represent a character buffer for a user specified custom banner to be displayed instead of the Sun banner, when the value of EEPROM location is 0x020 is 0x012. All locations up to the terminator (0x00) are displayed; each byte not filled with the hexadecimal equivalent of an ASCII character should contain zeroes.

Address[0x068-0x0B7]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	0x00				

Test Pattern

These two bytes are used to provide a known data test pattern to check the EEPROM data lines.

Address[0x0B8-0x0B9]:

0xAA	0x55

System Configuration

There are currently 208 bytes used to represent the hardware configuration of the system. The system configuration is divided into 12 ''slot-configuration'' blocks of 16 bytes each, and 1 sentinel block of 16 bytes to denote the end of the configuration table. The first 16-byte block represents the board specific information in slot 1 of the card cage. The second 16-byte block represents slot 2, and so on.

All empty slots are identified as Board Type Zero (the value 0x00 is in the first byte of that block). The Sentinel Block (0xFF) resides immediately after the configuration block for the last board in the system. The sentinel block should be the 2nd configuration block in a 1-slot system, 4th block for a 3-slot, and so on.

The first byte of each block identifies the type of board that occupies that slot. The information contained in the other bytes of each block may vary, as shown on the following pages.

The layout of the system configuration blocks is illustrated in the following table. B/S means that the contents of that byte are board specific. N/U means ''not used''. The value in each byte depends on what type of board is in which slot. Each block always begins with the ''Board Type'' value. Logically, the first address, 0x0BC, would begin the CPU board block, because the CPU board is always in Slot 1, meaning that the ''Board Type'' value would be 0x01. ''Board Type'' values are shown in Table 14-3 .

This table shows each offset address in the configuration block, followed with a colon and the type of information represented in that byte. 0x0BC is the first byte of the block representing the board in Slot 1, and, in a 12-slot system, 0x16C is the first byte representing the board in Slot 12. The beginning address for each block is shown in boldfaced type: **0BC**.

"0BD:B/S" means that the contents of that byte are board specific; N/U means "not used". If you turn to the CPU Board 16-byte representation, you will see that this byte would contain the hexadecimal value for the amount of memory on the CPU board, which will vary.

Table 14-2 *Configuration Block Layout Address[0x0BC-0x18B]*

0BC:Board Type	0BD:B/S	0BE:B/S	0BF:B/S	0C0:B/S	0C1:B/S	0C2:B/S	0C3:B/S
0C4:B/S	0C5:B/S	0C6:B/S	0C7:B/S	0C8:B/S	0C9:B/S	0CA:B/S	0CB:B/S
0CC:Board Type	0CD:B/S	0CE:B/S	0CF:B/S	0D0:B/S	0D1:B/S	0D2:B/S	0D3:B/S
0D4:B/S	0D5:B/S	0D6:B/S	0D7:B/S	0D8:B/S	0D9:B/S	0DA:B/S	0DB:B/S
0DC:Board Type	0DD:B/S	0DE:B/S	0DF:B/S	0E0:B/S	0E1:B/S	0E2:B/S	0E3:B/S
0E4:B/S	0E5:B/S	0E6:B/S	0E7:B/S	0E8:B/S	0E9:B/S	0EA:B/S	0EB:B/S
**0EC:Board Type	0ED:B/S	0EE:B/S	0EF:B/S	0F0:B/S	0F1:B/S	0F2:B/S	0F3:B/S
0F4:B/S	0F5:B/S	0F6:B/S	0F7:B/S	0F8:B/S	0F9:B/S	0FA:B/S	0FB:B/S
0FC:Board Type	0FD:B/S	0FE:B/S	0FF:B/S	100:B/S	101:B/S	102:B/S	103:B/S
104:B/S	105:B/S	106:B/S	107:B/S	108:B/S	109:B/S	10A:B/S	10B:B/S
10C:Board Type	10D:B/S	10E:B/S	10F:B/S	110:B/S	111:B/S	112:B/S	113:B/S
114:B/S	115:B/S	116:B/S	117:B/S	118:B/S	119:B/S	11A:B/S	11B:B/S
11C:Board Type	11D:B/S	11E:B/S	11F:B/S	120:B/S	121:B/S	122:B/S	123:B/S
124:B/S	125:B/S	126:B/S	127:B/S	128:B/S	129:B/S	12A:B/S	12B:B/S
12C:Board Type	12D:B/S	12E:B/S	12F:B/S	130:B/S	131:B/S	132:B/S	133:B/S
134:B/S	135:B/S	136:B/S	137:B/S	138:B/S	139:B/S	13A:B/S	13B:B/S
13C:Board Type	13D:B/S	13E:B/S	13F:B/S	140:B/S	141:B/S	142:B/S	143:B/S
144:B/S	145:B/S	146:B/S	147:B/S	148:B/S	149:B/S	14A:B/S	14B:B/S
14C:Board Type	14D:B/S	14E:B/S	14F:B/S	150:B/S	151:B/S	152:B/S	153:B/S
154:B/S	155:B/S	156:B/S	157:B/S	158:B/S	159:B/S	15A:B/S	15B:B/S
15C:Board Type	15D:B/S	15E:B/S	15F:B/S	160:B/S	161:B/S	162:B/S	163:B/S
164:B/S	165:B/S	166:B/S	167:B/S	168:B/S	169:B/S	16A:B/S	16B:B/S
16C:Board Type	16D:B/S	16E:B/S	16F:B/S	170:B/S	171:B/S	172:B/S	173:B/S
174:B/S	175:B/S	176:B/S	177:B/S	178:B/S	179:B/S	17A:B/S	17B:B/S
*17C:FF	17D:N/U	17E:N/U	17F:N/U	180:N/U	181:N/U	182:N/U	183:N/U
184:N/U	185:N/U	186:N/U	187:N/U	188:N/U	189:N/U	18A:N/U	18B:N/U

* This is the Sentinel Block ** Sentinel Block goes here for 3-slot system

Possible "Board Type" values for the first byte of the board configuration blocks are defined in the following table. Note that these are Board Type definitions and not system slot assignments, or addresses. For slot assignments, refer to the appropriate *Cardcage Slot Assignments and Backplane Configuration* document.

The value shown under ***Type*** in the table below is entered in the first byte of the configuration block that represents that board. The beginning EEPROM address of the configuration block depends on the slot assignment (refer to the representation on the previous page).

Table 14-3 *Board Type Values*

Type	*Board Definition*
0x00	None (Empty slot)
0x01	CPU
0x02	Memory
0x03	Color
0x04	Frame Buffer
0x05	FPA
0x06	SMD Disk Controller
0x07	Tape Controller
0x08	Ethernet Controller
0x09	MTI/ALM
0x0A	Graphics Processor (GP)
0x0B	SCP Controller
0x0C	SCSI Controller
0x0D	Integrated Personal Computer (SunIPC)
0x0E	Graphics Board (GB)
0x0F	3/75 SCSI with Memory
0x10	MAPKIT
0x11	Channel Adapter
0x12	ALM-2
0x13	MCP/SCP-2
0x14-0x7F	Reserved Sun Hardware
0x80-0xFE	Reserved Non-Sun Hardware
0xFF	Sentinel Block

Each configuration block is graphically represented on the following pages to show the values that must be present in the sixteen bytes to represent the various Sun PC Boards. Each byte is shown as a box, filled with the type of information represented by the value in that byte.

Table 14-4 **No-Board Configuration Block**

1st Byte	0x00
2nd Byte	0x00
3rd Byte	0x00
4th Byte	0x00
5th Byte	0x00
6th Byte	0x00
7th Byte	0x00
8th Byte	0x00
9th Byte	0x00
10th Byte	0x00
11th Byte	0x00
12th Byte	0x00
13th Byte	0x00
14th Byte	0x00
16th Byte	0x00

Use this block to represent an empty slot. For example, assume that the block beginning with EEPROM address 0x0BC (the first block) contains the CPU board configuration values, because the CPU board is in Slot 1. Let us say that Slot 2 is empty. In order to represent this, the second configuration block, beginning with address 0x0CC, would contain 0x00 in each of the next sixteen bytes, up to and including address 0x0DB.

Table 14-5 **CPU Board Configuration Block**

1st Byte:	Board Type: 0x01
2nd Byte:	# Mbytes Memory in Hex
3rd Byte:	Installed Options: Bit 0 = 1 (FPU) Bit 1 = 1 (DCP/DES) Bit 2 = 0 Bit 3 = 0 Bit 4 = 0 Bit 5 = 0 Bit 6 = 0 Bit 7 = 0
4th Byte:	# Kb of Cache (hex)
5th Byte:	Sun-3/60 Color Frame Buffer 0x00 = No Board 0x01 = Hi Res B/W 0x02 = Lo Res Color 0x03 = Hi Res Color
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

How to Program the CPU Configuration Block

Because the CPU board is always in Slot 1, this example will use the actual EEPROM offset addresses of each byte in the CPU board configuration block. If the CPU board were in another slot, the addresses shown would depend on the slot assignment, as shown in Table 14-2 .

However, it is likely that, if you open EEPROM location 0x0BC, the value in the first byte will be 0x01, to represent the CPU board:

```
>q 0bc Return
```

```
EEPROM 0BC: 01?  Return
```

The next location (Byte 2) should now be displayed. It represents how much memory is on the CPU board. Because the Sun-4 CPU board has no on-board memory, the display should read:

```
EEPROM 0BD: 00 ? [Return]

EEPROM 0BE:01 ?
```

The content of the next location, assigned to "Installed Options", is now displayed.

To fill in the "Installed Options" byte, you must convert the binary value — represented by a one or zero in Bits 0 through 7 — to a hexadecimal value. For example, if the CPU board contains both a Floating Point and a DCP chip, the binary value would be:

```
00000011
```

Converted to hexadecimal, the value in the third byte would be 0x03.

If only a Floating Point Chip is present, the value would be 0x01. If only a DCP chip is present, the binary value would be

```
00000010
```

meaning that the hexadecimal value in the third byte would be 0x02. And, of course, if neither option is present, the value in the third byte would be 0x00.

If you were to change the "Installed Options" value, you would simply enter the new value after the question mark, and press [Return] to view the fourth byte, which represents the quantity of cache memory present on the CPU board.

If, at any time during this process, you want to return to the monitor prompt, enter a non-hexadecimal character after the new value you just entered, BEFORE pressing [Return]. Refer to the *PROM Monitor Commands* chapter for more information on procedures for modifying memory locations.

In this example, the CPU board did not have a Floating Point Unit, and has been upgraded to include that chip. No data encryption chip is present. You would enter:

```
EEPROM 0BE:00 ? 01 [Return]

EEPROM 0BF:00 ?
```

The fourth byte (location 0x0BF) would contain the value 0x40 if there were 64 Kilobytes of cache memory on the CPU board.

As shown in the CPU board configuration example, the fifth byte (location 0x0C0) is reserved for Sun-3/60 configuration information. The values in this byte tell us whether or not a Color Frame Buffer board is attached to the 3/60 CPU board, and whether the board interfaces with a high or low resolution, black and white or color monitor. A 3/60 system with a High Resolution Black and White Monitor would show this value when you increment to the fifth byte:

```
EEPROM 0C0:01 ?
```

Table 14-6 **Memory Board Configuration Block**

1st Byte	0x02
2nd Byte	# Mbytes of Memory in hex
3rd Byte	Reserved
4th Byte	Reserved
5th Byte	Reserved
6th Byte	Reserved
7th Byte	Reserved
8th Byte	Reserved
9th Byte	Reserved
10th Byte	Reserved
11th Byte	Reserved
12th Byte	Reserved
13th Byte	Reserved
14th Byte	Reserved
15th Byte	Reserved
16th Byte	Reserved

Table 14-7 **Color Board Configuration Block**

1st Byte:	0x03
2nd Byte:	Type (2,3 or 5) 0x2 = Sun2 0x3 = Sun3 0x5 = CG5
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-8 **Frame Buffer Board Configuration Block**

1st Byte:	0x04
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-9 **FPA Board Configuration Block**

1st Byte:	0x05
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-10 **SMD Disk Board Configuration Block**

1st Byte:	0x06
2nd Byte:	Manufacturer 1 = Xylogics 450 2 = Xylogics 451
3rd Byte:	Controller #
4th Byte:	# of Disks
5th Byte:	Drive #0 Capacity 0 = No Disk 1 = 8" 130 Mb (450/451) 2 = 8" 280 Mb (451) 3 = 10.5" 380 Mb (450/451) 4 = 10.5" 575 Mb (451)
6th Byte:	Drive #1 Capacity
7th Byte:	Drive #2 Capacity
8th Byte:	Drive #3 Capacity
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-11 **1/2 Inch Tape Controller Configuration Block**

1st Byte:	0x07
2nd Byte:	Manufacturer 1 = Xylogics (472) 2 = Ciprico (TM1000)
3rd Byte:	Controller #
4th Byte:	# of Tape Drives
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-12 Second Ethernet Controller Board Configuration Block

(For the LAN Gateway Interface)

1st Byte:	0x08
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-13 **MTI/ALM Board Configuration Block**

1st Byte:	0x09
2nd Byte:	# Terminals
3rd Byte:	Manufacturer 1= Systech 2 = Sun Microsystems
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-14 **GP Board Configuration Block**

1st Byte:	0x0A
2nd Byte:	Type (Plus or 2) 0x1 = GP+ 0x2 = GP2
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-15 **SCP Board Configuration Block**

1st Byte:	0x0B
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-16 **SCSI Board Configuration Block**

1st Byte:	0x0C
2nd Byte:	Type (2 or 3) 0x2 = Sun2 0x3 = Sun3
3rd Byte:	# of Tape Drives
4th Byte:	# of Disk Drives
5th Byte:	Tape Controller 0x1 = Sysgen 0x2 = MT02
6th Byte:	Disk Controller 0x1 = MD21 0x2 = Adaptec
7th Byte:	Disk Drive #0 Capacity 0x1 = 5.25" 71 Mb 0x2 = 5.25" 141 Mb 0x3 = 5.25" 327 Mb
8th Byte:	Disk Drive #1 Capacity
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-17 **SunIPC Board Configuration Block**

1st Byte:	0x0D
2nd Byte:	Reserved
3rd Byte:	Math Coprocessor Option: 0 = absent 1 = installed
4th Byte:	Disk Drive Option * 0 = None 1 = Single 5 1/4" Floppy 2 = Dual 5 1/4" Floppy
5th Byte:	Drive 0 Capacity 0 = None 1 = r/w 1.2 MB, read 360 KB 2 = r/w 1.2 MB
6th Byte:	Drive 1 Capacity 0 = None 1 = r/w 360 KB
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

NOTE ** The disk drive option can readily be moved to other SunIPC boards and therefore this information may not always be correct.*

Table 14-18 **GB Board Configuration Block**

1st Byte:	0x0E
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-19 **3/75 SCSI Memory Board Configuration Block**

1st Byte:	0x0F
2nd Byte:	# MB of Memory (Hex) 0x00, 0x02 or 0x04
3rd Byte:	Type of SCSI 0x02 = Sun-2 0x03 = Sun-3
4th Byte:	# of Tape Drives (hex)
5th Byte:	# of Disk Drives (hex)
6th Byte:	Tape Controller 0x01 = Sysgen 0x02 = MT02
7th Byte:	Disk Controller 0x01 = MD21 0x02 = Adaptec
8th Byte:	Disk Drive #0 Capacity 0x01 = 5.25" 71 MB 0x02 = 5.25" 141 MB
9th Byte:	Disk Drive #2 Capacity
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-20 **MAPKIT Configuration Block**

1st Byte:	0x10
2nd Byte:	1=INI
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-21 **Channel Adapter Configuration Block**

1st Byte:	0x11
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-22 **ALM-2 Configuration Block**

1st Byte:	0x12
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-23 **MCP/SCP-2 Configuration Block**

1st Byte:	0x13
2nd Byte:	Reserved
3rd Byte:	Reserved
4th Byte:	Reserved
5th Byte:	Reserved
6th Byte:	Reserved
7th Byte:	Reserved
8th Byte:	Reserved
9th Byte:	Reserved
10th Byte:	Reserved
11th Byte:	Reserved
12th Byte:	Reserved
13th Byte:	Reserved
14th Byte:	Reserved
15th Byte:	Reserved
16th Byte:	Reserved

Table 14-24 **Sentinel Block**

1st Byte:	0xFF
2nd Byte:	Not Used
3rd Byte:	Not Used
4th Byte:	Not Used
5th Byte:	Not Used
6th Byte:	Not Used
7th Byte:	Not Used
8th Byte:	Not Used
9th Byte:	Not Used
10th Byte:	Not Used
11th Byte:	Not Used
12th Byte:	Not Used
13th Byte:	Not Used
14th Byte:	Not Used
15th Byte:	Not Used
16th Byte:	Not Used

Key Table Selector

The value of this byte is used to determine the appropriate key tables to be used based on the following table. EEPROM key tables refer to the tables entered in EEPROM addresses 0x190-0x20F (lower case) and 0x210-0x28F (upper case).

Key Table Selector	Definition
0x58	Use EEPROM key tables
Other than 0x58	Use PROM Key tables

Address [0x18C]:

Key table selector

EEPROM Locale Specifier

This byte contains the specifier for the locale (country, language, codeset) for which the system is configured. The specifier is provided by the SunOS operating system.

Address [0x18D]:

Locale Specifier

Keyboard ID

The Boot PROM checks the EEPROM Key Table Selector (address 0x18C), and if the value is 0x58, it then compares the value in the address 0x18E with the keyboard type it finds in the system. This byte contains the hard coded keyboard type. The firmware uses this value only in the case of an EEPROM key table request.

Address [0x18E]:

Keyboard ID

Custom Logo Selector

The value of this byte is used to determine the appropriate Logo bit-map to be displayed upon power-up, based on the following table.

Custom Logo Selector	*Definition*
0x12	Use EEPROM logo bit-map (see Custom Logo, below)
Other than 0x12	Use Sun logo bitmap

Address [0x18F]:

Custom Logo Selector

EEPROM Lower Case Key table

An array of 128 bytes from address 0x190 can be used for a different lower case key table. This table is used by the firmware if location 0x18C is set to 0x58 and Keyboard ID matches the hard coded ID.

Address [0x190-0x20F]:

128 bytes for lower case key table

EEPROM Upper Case Key Table

An array of 128 bytes from address 0x210 can be used for a different upper case key table. This table is used by the firmware if location 0x18C is set to 0x58 and Keyboard ID matches the hard coded ID.

Address [0x210-0x28F]:

128 bytes for upper case key table

Custom Logo

This 64x8 matrix may contain a Custom Logo bit-map that can be selected for power-up display by setting location 0x18F to 0x12.

Address [0x290-0x48F]:

64X8 bytes Custom Logo

Reserved Area

Write Count

These write counters are for the Reserved area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x500-0x505]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Reserved Area

Checksum

Each EEPROM area maintains three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x508-0x50A]:

Checksum #1
Checksum #2
Checksum #3

ROM Area

Write Count

These write counters are for the ROM area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x600-0x605]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

ROM Area

Checksum

Each EEPROM area maintains three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x608-0x60A]:

Checksum #1
Checksum #2
Checksum #3

Software Area

Write Count

These write counters are for the Software area of the EEPROM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address[0x700-0x705]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Software Area

Checksum

Each EEPROM area will maintain three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address[0x708-0x70A]:

Checksum #1
Checksum #2
Checksum #3

15

Sun386i Self-tests and Initialization

Sun386i Self-tests and Initialization **295**

15.1. Hardware Requirements 295
15.2. Memory Test Options 296
15.3. Functional Requirements 296
15.4. Power-On Theory of Operation 298

15

Sun386i Self-tests and Initialization

15.1. Hardware Requirements

In order to perform the power-up tests, the following criteria must be met. The CPU must be functional and the ability to fetch instructions from the Boot PROM must be intact.

The Power-On Self Test (POST) sequence has no dependence upon the presence of video capability. If the serial port is not attached to a terminal, all test numbers and error status will still be communicated to the LEDs on the CPU board at the back of the workstation. Additional hardware requirements are as follows:

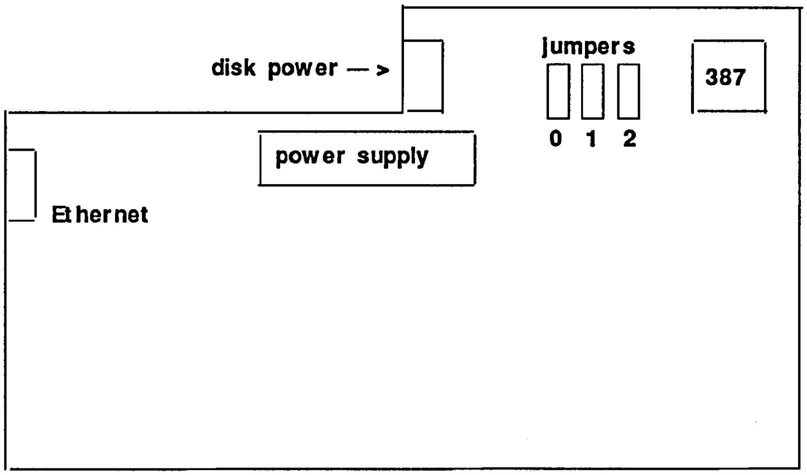
1. a minimum of 128K bytes of programmable read only memory (PROM)
2. eight diagnostic LEDs, visible to user in the event of self test errors when the workstation console or terminal display may not be presumed to work.
3. Ethernet, disk, tape, or communications line to boot the SunOS, the Diagnostic Executive, or a stand-alone program.
4. a serial cable and terminal for video monitor interaction if no video card is present.
5. a correctly programmed IDPROM to boot the SunOS operating system over Ethernet.

CPU Board Jumper Settings

The physical settings are as follows. These settings either affect the hardware directly or cause the PROM to run in a different mode. For more information, refer to the section on Determination of Run-time Mode.

If you are facing the Ethernet and Serial Port connector side of the board, the three jumper positions are located on the opposite side of the board, beyond the power supply and between the disk power connector and the 80387 chip. The pins normally have no jumper installed. To set the system for a diagnostic boot, install a jumper in position 0 (labeled W600). Position 2 (labeled W602) is for manufacturing mode and is not for the customer's use. Jumper position 1 (labeled W601) is reserved for future use.

Figure 15-1 *Sun386i Jumper Location*



15.2. Memory Test Options

During a normal power-on sequence, the PROM uses configuration information in the NVRAM to determine how much memory to test. (The NVRAM performs the same function as the Sun-3 and Sun-4 EEPROM.)

If the CPU board jumpers are set accordingly, or if the NVRAM soft switch is set, the Power-On Self-Tests (POST) execute in ''diagnostic mode''. This mode results in a more complete (and lengthy) testing of the system, and the RAM in particular. When in Diagnostic Mode, the time duration of the POST sequence is dependent only upon the amount of RAM in the system.

15.3. Functional Requirements

The Sun386i CPU Boot PROM is responsible for many system functions both during and after the boot sequence. The primary duty during the Power-On phase is, of course, to verify minimal functional integrity such that the boot code can load either the operating system or some other stand-alone code. The PROM must also provide an interactive user interface on several levels, from the LEDs up to serial and video I/O drivers for the PROM monitor.

In addition, because many of the PROM based functions are useful to system programs, a vector table is provided so that these programs can access these utilities without needing to duplicate code.

Normal Boot

During a ''Normal'' boot, the ''progress meter'' on the console display fills in to let you know that testing is in progress. When the rectangle is half-way filled in, self-tests are over, and a successful operating system boot-up causes the rest of the rectangle to fill in. The console screen would also notify you if an error occurred. If you connected a terminal to the serial port during a ''Normal'' boot, POST error messages would be displayed on the terminal screen.

During the POST sequence, the current test number will be shown on the LED bank of the CPU board. Thus, if a fatal error occurs and the operating system cannot boot, the LEDs will show which test caused the failure.

The boot device polling sequence for the Sun386i during a default (normal) boot is:

(a) Floppy Disk

(b) SCSI Unit 4 - 1/4-inch tape drive in the peripheral box

(c) SCSI Unit 2 - winchester disk in the system box

(d) SCSI Unit 0 - winchester disk in the peripheral box

(e) Ethernet

The boot device is polled to verify that it exists and then the boot block code is read and verified. Note that you can configure the NVRAM to bypass the polling sequence and instead use a boot device specified in the NVRAM. The NVRAM layout chapter describes how to do this.

Refer to the PROM Monitor Command chapter for information on using the `b` command to boot a specific program, using a specific path.

Booting of the SunOS operating system is executed after the POST sequence if all of the following conditions are satisfied:

(a) the CPU board jumpers and the NVRAM soft switch are set for normal mode.

(b) no fatal errors are detected during the POST sequence.

Fatal errors are announced through the LEDs, speaker, video and serial ports, and then a scope loop is entered. Non-fatal errors are announced in the same way, but do not stop execution of the boot sequence. Any error encountered is noted in the NVRAM so that the program to be booted can use that information if necessary.

(c) you do not press the [Break] key (on a terminal) or execute an `L1-A` sequence (on a Sun keyboard) during the boot sequence.

CAUTION **Aborting the boot causes the PROM monitor program to be invoked, and should not be used if the SunOS operating system is already running. On systems that have a disk, aborting the SunOS in that manner may cause damage to the file systems.**

Diagnostic Boot

During a "Diagnostic" boot, POST error messages would be visible on both the console screen and that of a terminal attached to the serial port. POST status is reported in the form of the "progress meter" on the console screen, and in ASCII form on a terminal attached to the serial port.

The Diagnostic boot path, after executing self-test, permits additional testing of boot paths and other hardware functions through operator action, or, if no operator action is taken, performs a default boot of the Diagnostic Executive.

You may also configure the NVRAM to boot the program of your choice during a diagnostic boot. Refer to the NVRAM layout chapter for more information.

15.4. Power-On Theory of Operation

The following paragraphs explain what is happening internally during power-on or reset, and how different modes function.

System Reset Analysis: Hard and Soft Resets

When the Sun386i CPU Board is powered-on or reset, the CPU begins execution at the beginning of the POST code which is located after both the "Sunromvec" table and the Interrupt Handler Vector table.

The program then switches the Sun386i to protected mode and determines whether the reset was a hard reset (i.e. power on) or a soft reset (i.e. the system was running and then was reset internally without being turned off).

If the PROM determines that the reset was soft, the POST sequence will be skipped and the SunOS operating system will be rebooted without changing the state of any system devices. If the PROM determines that the system was just turned on and is totally unverified and uninitialized, the POST sequence will both initialize and verify Sun386i functionality.

Determination of Run-time Mode

Upon starting the POST sequence, the PROM looks at the system jumpers and NVRAM based "soft switch" setting to determine which run mode to use. The order of priority to determine the run mode is 1) CPU board jumpers and 2) NVRAM soft switch settings. This order is necessary to insure that the Sun386i POST can always be forced to run in any desired mode, regardless of system condition. If there are no jumpers installed on the CPU board (i.e. normal mode), the PROM uses a valid soft switch setting to determine its POST run-time mode.

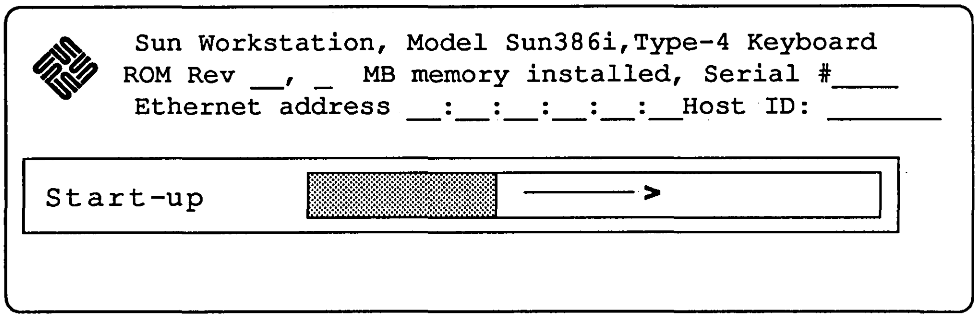
Normal (Default) Mode

If the Run-time Mode is set to normal, the system is assumed to be functioning correctly. In that case, the two-phase self-test is executed, and the loader attempts to load the SunOS operating system upon completion of the tests.

During self-test, memory will be sized and memory parity initialized. However, the amount of memory tested will be a function of a parameter within the NVRAM on the CPU board that specifies the amount of memory to be tested.

The following "progress meter" appears on the workstation's video screen to indicate the progress of the POST sequence:

Figure 15-2 *Sample Normal Sun386i Start-Up Display*



```
Sun Workstation, Model Sun386i,Type-4 Keyboard
ROM Rev __, _  MB memory installed, Serial #____
Ethernet address __:__:__:__:__:__Host ID: _______

Start-up                      ———————>
```

Diagnostic Mode

Diagnostic mode causes the self-test to be executed similarly as before, except all of memory is tested and self-test status information is directed to the serial port until all hardware required for the video monitor (console) has been successfully tested.

Any fatal hardware failures during the self-tests will invoke scope loops to permit troubleshooting the failure. You may program the NVRAM to loop forever on any of the POST tests. (Refer to the NVRAM Layout section for information on this option).

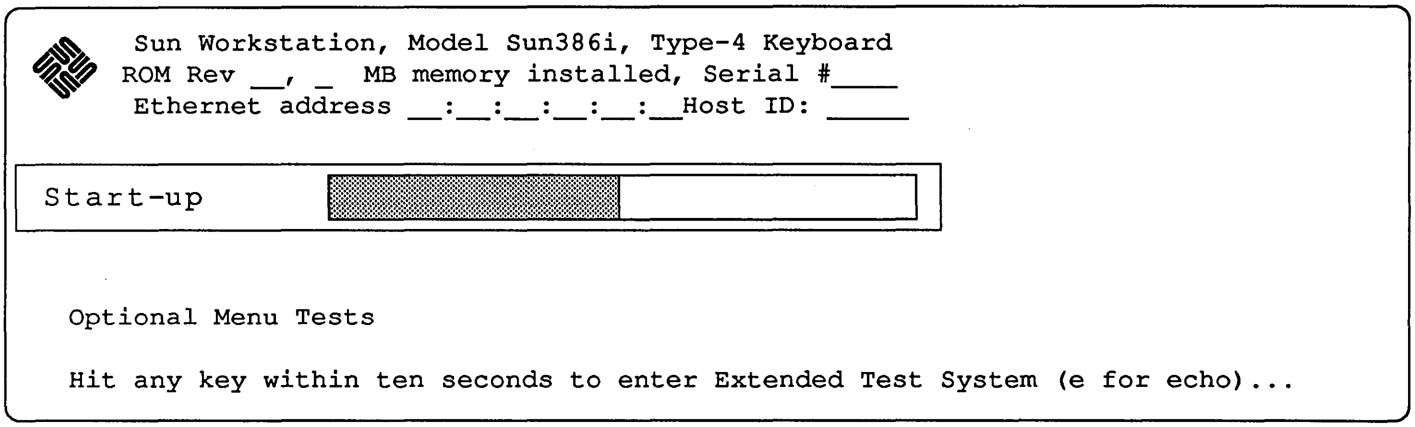
An RS-232 terminal with its characteristics set to 9600 Baud, 8 data bits, 1 stop bit and no parity should be connected to the serial port on the CPU board if you wish to view self-test status and interact with the system when the workstation video display or keyboard is not functional.

Limited interaction with the self-test program is possible by way of the serial port in this mode. The following input characters may be used during POST:

1. Pressing the `b` (a mnemonic for *burn-in*) key while the self-tests are in progress executes the POST sequence forever. This action is analogous to setting the Run-time mode jumper to Manufacturing Mode.
2. Pressing the `s` key while the self-tests are in progress immediately restarts the POST sequence.
3. Given that one of the POST tests failed, it will continue to re-execute forever unless interrupted. Pressing the space bar terminates the failed test and the begins the next test.
4. By default, an unsuccessful power-up test prints one error message before entering an infinite scope loop. Once inside of the scope loop, the failing test will *not* print any more messages. However, to see test messages while in the scope loop, you may press the `p` key. Subsequent pressing of the `p` key will act like a toggle switch for the message mode.
5. Pressing the Esc key during the POST sequence skips all of the remaining power-up tests.

During the POST phase, the PROM displays the standard power-on screen. After the self-test has completed successfully, the screen will look like the following:

Figure 15-3 *Sample Diagnostic Mode Sun386i Start-Up Display*



```
   Sun Workstation, Model Sun386i, Type-4 Keyboard
  ROM Rev __, _  MB memory installed, Serial #____
   Ethernet address __:__:__:__:__:__Host ID: _____

 Start-up

   Optional Menu Tests

   Hit any key within ten seconds to enter Extended Test System (e for echo)...
```

If you do not press a terminal key (either video console or serial port terminal) within the 10 second delay time, the program will perform a default boot of the Diagnostic Executive program.

However, if you *do* press a terminal key during this time, the Extended Test Menu is displayed on the console screen, from which you may select any one of a number of Extended Test options that permit testing such things as memory, serial ports, the video display and the Ethernet controller, as well as test booting from any of the peripheral I/O boot devices. See *Chapter 17* for a description of Extended Menu Test Options.

Manufacturing (Burn-In) Mode

Manufacturing mode acts exactly like Diagnostic mode until the end of the POST sequence. At that time, the system resets and starts the self-tests again. This mode is invoked with CPU board jumper settings only, and the only way to exit is to power-down and remove the jumper.

Power On Self Test (POST)

The self-test diagnostics are divided into a sequence of hardware ''crawl-out'' self-tests that are written primarily in assembly language, and menu selectable tests written primarily in ''C'' language, which are executable under the PROM Monitor. The purpose of these diagnostics is to test enough of the system hardware to clear a path to one of the possible boot devices in order to down-load the operating system.

These self-tests assume that only the CPU is functioning correctly and can successfully fetch instructions from the Boot PROM.

The tests begin without the ability to use a stack (i.e. no calls or parameter passing, just relative jumps and register sharing). After the NVRAM has been verified, the POST sets up a stack area in a reserved block of NVRAM so that there is still no dependence on any RAM.

The POST begins with a series of tests that verify various guaranteed reset conditions and start the actual initialization of Sun386i hardware. Further testing continues with various hardware subsystems that are considered essential to normal operation. These subsystems include the relocation register, refresh register, timers, interrupts, DMA channels, NVRAM, several checksums, and at least enough RAM space to load the boot block that contains the second phase of the POST confidence tests.

There are additional PROM based tests that are activated during the Diagnostic and Manufacturing modes to check some areas that can best be tested in the PROM but that are too lengthy for the normal power up sequence. Refresh (Manufacturing Mode) and total memory (Diagnostic Mode) testing are examples of those tests.

Fatal errors are announced (if possible) by the LEDs and by transmitting error messages that consist of a hex number followed by an ASCII string to both the serial port and the video port (if it is enabled). After a fatal error has been announced, the PROM initiates a tight scope loop to facilitate debug and repair.

As soon as the essential system devices have been verified, the PROM loads the second phase of the POST in conjunction with the boot block code located in the file `/boot` on the valid boot device. This code continues the POST sequence by testing RAM, cache functionality, and peripheral devices including the hard disk, floppy, and Ethernet.

Most of the potential errors encountered in this area are considered non-fatal and therefore will not cause the boot sequence to abort.

All are noted in the NVRAM so that down-loaded programs may use that information to help diagnose and repair systems. See the section on Fatal and Warning class errors for more information.

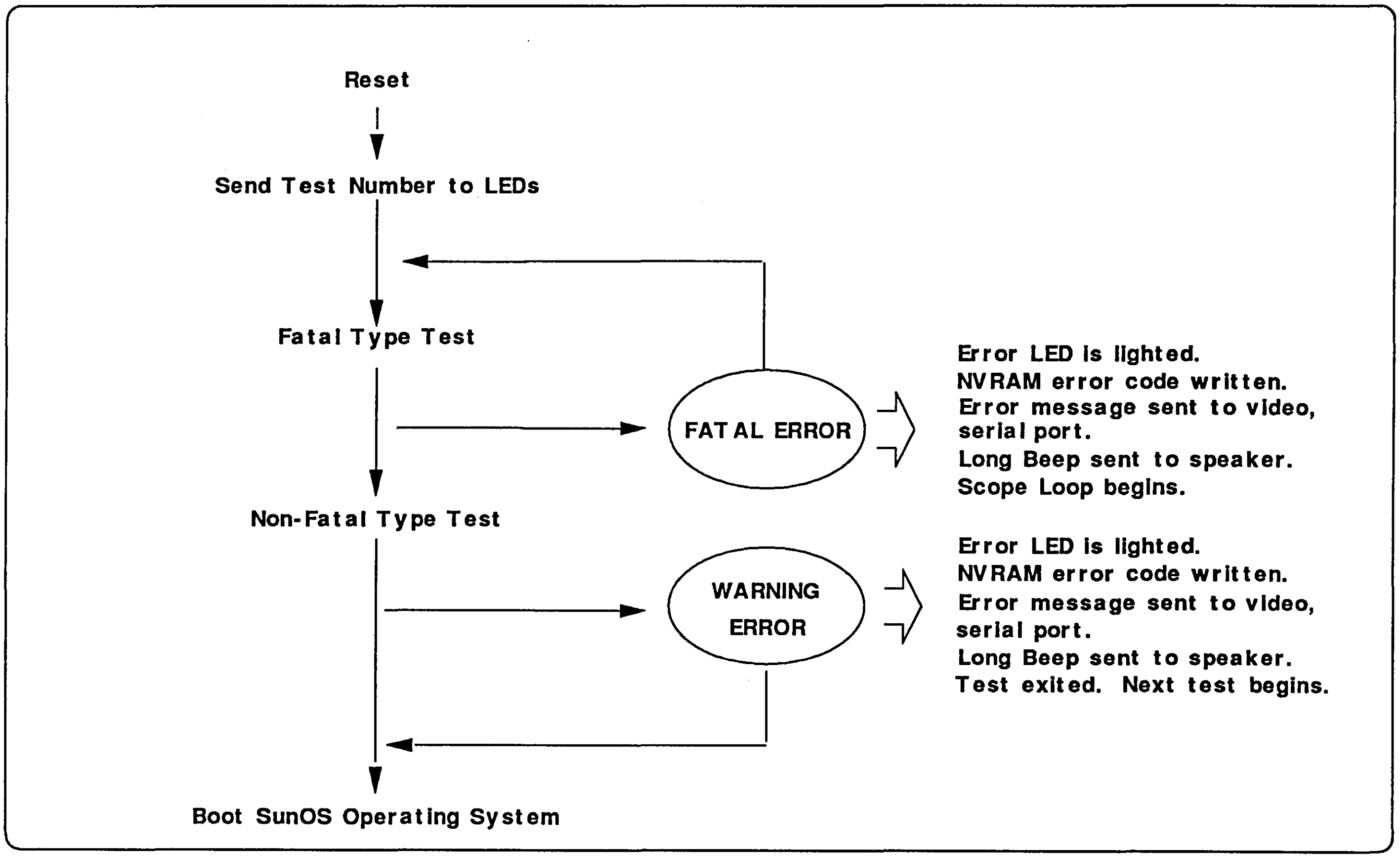
Error Classification and Reporting

The handling of errors found during the POST sequence is based upon the importance associated with the given hardware subsystem. Many of the tests deal with hardware that is considered essential to system usage; any errors found during any such test are considered fatal and result in abortion of the POST sequence so that the error can be "looped-on".

However, some of the POST tests deal with hardware that is not essential, in most cases, to minimum system performance. That is, a typical Sun386i system could boot and "get by" without that functionality. An error in such a non-essential test is announced in much the same way as a fatal error. The difference in the handling of a non-fatal or Warning error is that after the error is announced, the POST continues with the next test and finally attempts to boot the operating system.

The following is a graphic breakdown of the flow of control for the POST error handlers:

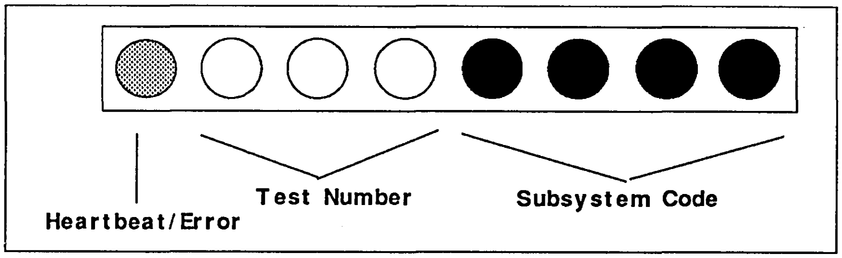
Figure 1-4 *Normal Mode Sun386i POST Error Handling*


Reset
Send Test Number to LEDs
Fatal Type Test
FATAL ERROR
Error LED is lighted.
NVRAM error code written.
Error message sent to video, serial port.
Long Beep sent to speaker.
Scope Loop begins.
Non-Fatal Type Test
WARNING ERROR
Error LED is lighted.
NVRAM error code written.
Error message sent to video, serial port.
Long Beep sent to speaker.
Test exited. Next test begins.
Boot SunOS Operating System

NOTE *Note: Error messages are sent to the console display only if it is enabled.*

The first line of communication for reporting POST status and errors is the LED register that controls the 8 individual LEDs on the CPU board. The CPU board Boot PROM uses the eight LEDs to communicate the current subsystem under test, the individual test being run, as well as a dual heartbeat/error LED to signify that the test is still running or that an error was found.

Figure 1-5 *Sun 386i LED Display Interpretation*


Heartbeat/Error
Test Number
Subsystem Code

The following table explains the subsystem code read from bits 0 through 3.

Table 15-1 *LED Subsystem Code Interpretation*

LED Display	*Subsystem Description*
○○○○○○○○	CPU Board: misc.
○○○○○○○●	CPU Board: 82380 reset defaults
○○○○○○●○	CPU Board: 82380 functional
○○○○○○●●	CPU Board: misc.
○○○○○●○○	Memory Board: Slot 1
○○○○○●○●	Memory Board: Slot 2
○○○○○●●○	Memory Board: Slot 3
○○○○○●●●	Memory Board: Slot 4
○○○○●○○○	Video Board: Slot 4
○○○○●○○●	Mouse/Keyboard
○○○○●○●○	Floppy Disk
○○○○●○●●	SCSI Hard Disk Drive
○○○○●●○○	SCSI Cartridge Tape Drive
○○○○●●○●	Ethernet
○○○○●●●○	Reserved
○○○○●●●●	CPU Board, Reserved

When a test begins, the LED register is updated with the subsystem code and test number. While the test is running, and in fact during the entire POST sequence, the heartbeat LED flashes to show that the system is still alive. If an error occurs during that test, the heartbeat LED stays on to signify that an error has been detected.

If the error is deemed fatal, the LED remains on while the scope loop proceeds. If the error is just a warning, the LED still stays on to signify an error occurred, but the test lights continue to show the numbers of the ongoing tests.

In an effort to insure that you do not miss an error occurrence, the Sun386i speaker also sounds a relatively long beep to announce the error.

To avoid confusion and allow the booted software to analyze the POST results, the PROM writes a copy of the LED register (i.e. subsystem and test number) to a byte in the NVRAM so that the NVRAM will still record the failing test number if a Warning error occurs (where the LED information is lost).

In addition, any error found causes a text error message to be displayed on the serial port and the video screen (if enabled). The message will be in one these formats, depending on the run-time mode:

```
Normal     14: Memory Board: Slot 1 Failed

Diagnostic 14: RAM Data Test Failed. Exp: xxxx Obs: xxxx Addr: xxxx
```

The hexadecimal number is a copy of the value of the least significant 7 bits of the LED register. From this number, the subsystem and test number can be determined. The text that follows will be a short message that briefly describes the failure. These messages are accessed from a table internal to the PROM so that future versions may substitute foreign language strings for ease of internationalization.

The following tables summarize the LED patterns for the power-on self-tests.

Table 15-2 *POST LED Interpretation*

Sub-system#	*Test #*	*LED pattern*	*Hex value*	*Error type*	*FRU (component)*
0	0	○○○○○○○○	0x00	---	-----
0	1	○○○●○○○○	0x10	fatal	CPU Bd (80386)
0	2	○○●○○○○○	0x20	warning	CPU Bd (LED register)
0	3	○○●●○○○○	0x30	fatal	CPU Bd (Boot PROM)
0	4	○●○○○○○○	0x40	warning	CPU Bd (Serial Port)
0	5	○●○●○○○○	0x50	warning	CPU Bd (Parallel Port)
0	6	○●●○○○○○	0x60	fatal	CPU Bd (NVRAM)
0	7	○●●●○○○○	0x70	fatal	CPU Bd (IDPROM)
1	0	○○○○○○○●	0x01	fatal	CPU Bd (82380)
1	1	○○○●○○○●	0x11	fatal	CPU Bd (82380)
1	2	○○●○○○○●	0x21	fatal	CPU Bd (82380)
1	3	○○●●○○○●	0x31	fatal	CPU Bd (82380)
1	4	○●○○○○○●	0x41	fatal	CPU Bd (82380)
1	5	○●○●○○○●	0x51	fatal	CPU Bd (82380)
1	6	○●●○○○○●	0x61	fatal	CPU Bd (82380)
1	7	○●●●○○○●	0x71	warning	CPU Bd (Enet Controller)
2	0	○○○○○○●○	0x02	fatal	CPU Bd (82380)
2	1	○○○●○○●○	0x12	fatal	CPU Bd (82380)
2	2	○○●○○○●○	0x22	fatal	CPU Bd (82380)
2	3	○○●●○○●○	0x32	fatal	CPU Bd (82380)
2	4	○●○○○○●○	0x42	fatal	CPU Bd (82380)
2	5	○●○●○○●○	0x52	fatal	CPU Bd (82380)
2	6	○●●○○○●○	0x62	fatal	CPU Bd (82380)
2	7	○●●●○○●○	0x72	warning	CPU BD (SCSI Bus Ctrlr)
3	0	○○○○○○●●	0x03	non-error	CPU Bd (82380)
3	1	○○○●○○●●	0x13	fatal	CPU Bd (82380)
3	2	○○●○○○●●	0x23	warning	CPU Bd (Floppy Ctrlr)
3	3	○○●●○○●●	0x33	fatal	CPU Bd (P2 Bus)
3	4	○●○○○○●●	0x43	fatal	CPU Bd (P2 Bus)
3	5	○●○●○○●●	0x53	fatal	CPU Bd (MUSCI ASIC)
3	6	○●●○○○●●	0x63	----	unused
3	7	○●●●○○●●	0x73	----	unused
4	0	○○○○○●○○	0x04	fatal	Memory Bd (slot 1)
4	1	○○○●○●○○	0x14	fatal	Memory Bd "
4	2	○○●○○●○○	0x24	fatal	Memory Bd "
4	3	○○●●○●○○	0x34	fatal	Memory Bd "
4	4	○●○○○●○○	0x44	both	Memory Bd (cache)

Table 15-2 *POST LED Interpretation— Continued*

Sub-system#	*Test #*	*LED pattern*	*Hex value*	*Error type*	*FRU (component)*
4	5	○●○●○●○○	0x54	fatal	Memory Bd "
4	6	○●●○○●○○	0x64	fatal	Memory Bd "
4	7	○●●●○●○○	0x74	----	Reserved,OBP Testing
5	0	○○○○○●○●	0x05	fatal	Memory Bd (Slot 2)
5	1	○○○●○●○●	0x15	fatal	Memory Bd "
5	2	○○●○○●○●	0x25	fatal	Memory Bd "
5	3	○○●●○●○●	0x35	fatal	Memory Bd "
5	4	○●○○○●○●	0x45	both	Memory Bd (cache)
5	5	○●○●○●○●	0x55	both	Memory Bd "
5	6	○●●○○●○●	0x65	both	Memory Bd "
5	7	○●●●○●○●	0x75	----	Reserved,OBP Testing
6	0	○○○○○●●○	0x06	fatal	Memory Bd (Slot 3)
6	1	○○○●○●●○	0x16	fatal	Memory Bd "
6	2	○○●○○●●○	0x26	fatal	Memory Bd "
6	3	○○●●○●●○	0x36	fatal	Memory Bd "
6	4	○●○○○●●○	0x46	both	Memory Bd (cache)
6	5	○●○●○●●○	0x56	both	Memory Bd "
6	6	○●●○○●●○	0x66	both	Memory Bd (cache)
6	7	○●●●○●●○	0x76	----	Reserved,OBP Testing
7	0	○○○○○●●●	0x07	fatal	Memory Bd (Slot 4))
7	1	○○○●○●●●	0x17	fatal	Memory Bd "
7	2	○○●○○●●●	0x27	fatal	Memory Bd "
7	3	○○●●○●●●	0x37	fatal	Memory Bd "
7	4	○●○○○●●●	0x47	both	Memory Bd (cache)
7	5	○●○●○●●●	0x57	both	Memory Bd "
7	6	○●●○○●●●	0x67	both	Memory Bd "
7	7	○●●●○●●●	0x77	----	Reserved,OBP Testing
8	0	○○○○●○○○	0x08	warning	Video Bd (VRAM)
8	1	○○○●●○○○	0x18	warning	Video Bd "
8	2	○○●○●○○○	0x28	warning	Video Bd (Color LUT)
8	3	○○●●●○○○	0x38	warning	Video Bd (ALL)
8	4	○●○○●○○○	0x48	warning	Video Bd (Mouse/Kbd SCC)
8	5	○●○●●○○○	0x58	----	unused
8	6	○●●○●○○○	0x68	----	unused
8	7	○●●●●○○○	0x78	----	Reserved,OBP Testing
9	0	○○○○●○○●	0x09	warning	Mouse/Keyboard (Keybd)
9	1	○○○●●○○●	0x19	----	unused
9	2	○○●○●○○●	0x29	----	unused
9	3	○○●●●○○●	0x39	----	unused
9	4	○●○○●○○●	0x49	----	unused
9	5	○●○●●○○●	0x59	----	unused
9	6	○●●○●○○●	0x69	----	unused
9	7	○●●●●○○●	0x79	----	Reserved,OBP Testing
10	0	○○○○●○●○	0x0A	warning	Floppy Disk
10	1	○○○●●○●○	0x1A	----	unused

Table 15-2 *POST LED Interpretation— Continued*

Sub-system#	*Test #*	*LED pattern*	*Hex value*	*Error type*	*FRU (component)*
10	2	○○●○●○●○	0x2A	----	unused
10	3	○○●●●○●○	0x3A	----	unused
10	4	○●○○●○●○	0x4A	----	unused
10	5	○●○●●○●○	0x5A	----	unused
10	6	○●●○●○●○	0x6A	----	unused
10	7	○●●●●○●○	0x7A	----	Reserved,OBP Testing
11	0	○○○○●○●●	0x0B	warning	SCSI Hard Disk (boot device)
11	1	○○○●●○●●	0x1B	warning	SCSI Hard Disk (controller)
11	2	○○●○●○●●	0x2B	----	unused
11	3	○○●●●○●●	0x3B	----	unused
11	4	○●○○●○●●	0x4B	----	unused
11	5	○●○●●○●●	0x5B	----	unused
11	6	○●●○●○●●	0x6B	----	unused
11	7	○●●●●○●●	0x7B	----	Reserved,OBP Testing
12	0	○○○○●●○○	0x0C	warning	SCSI Tape Drive (Boot Device)
12	1	○○○●●●○○	0x1C	warning	SCSI Tape Drive (Controller)
12	2	○○●○●●○○	0x2C	----	unused
12	3	○○●●●●○○	0x3C	----	unused
12	4	○●○○●●○○	0x4C	----	unused
12	5	○●○●●●○○	0x5C	----	unused
12	6	○●●○●●○○	0x6C	----	unused
12	7	○●●●●●○○	0x7C	----	Reserved,OBP Testing
13	0	○○○○●●○●	0x0D	warning	Ethernet (Boot Device)
13	1	○○○●●●○●	0x1D	----	unused
13	2	○○●○●●○●	0x2D	----	unused
13	3	○○●●●●○●	0x3D	----	unused
13	4	○●○○●●○●	0x4D	----	unused
13	5	○●○●●●○●	0x5D	----	unused
13	6	○●●○●●○●	0x6D	----	unused
13	7	○●●●●●○●	0x7D	----	Reserved,OBP Testing

Test Descriptions

The following describes all POST diagnostics along with their associated test numbers, status messages, error messages, and optional information:

CPU Reset Test

Subsystem 0;Test 1

Upon a CPU reset, the 80386 performs an internal self-test. The result of this self-test is reported through the EAX register. Any non-zero value is considered an error and is reported as such.

LED Register Test

Subsystem 0;Test 2

This test sequentially lights up each of the LEDs, beginning with the least significant bit and ending with the most significant bit. It indirectly tests the CPU' s ability to fetch instructions from the boot PROM and transfer data across the data bus.

This test does have a test number associated with it, but its pattern is not visible due to the nature of the test.

Verify Wait State Registers Reset

Subsystem 1;Test 0

This test verifies the reset condition of the wait state registers. Power on default = 0xFF.

Verify Refresh Register Reset

Subsystem 1;Test 1

This test verifies the reset condition of the refresh register. Power on default = 0xFC.

Verify Relocation Register Reset

Subsystem 1;Test 2

This test verifies the reset condition of the relocation register. Power on default = 0x0.

Verify 82380 Relocation

Subsystem 1;Test 3

This test performs the remapping of the 82380 to memory space. It then verifies that the relocation occurred correctly by reading the diagnostic register at the new address.

Verify DMA Controller Reset

Subsystem 1;Test 4

This test verifies the reset condition of certain DMA Controller registers.

Boot PROM Checksum Test

Subsystem 0;Test 3

This test ''exclusive-ors'' all locations in the Boot PROM and verifies that the result is equal to 0. If a non-zero value is obtained, then the test will have failed.

Serial Port A Internal Loopback Test

Subsystem 0;Test 4

This test puts the SCC chip into internal loop back mode and transmits 10 characters at the most popular setting of 9600 baud, 8 data bits, 1 stop bit, with even parity and verifies data is received without errors. It then re-initializes the chip so it can receive data or transmit messages.

NVRAM Diagnostic Area Test

Subsystem 0;Test 6

This test verifies the validity of the NVRAM area used by diagnostics.

MUSCI Diagnostic Register Test

Subsystem 3;Test 5

This test verifies that the diagnostic register used by the POST code is valid.

Refresh (Timer 1) Downcount Test

Subsystem 2;Test 0

This test verifies the functionality of Timer 1 of the 82380. This timer is used as the refresh counter and thus can only be tested before refresh is started. The test sets a simple countdown mode and checks both that the timer is counting-down and that it sets the TOUT status line upon completion.

Start Refresh Timer, Wait State, Refresh Regs.

Subsystem 1;Test 5

This test initializes and triggers the refresh timer (#1) and the refresh control register along with setting up the wait state registers of the 82380. All actions are verified by reading the appropriate 82380 register.

Timer 0 Downcount Test

Subsystem 2;Test 1

This test verifies the functionality of Timer 0 of the 82380. This timer is used as a general purpose counter during the POST sequence. The test sets a simple countdown mode and checks both that the timer is counting-down and that it sets the TOUT status line upon completion.

Speaker Timer (2) Downcount Test

Subsystem 2;Test 2

This test verifies the functionality of Timer 2 of the 82380. This timer is used as the frequency generator for the Sun386i speaker. The test sets a simple countdown mode and checks both that the timer is counting-down and that it sets the TOUT status line upon completion. Upon completion, the Sun386i speaker will "beep" to verify that the speaker circuit is functional.

Clock Tick Timer (3) Downcount Test

Subsystem 2;Test 3

This test verifies the functionality of Timer 3 of the 82380. This timer is used as the "clock tick" generator for the Sun386i I/O polling functions. The test sets a simple countdown mode and checks both that the timer is counting-down and that it sets the TOUT status line upon completion.

Cache MBAR Test

Subsystem 4 - 7;Test 4

Any KALI (cache memory controller) caches found will be tested for MBAR integrity (ability to be mapped correctly). This test must be done in PROM. The KALI controlling the first RAM area (4 Mbytes) is considered vital (fatal error), but any problems found with subsequent KALIs is considered non-fatal (warning error) so that the down-loaded diagnostics can be used to further isolate the problem. Each valid KALI is mapped to a section of DRAM so that DRAM refresh can be started. The DRAM array requires at least 8 refresh cycles (roughly 125 microseconds) before being accessed, and the KALI must be mapped for that to happen. The cache will remain disabled, however, so that DRAM can be accessed directly at the start.

RAM Bus Access Test

Subsystem 3;Test 3

This test determines only if memory can be successfully accessed. The exact memory location is irrelevant. This test will only verify that at least one location out of several attempted could be written and read back correctly. If there are any particular failures, the other tests must isolate the specific error. One successful location constitutes a passing condition. This test serves primarily to validate the P2 Bus and its connectors.

RAM Bus Data Path Test

Subsystem 3;Test 4

This test verifies the path for the data lines and drivers. A marching one's and zero's pattern is used (0,1,2,4,8,10...). Several locations are selected for the test and if any one of them pass the entire patterns test, the test will have passed.

RAM Address Test

Subsystem 4-7;Test 0

This test exercises the address buffer and address up to 1 Mbyte. Any address or DRAM fault that is isolated will cause this test to fail. The test assumes that memory is accessible and that the data path is valid.

RAM Data Test - Phase I

Subsystem 4-7;Test 3

This test checks the first 256 Kbyte section of lower memory to ensure that the RAM area required by the PROM to support video, keyboard, and stack operations will be verified prior to use. If any location comes back with the wrong data, the test will have failed. Note that the rest of critical RAM (up to 1 megabyte) will be tested after video initialization and testing so that the progress meter can be displayed as soon as possible. Note also that at this point, the PROM is still using NVRAM as a stack area so there is no dependence on this RAM for the testing.

At this point, the page tables are initialized and the 80386 is put into virtual memory mode. This action allows the video drivers to be used to display text on the screen. The PROM sets up additional mapping so that virtual/physical one-to-one mapping is supported for the monitor and standalone diagnostics. This mapping is deleted once the operating system boots.

Bus Sizing Routine

No test number is assigned to this routine, and no errors reported. However, it is necessary to check the P2 bus for valid boards so that the remaining power-on tests can act on that information. For example, if there is a video board present, the video tests must know what type (color/mono) to determine how to test it. The amount and location of RAM cards must be determined and future boards (with on-board diagnostics) must be located and activated.

In normal mode, this routine is transparent to the user. In diagnostic mode, however, the routine prints out the result of its search in this format:

```
Sizing System bus...

    Slot ID 4(81)=> Graphics board.
    Slot ID 3 (63)=> SIMM board.
    Slot ID 2 (FF)=> no valid board ID.
    Slot ID 2 (FF)=> no vlaid board ID.
```

NOTE *Tests 0 - 3, Subsystem 8 are used only if a valid Sun386i video board is detected. Option Video Boards trigger a separate set of Option Board PROM (OBP) tests.*

Video RAM Address Test

Subsystem 8;Test 0

This test runs a simple address line test to quickly verify the address lines on the video board.

Video RAM Data Test

Subsystem 8;Test 1

This test uses the configuration information from the sizing routine to determine what video board (color or mono) is installed and tests the video memory that it finds. A simple data pattern test is performed to verify minimum functionality.

Color LUT RAM Test (Only for Color Systems)

Subsystem 8;Test 2

This test verifies the RAM inside the Brooktree Color Look-Up Table (LUT) for color frame buffers. A simple four pattern test will be used.

Keyboard/Mouse Internal Loopback Test

Subsystem 8; Test 4

This test puts the SCC chip into internal loop back mode and transmits a walking-one pattern at the most popular setting of 1200 baud, 8 data bits, 1 stop bit, with even parity. It then verifies that the data is received without any errors.

Keyboard Reset Test

Subsystem 9;Test 0

This test issues a reset to the keyboard and verifies that a passed status is returned to the CPU board. The keyboard will "beep" after the test is done.

Video Initialize Sequence

Subsystem 8;Test 3

This routine does not test the video board *per se* but enough is happening to justify a status update. During this routine, the PROM expands the character font into RAM, initializes the video RAM to all pixels on (1 for mono, 0xFF for color), sets some LUT offsets for its own use and enables the video itself so that the user can see what is happening. After all of that, the logo and banner will be drawn as well as the "progress meter", which has been updated to show prior progress. From this point on, the meter will be updated as each test runs.

Verify Interrupt Vector Registers

Subsystem 1;Test 6

This test verifies that vectors for all Sun386i device interrupts have been written correctly to the 82380 vector registers.

Verify Interrupt Request Lines

Subsystem 2;Test 4

This test uses Timer 0, which generates the IRQ8 interrupt, to verify that, with interrupts disabled, the interrupt request line still registers a request for IRQ8 when the terminal count is reached.

Verify Interrupt Occurs & IRQ8 Status

Subsystem 2;Test 5

This test uses Timer 0, which generates the IRQ8 interrupt, to verify that a valid interrupt can be serviced by the 82380.

Verify Interrupt Masking

Subsystem 2;Test 6

This test uses Timer 0, which generates the IRQ8 interrupt, to verify that a valid interrupt can be successfully masked by the 82380.

RAM Data Test - Phase II

Subsystem 4-7;Test 1

This test checks the RAM area between 256K and 1 Mbyte to ensure that any programs that are to be booted after the self-tests will have verified memory in which to execute. Note that the rest of RAM will be tested, if the NVRAM is so configured, during the Phase II confidence test. This test and all that follow will be responsible for updating the progress meter on systems with video capability.

DRAM Refresh Test

Subsystem 4-7;Test 2

This test ensures that the refresh request bit toggles and then writes a pattern into memory, delaying 30-plus seconds and verifying that the data is still valid. This test will only be done in the Manufacturing version due to the length of the run time.

RAM Parity Test

Subsystem 4-7;Test 3

This test checks the parity NMI circuitry by setting the opposite parity bit, writing the wrong parity to several locations, setting the parity bit back to the normal position, and reading the locations to ensure that a parity error occurs. The test will have failed if no parity error occurs.

ID PROM Checksum Test

Subsystem 0;Test 7

This test adds all 16 locations in the ID PROM and verifies that the total byte sum is equal to zero. If a non-zero value is obtained, the test will have failed.

DMA Verify Transfer Mode Test

Subsystem 3;Test 0

This test checks that channels can run in verify transfer mode. In this mode, a DMA cycle is generated but no data is actually written. Channel 4 is checked to ensure that it updates the status registers correctly.

DMA Block Transfer Test

Subsystem 3;Test 1

This test performs a block transfer of PROM code to a point in RAM and verifies that the transfer occurred, that the data is where it should be, and that it is valid.

Boot Path Device Checkout

Subsystem A-D;Test 0

This test polls the currently enabled boot path and performs status tests where possible to determine the health of the device. If no device is specified in the NVRAM, the PROM will test each boot-path until a device with a valid boot block is found. A failure on the active boot path implies that the second phase of the POST can not be loaded. Error reporting is limited to driver error messages. Since the most likely course of action will be to actively check out the failing path, the PROM will drop into the Monitor so that the Extended Menu Tests can be used to isolate the problem.

Phase 2 Confidence Test Begins At This Point

After the boot device has been checked, the PROM down-loads the second phase of the POST diagnostics, which is linked together with the boot block code in the file `/boot`. At this point all hardware necessary to run the code out of RAM has been checked. The Phase 2 confidence tests will be used to check all the rest of the Sun386i system that is required for full scale operations. Channel 0 will be checked to ensure that it updates the status and current registers correctly.

Parallel Port Test

Subsystem 0;Test 5

This test verifies that the parallel port register is functional.

Cache CTAGS Test

Subsystem 4-7;Test 5

This test performs a write/read check of all 128 cache tag registers for each KALI (cache controller chip). The VALID, MODIFIED, and Physical Address fields will be verified.

The KALI controlling the first RAM area (4M bytes) is considered vital (fatal error), but any problems found with subsequent KALIs are considered non-fatal (warning error) so that the down-loaded diagnostics can be used to further isolate the problem. If a problem is found on any given KALI, the POST will attempt to configure it and its associated DRAM array out of the system (i.e. a bad KALI will be skipped over so the system memory is contiguous).

Cache Static RAM Test

Subsystem 4-7;Test 6

This test checks the KALI internal static RAM array by setting up a situation in which all CTAGS are valid so that no DRAM interaction is initiated.

The KALI controlling the first RAM area (4M bytes) is considered vital (fatal error), but any problems found with subsequent KALIs are considered non-fatal (warning error) so that the down-loaded diagnostics can be used to further isolate the problem. If a problem is found on any given KALI, the POST will attempt to configure it and its associated DRAM array out of the system (i.e. a bad KALI will be skipped over so the system memory is contiguous).

Optional RAM Tests (After 1 Mbyte)

Subsystem 4-7;Tests 1

Based on the NVRAM configuration, none, some, or all of the system memory may be tested for data and address type errors. The goal will be to test up to 32 megabytes in a fairly thorough manner without requiring more than two minutes for a fully loaded system.

Floppy Controller Check

Subsystem 3;Test 2

This test resets and verifies the floppy controller status and issues a recalibrate command to the drive to verify that the mechanics are working. This test is not executed when the floppy disk is the boot device.

Ethernet Internal Loopback Test

Subsystem 1;Test 7

This test resets the 82586 Ethernet Controller, puts it in internal loopback mode and verifies that the data path out to the drivers and receivers is healthy. This test is not executed when Ethernet is the boot device.

Hard Disk Controller Check

Subsystem 11;Test 1

This test checks the hard disk controller and issues a few commands to the drive to verify that the mechanics are working. This test is not executed when a hard disk is the boot device.

Tape Drive Controller Check

Subsystem 12;Test 1

This test checks the tape drive controller and issues a few commands to the drive to verify that the mechanics are working. This test is not executed when the SCSI tape is the boot device.

POST CODE ENDS. BOOT CODE STARTS.

Initialization of System Parameters

It is the responsibility of the Boot PROM to initialize the system into a known state on power up and on a soft reset. The only difference between a hard and a soft reset is memory initialization.

After a hard reset, all memory locations (memory is sized prior to this point) are initialized to 0x0 in order to set the proper parity. On a soft reset, memory is not initialized, but is left in its existing state. This is done for time considerations as well as for software reasons. The remaining initialization is identical for hard and soft reset conditions.

The specific initialization values for the major components are described in the table that follows:

Table 15-3 *Initialization Values*

Component	*What Is Initialized*	*Setting*
80386:	Protected/Virtual Paging	enabled
80387:		taken out of tri-state mode.
82380 Subsystems:		
Timer 0:	Assignable counter	disabled
Timer 1:	DRAM Refresh timer	enabled
Timer 2:	speaker frequency	disabled
Timer 3:	Clock Tick Generator	enabled
DMA Controller:	Master Clear Register	triggered
Interrupt Controller:	All interrupt, exception, and trap handlers	initialized
	All device interrupts except NMI & Timer 3	masked
Refresh Register:	AT Bus Refresh Timing	enabled for 8 bit mode
Wait State Register:		zero wait states
Relocation Register:	82380 Remapped	to memory map addr = 0xB0000000
Internal Control Port:	Tmr 2, shutdown detect	disabled
Diagnostic Ports:	Watchdog Signature	set to 0xAA
Serial Port:	POST Errors & console	9600 baud.
RAM Memory		set to 0x0 only during hard reset
Video Memory:		cleared only during hard reset
Color LUTs	color, foreground/background	set by NVRAM byte during hard reset only. Def= White Background
Keyboard/Mouse:	Video Board SCC Ports A, B	baud = 1200.
BABE (Ethernet Controller) Chip:	Ethernet Control Register	set to 0x100
MUSCI (SCSI Controller) Chip:	System Control Register	set to 0xDE
KALI (Cache Controller) Chips:	KALI RESET & remapped	CCR = 0x80, MBAR set

Sun386i Future Option Board Feature

To support future Sun options, the CPU Boot PROM will go out to the assigned slots and check for new installed options. The diagnostic will check the base address for any existing PROMs on that board by looking for a device ID. If found, the size of the PROMs will be read, a checksum done and compared with the stored value in the option PROM. If it is a valid PROM, with a valid size (i.e. firmware included), the Boot PROM will transfer the code from the PROM into RAM and proceed to jump to it.

When the option board has completed testing, it will pass control back to the CPU PROM diagnostics, having reported and acted on any errors found.

Test 7 in each subsystem has been reserved for option board tests. Test 0x78 = video board option, for example (refer to the POST LED Interpretation tables). This option test number signals the CPU Boot PROM to use the Option Board tests and error messages instead of the CPU board POST error messages.

16

Sun386i Monitor Commands

Sun386i Monitor Commands ... 319

16.1. Conventions ... 319

16.2. List of Sun386i Monitor Commands ... 323

16

Sun386i Monitor Commands

All PROM actions are channeled through the Monitor to some extent. However, while much of the Monitor code is automatic, the Monitor is first and foremost an interactive tool. The low level Monitor commands provide a number of debugging/trouble-shooting functions. Several utilities will also be available with Sun386i firmware. You may initiate these activities with the Monitor commands:

- Boot the SunOS operating system, the Diagnostic Executive, or a stand-alone program over any boot path.
- Test peripheral paths: Ethernet, disk, tape, serial.
- Examine and modify CPU registers, memory, and devices on both the I/O and Memory maps.
- Set and clear breakpoints, and single step instructions.
- Display contents of the NVRAM in a formatted manner.
- Run optional diagnostic tests of such things as video, SCC, and Ethernet. The following text discusses Monitor command line syntax and a describes each of the commands.

16.1. Conventions

In this text, if a command has more than one distinct format, each one is shown on a separate line. Characters in **`bold typewriter`** font mean that you should enter them exactly as shown. Plain `typewriter` font represents what you should see on the screen. Words in *`typewriter italic`* show the type of optional information you are to enter. *Roman italic* or **boldfaced** font is also used for notes or emphasis within the text. Optional arguments and default values are listed in the descriptions.

Special Monitor Commands

The following commands may be used to augment other commands, and aid in programming or debugging.

Address Increment/Decrement Command

By preceding the command with a **+** or **–**, you can cause the address display to increment or decrement to the next location.

While traversing a range of addresses, type a + or – to change direction after the program displays the content and waits for input.

For example:

```
>l 0 Return

00000000 00000000 ?  Return

00000004 00000001 ?  Return

00000008 00000002 ? - Return

00000004 00000001 ?  Return (contents of previous location are displayed)

00000000 00000000 ? + Return (after decrement, you enter a plus sign)

00000004 00000001 ?  Return (you increment again)

00000008 00000002 ?  Return (you increment again)
```

The `^T` Command

Entering the `^` character and then the **`t`** key, followed by a virtual address, displays the physical address to which that address is mapped, along with a detailed description of all the bits in the page table entry and page directory.

For example, entering

```
>^t 1000 Return
```

results in this sort of display:

```
Virtual Addr 1000 is mapped to Physical Addr 1000
Page Dir [0x0] = 0x15025, Page Table [0x1]=0x1003.
Page 1 has these attributes:

       Valid bit  = 1
             R/W = 1
             U/S = 0
      Access bit = 0
       Dirty bit = 0
```

Entering **`^t`** with no argument provides information with reference to virtual address 0x00.

The `^I` Command

Entering the `^` character and then the **`i`** key, followed with a command, displays compilation information for the system firmware. It includes the date, host name and build directory path. For example:

Compiled at 6/7/87 on *`hostname`* in *`/directory_name`*

The ^C Command

`^C source destination n`

Entering the ^ character and then the **c** key, followed with the parameters shown, causes a block of *n* length to be copied from `source` to `destination` address, byte by byte.

Displaying and Modifying Memory

A number of the commands listed here can be used to display and/or modify the system's memory and registers. Regardless of the type of memory (RAM, NVRAM, etc.) or register, these commands have the same command syntax. This section describes the memory modification syntax used by the monitor commands. The example here references long words (32 bits) of memory, but the syntax is the same for bytes (8 bits) and words (16 bits).

The `address` argument specifies the initial memory location that is to be displayed and/or modified.

The second argument determines whether the current content of a memory location is to be displayed and/or modified. Entering *only* the address of a memory location after the command *displays* the content of that address.

```
>command FFE80000 Return

FFE80000: 12345678 ?
```

At this point, you can respond in one of *three* different ways.

1. Simply pressing the Return key displays the contents of the next memory location (in this case, 0xFFE80004) as shown below.

```
    >command FFE80000 Return

    FFE80000: 12345678 ?  Return

    FFE80004: 00000001 ?
```

2. Successively pressing the Return key displays the contents of successive memory locations. Assuming that you pressed Return four times, the contents of memory locations 0xFFE80004, 0xFFE80008, FFE8000C and 0xFFE80010 would be displayed.

To exit the command, enter any *non-hexadecimal* character (i.e. q for quit) before pressing Return Note that you will now return to the monitor's basic command level, with the > prompt.

```
>command FFE80000 Return

FFE80000: 12345678 ? q Return

>
```

3. The *third* response to the display of memory location contents is to *modify* those contents. You enter the *new* hexadecimal value immediately following the question mark ?, BEFORE pressing Return. The following display demonstrates how the value of memory location 0xFFE80000 can be changed from "12345678" to "00ABCDEF".

NOTE *Following the assignment of the new value to memory location 0xFFE80000, the new value will not be displayed; instead, the contents of the next memory location are shown. You will not be returned to the monitor's basic command level.*

```
>command FFE80000 Return

FFE80000: 12345678 ? 00ABCDEF Return

FFE80004: 00000001 ?
```

Entering a memory location's virtual address followed *only* by a non-hexadecimal character before pressing the Return key causes the monitor to *display* the contents of that location then exit to the monitor command level.

```
>command FFE80004 q Return

FFE80004: 00000001
>
```

Entering a non-hexadecimal character *and* a hexadecimal value after an address causes the monitor to display the original value at that virtual address, assign a new value, then display that value. For instance, assume that the original content at address 0xFFE80010 is "00000005". The command

```
>command FFE80010 ? 55555550 Return
```

displays the original value "00000005" then assigns the value "55555550" to address FFE80010 before displaying the contents of the next address, as shown below:

```
>command FFE80010 ? 55555550 Return
FFE80010: 00000005 -> 55555550
FFE80014: 00000006 ?
```

Entering a hexadecimal value immediately after the virtual address corresponding to a memory location assigns the new value to that memory location, displays the newly-assigned value, then returns to the monitor. If you enter:

```
>command FFE80014 66666660 Return
```

The program will answer back:

```
FFE80014 -> 66666660
>
```

You may also enter multiple display and/or modify commands for multiple memory locations on the same command line. If you enter this command line:

```
>command FFE80000 ? 00000000 ? ? 22222220 33333330 q Return
```

You will see this on the screen:

```
FFE80000: 12345678 -> 00000000
FFE80004: 00000001
FFE80008: 00000002 -> 22222220
FFE8000C -> 33333330
FFE80010: 00000004
>
```

The first part of the command line,

>*command* **`FFE80000 ? 00000000`**

displays the original contents of location FFE80000 before assigning the new value "00000000" to it.

The next question mark directs the monitor to display the contents of FFE80004. The next part of the command line, **`? 22222220`**, tells the monitor to display the original contents of FFE80008, before assigning the new value "22222220" to it. The **`33333330`** tells the monitor to assign the value "33333330" to memory location FFE8000C. Finally, the **`q`** causes the monitor to exit to the command level after the contents of FFE80010 are displayed.

16.2. List of Sun386i Monitor Commands

Following are descriptions of each PROM monitor command and examples of argument options.

Monitor `b` Command

`b` ? or **`b`** *`! boot_device path argument_list`*

The `boot` command loads and executes the SunOS operating system, an NVRAM-specified program, or a user-specified program. The boot program can be loaded from the default device, the device specified in the NVRAM, or the boot device specified in the command argument. A *`boot_device`* is a secondary storage device (disk, Ethernet or tape) that contains the program to be loaded and executed.

If the system is set for a normal boot, the value in NVRAM address 0x18 is *not* equal to 0x12, and the boot command is entered without arguments, the system will boot the SunOS operating system, using the following default boot device polling sequence.

1. Floppy Disk
2. SCSI Tape (Unit4) (Peripheral Box)
3. SCSI Disk (Unit 6) (Half-Height Disk/CD ROM — CPU Box)
4. SCSI Disk (Unit 5) (Half-Height Disk/CD ROM — Peripheral Box)
5. SCSI Disk (Unit 3) (Secondary Disk — CPU Box)
6. SCSI Disk (Unit 2) (Main Disk — CPU Box)
7. SCSI Disk (Unit 1) (Secondary Disk — Peripheral Box)
8. SCSI Disk (Unit 0) (Main Disk — Peripheral Box)
9. Ethernet

If the NVRAM value at address 0x18 *is* equal to 0x12, and the **b** command is entered by itself, the system will boot the SunOS operating system from an NVRAM-specified device. The boot device is specified in locations 0x19 through 0x1D, inclusive, of the NVRAM.

If Diagnostic Mode is enabled and command **b** is entered by itself, the system will boot an NVRAM-specified program from an NVRAM-specified device. In this case, the boot path is specified in locations 0x28 through 0x50, inclusive, of the NVRAM; the boot device is specified in locations 0x22 through 0x26, inclusive, of the NVRAM. If the boot attempt fails, the user is returned to the monitor's command level.

In order to boot from a specific device, the **b** command must be followed with a boot device abbreviation, such as those shown below. Enter **b ?** to view the boot device identifier arguments that your PROM monitor will accept.

```
b! device(controller,unit,partition)path argument_list
```

device may be one of the following:

fd — floppy disk
sd — SCSI disk
ie — Intel Ethernet
st — SCSI tape

You must surround `controller,` `unit,` and `partition` with parentheses, and place a comma after each entry. To invoke the default values, simply enter:

```
b device (,,)
```

controller stands for the Controller Number, referring to the tape or disk controller board. The default is 0.

unit refers to Unit Number, meaning disk number. The default is 0.

partition is the partition number of the boot device. The default is 0.

path is the path and filename of the program to boot.

argument_list is the list of arguments for the boot program. Up to seven optional arguments (which are passed to the boot program) may follow the path argument.

If you do not want the system to be reset prior to booting, you must insert `!` before the device identifier argument.

The boot command given below would boot the video diagnostic from the `/stand` directory over the Ethernet. Because the `!` argument does *not* precede the device identifier argument, the system is reset before the booting process begins. Note that one optional argument (`-t`) is passed on to program `video.diag`.

```
b ie(0,0,1)stand/video.diag -t
```

Monitor `c` Command

`c` *virtual_address*

The `continue` command resumes execution of an interrupted program. You can specify the virtual address of the instruction to be executed when the program restarts.

By default, the program resumes execution at the address pointed to by the program counter (EIP).

NOTE *This command is helpful if you should use the* `L1-A` *sequence and decide you did not want to abort the operating system.*

Monitor `d` Command

`d`

The `d` (dump) command displays the state of the processor. The processor state display will consist of the registers listed below:

- Processor Registers (EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, EFLAGS, EIP).
- Segment Registers (ES, CS, SS, DS, FS, GS).
- Memory Management Registers (GDTR, LDTR, IDTR, TR).
- Control Registers (CR0, CR2, CR3).
- Debug Registers (DR0, DR1 , DR2 , DR3, DR6, DR7)
- Test Registers (TR6, TR7).

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`)
- You have manually "broken" into the monitor (by typing `L1-a` on the console's keyboard or **`break`** on the "dumb" terminal's keyboard).

Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word `command` in the description with the letter `d`.

Monitor **e** Command

`e virtual_address`

The `display/modify memory` command displays and/or modifies the content of one or more virtual addresses in *word* mode (i.e. each virtual address will be treated as a 16-bit unit). That is, the content of one or more words can be *displayed*, *modified* or, both *displayed* and *modified*.

See the previous sections *Displaying Modifying Memory* and *Special Monitor Commands* for a description of this command's syntax. Use the letter **e** in place of the word `command` in the description. Use the plus and minus commands to specify whether you want to increment or decrement to the next location.

Monitor **f** Command

`f start_virtual_address end_virtual_address pattern size`

The `block write` command writes the `pattern` you enter into each byte, word or long word in the range of virtual addresses you specify with the `start_virtual_address` and `end_virtual_address` arguments. By default, if the final, memory-cell-size argument is not present, bytes are written. Arguments `start_virtual_address end_virtual_address` and `pattern` are required while `size` is optional. The possible values for `size` are `b` (8-bit byte), `w` (16-bit word) or `l` (32-bit long word).

Monitor **g** Command

g *vector_number argument*
or

g *virtual_address argument*

When you use the `goto` command, you may go to (jump to) a *pre-determined*, *user-specified* or *default* routine. If a pre-determined or default routine is invoked, optional arguments `vector` (a hexadecimal number) and `argument` (a string) are passed to the routine to be executed.

In its other form, the argument `virtual_address` is used to indicate the virtual address of a *user-specified* routine, and the optional `argument` is passed along to that routine. If you do not supply the `vector/virtual_address` argument, the value in the Program Counter is used.

In order to set up a *pre-determined* routine, a user program has to set variable `*romp->v_vector_cmd` equal to the virtual address of the routine. Variable `*romp->v_vector_cmd` must be set prior to executing the `g` command. Pre-determined routines may or may not return to the monitor.

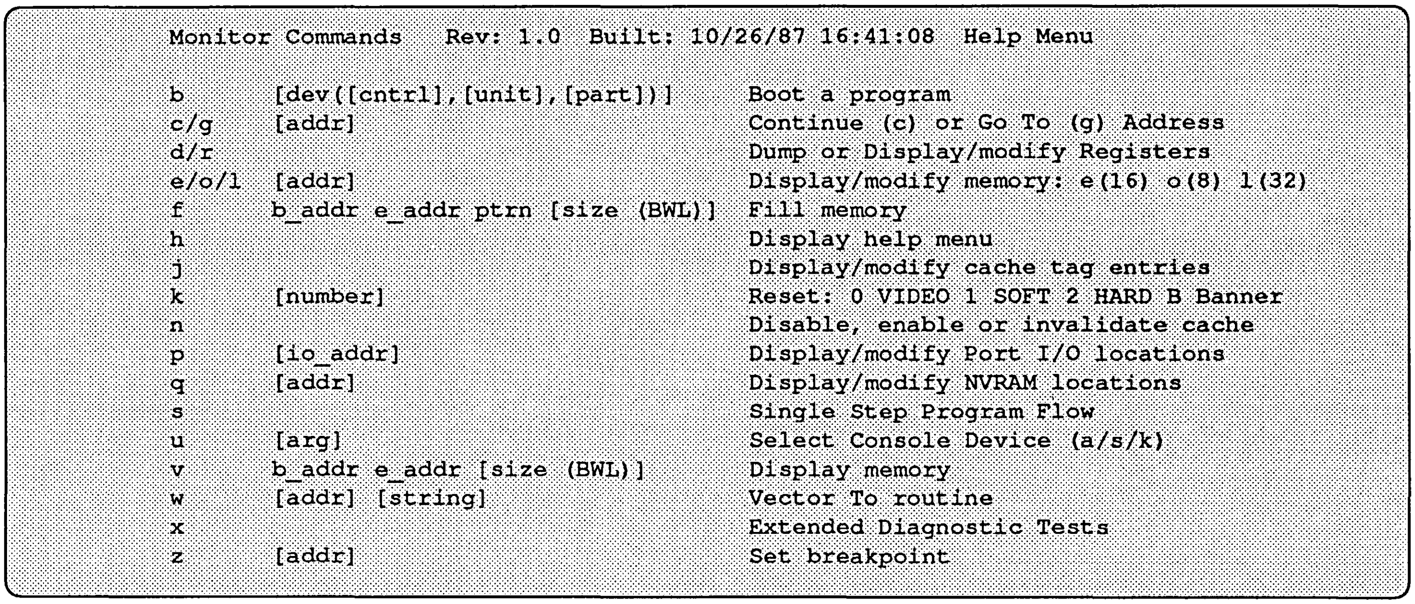
The *default* routine, defined by the monitor, simply prints the user-specified `vector` argument according to the user-specified format (given in `argument`) before returning to the monitor. The only allowable formats are `%x` and `%d`. Format `%x` prints argument `vector` as a hexadecimal number while format `%d` prints argument `vector` as a decimal number.

Monitor **h** Command

h

The `help` command brings up a Help display that describes the basic monitor commands and their argument(s). The **h** command does not accept arguments. The Help menu looks something like this:

Figure 16-1 *Sun386i Monitor Help Menu*



```
Monitor Commands   Rev: 1.0  Built: 10/26/87 16:41:08  Help Menu

b       [dev([cntrl],[unit],[part])]      Boot a program
c/g     [addr]                            Continue (c) or Go To (g) Address
d/r                                       Dump or Display/modify Registers
e/o/l   [addr]                            Display/modify memory: e(16) o(8) l(32)
f       b_addr e_addr ptrn [size (BWL)]   Fill memory
h                                         Display help menu
j                                         Display/modify cache tag entries
k       [number]                          Reset: 0 VIDEO 1 SOFT 2 HARD B Banner
n                                         Disable, enable or invalidate cache
p       [io_addr]                         Display/modify Port I/O locations
q       [addr]                            Display/modify NVRAM locations
s                                         Single Step Program Flow
u       [arg]                             Select Console Device (a/s/k)
v       b_addr e_addr [size (BWL)]        Display memory
w       [addr] [string]                   Vector To routine
x                                         Extended Diagnostic Tests
z       [addr]                            Set breakpoint
```

Monitor `j` Command

`j` *`cache_tag_offset`*

The `modify cache tag RAM` command displays and/or modifies the contents of one or more of the cache tag addresses. Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter **`j`**.

Monitor **k** Command

k *reset_level*

The `reset` command performs various levels of system resets. It can also be used to display the system banner. This command accepts one optional argument.

Entering **k 0** (the Low-Level Reset) resets the video monitor only.

Entering **k 1** invokes a Software Reset.

Entering **k 2** invokes a Power-On Reset (Hard Reset).

Finally, entering **k b** displays the banner on the video monitor. The default value of the argument is `0`.

Monitor `l` Command

`l` *virtual_address*

The `modify long words of memory` command displays and/or modifies the contents of one or more virtual addresses in *long* word mode. Each virtual address is treated as a 32-bit unit (long word). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter `l`.

Monitor `n` Command

`n` *`cache_command`*

The `control cache` command globally controls the cache. One argument is required. Entering `n d` *disables* the cache. Entering `n e` *enables* the cache. Finally, entering `n i` *invalidates* the cache.

Monitor `o` Command

`o` *virtual_address*

The `modify bytes of memory` command displays and/or modifies the content of one or more virtual addresses in *byte* mode. Each virtual address is treated as an 8-bit unit (byte). Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter `o`.

Monitor `p` Command

`p` *`port_address`*

The `modify I/O Port Address` command displays and/or modifies the contents of one or more of the I/O Port addresses in byte mode. Each port address is treated as an 8-bit unit. Optional argument *`port_address`* is a 16-bit quantity that specifies the initial I/O Port address to be displayed or modified.

Read *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter `p`.

Monitor `q` Command

`q` *`nvram_offset`* or `q *`

The `modify bytes of NVRAM` command displays and/or modifies the configuration information within the NVRAM in *byte* mode. This command works similarly to the memory commands discussed previously, except that the modified addresses are offset, and the changes you make remain when you power-down the workstation remain in effect until you modify the NVRAM again.

Read the previous section, *Displaying and Modifying Memory* for details on how to use this command. Replace the word *`command`* in the description with the letter `q`. Refer to the *Sun386i NVRAM Layout* chapter for information on what parameters are stored at which NVRAM addresses.

You may enter an asterisk instead of an NVRAM offset value as an argument, and the `q` command will erase the entire NVRAM.

Monitor **r** Command

r *`reg_name`*

The **r** command displays and/or modifies the content of of one or more of the Processor Registers (EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, FLG, EIP, ES CS, SS, DS, FS and GS).

If optional argument *`reg_name`* (from the list of processor registers shown above) is provided, the specified register will be displayed first. The default value of argument *`reg_name`* is `EAX`.

It is important to note that it is *not* possible to display the state of the processor at all times. In particular, processor state is only observable after:

- An unexpected trap
- A user program has "dropped" into the monitor (by calling monitor function `abortent`)
- You have manually "broken" into the monitor (by typing `L1-a` on the console's keyboard or [break] on the "dumb" terminal's keyboard).

Monitor **s** Command

s *`step_count`*

The **s** command single steps through the execution of the interrupted program. Optional argument *`step_count`* specifies the number of single steps to execute before displaying the monitor prompt. The default value for *`step_count`* is 1.

Monitor **u** Command

u *`port options baud_rate`*
or

u *`echo`*
or

u u *`virtual_address`*
The *input/output* command configures the input and output devices and their characteristics or displays the current input and output device set-up.

The **u** command requires arguments to specify from which device(s) you want the system to expect input or which device(s) will display output.

If you do not enter an argument after the u command, the program will display the current settings. If no serial port is specified when changing baud rates, the baud rate of the current input device is changed. The default serial port baud rate is 9600.

Upon normal power-up, the default console input device is the Sun keyboard, unless the NVRAM has specified another default input device. If the keyboard is unavailable, the system looks to serial port A for for input.

The default console output device is the Sun monitor (subject to change through NVRAM programming). If the workstation has a color monitor and a color board is unavailable, the program will look for a monochrome monitor as an output device.

You may alter the existing I/O settings while you are in the monitor mode, using the commands listed below; however, the default settings will be reinstated when the system is power cycled.

u Command Arguments:

The *`port`* argument can be one of the following:

Table 16-1 *Port Arguments*

Port Argument	Device
`a`	Serial Port A
`k`	Keyboard
`s`	Screen

The *`options`* arguments are:

Table 16-2 *Option Arguments*

Option Argument	Meaning
`i`	input
`o`	output
`r`	reset specified port
`u`	UART
`e`	input echoed to output
`ne`	input *not* echoed to output

The *`baud_rate`* argument specifies baud rate of serial port A.

The *`virtual_address`* argument specifies the virtual address of the UART (Universal Asynchronous Receiver/Transmitter).

Following are examples of port and options arguments:

- Enter **`u a`** to set up serial port A as both the input and output device.
- Enter **`u aio`** to set up serial port A as both the input and output device.
- Enter **`u ai`** to select serial port A for input only.
- Enter **`u ao`** or to select serial port A for output only.
- Enter **`u ar`** to reset serial port A to its default setting.
- Enter **`u k`** to select the keyboard for input.
- Enter **`u ki`** to select the keyboard for input.
- Enter **`u s`** to select the screen for output.
- Enter **`u so`** to select the screen for output.
- Enter **`u ks,sk`** to select the keyboard for input and the screen for output.
- Enter **`u a`***`baud rate`* to set the serial port speed.
- Enter **`u e`** to cause the output to echo the input.
- Enter **`u ne`** to cause the output **not** to echo the input.
- Enter **`u`** *address* to set the serial port virtual address.

A *previously mapped* UART can also be used for input and/or output. Command **`u u`** *`virtual_address`*, where *`virtual_address`* is the virtual address of a previously mapped UART, changes the virtual address of the UART.

If the **u** command is not followed by an argument, the current settings are displayed. The current input device, output device, baud rate for serial ports A and B, UARTs virtual address and input to output echo indicator are shown.

Monitor **v** Command

v *`start_virtual_address end_virtual_address size`*

The `display memory block` command displays the contents of each byte, word or long word in the range of virtual addresses specified by *`start_virtual_address`* and *`end_virtual_address`*. You must enter a low and high virtual address; the *`size`* argument is optional. The default is to display memory in byte format. In this format, sixteen consecutive bytes are displayed on each line. In word format, eight consecutive words appear on each line.

If long word format is enabled, four consecutive long words appear on each line. To the far right of each line, the character corresponding to each byte is also shown. All bytes that contain a non-ASCII code are shown as a period (.).

Legal values for *`size`* are **b** (8-bit byte), **w** (16-bit word), or **l** (32-bit long word).

To terminate the **v** command, press the *space bar*. To "freeze" the display, press any key *except* the space bar. To restart the **v** command again, press any key, *except* space bar.

Monitor **w** Command

w *`virtual_address argument`*

The `set execution vector` command allows you to vector to a *pre-determined* or *default* routine. Optional arguments *`virtual_address`* and *`argument`* are passed along to the to-be-executed routine.

In order to set up a *pre-determined* routine, a user program sets the variable `*romp->v_vector_cmd` equal to the virtual address of the *pre-determined* routine. Variable `*romp->v_vector_cmd` must be set prior to executing the **w** command. *Pre-determined* routines may or may not return to the monitor.

The *default* routine, defined by the monitor, simply prints the user-specified *`virtual_address`* argument according to the user-specified format (given in *`argument`*) before returning to the monitor. The only allowable formats are "%x" and "%d". Format "%x" prints argument *`virtual_address`* as a hexadecimal number; format "%d" prints it as a decimal number.

x

The `extended test system` command invokes the Extended Test Sequence. Refer to Chapter 17 for details.

Monitor **z** Command

z *number breakpoint_virtual_address type len*

Command **z** sets or resets breakpoints for debugging purposes. Optional argument *number* can have values from 0 to 3, corresponding to the processor debug registers DR0 to DR3, respectively. Up to four distinct breakpoints can be specified. If argument *number* is not specified, the monitor will choose a breakpoint number.

The argument *breakpoint_virtual_address* specifies the breakpoint address to set.

The argument *type* can have three values: **x** for Instruction Execution breakpoint, **m** for Data Write only breakpoint, and **r** for Data Reads-and-Writes-only breakpoint. The default value for *type* is **x**.

The argument *len* can also have three values: **b**, **w**, and **l**, corresponding to the breakpoint field length of byte, word, and long-word, respectively. The default value for *len* is `b`. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks.

If the argument *number* is specified but the argument *breakpoint_virtual_address* is not specified, then the corresponding breakpoint will be reset.

This command without any arguments will display all the existing breakpoints.

17

Sun386i Extended Menu Tests

Sun386i Extended Menu Tests **335**

17.1. General Description 335
17.2. Test Fixtures 335
17.3. Sun386i Extended Test Main Menu 337

17

Sun386i Extended Menu Tests

17.1. General Description

The Extended Menu Tests are a set of optionally selectable test routines that may be invoked in response to the prompt

```
Type a character within 10 seconds to enter Menu Tests...
```

when booting the workstation in Diagnostic mode, or they may be invoked by entering the **x** command in response to the standard Monitor prompt.

The Extended Menu Tests provide an additional level of testing beyond the POST diagnostics that are executed at power up.

17.2. Test Fixtures

Some of the Extended Tests require that special connectors are installed on the back of the system. The tables that follow describe these connectors.

Loopback Connector for Serial Port A

A loopback connector is required to test the serial port (ttya) at the back of the workstation when using the Extended Menu Tests. This shorting connector is type DB25P. A list of interconnects is provided below:

Table 17-1 *Serial Port Loopback Signals*

From Pin	*To Pin*
2(TxD)	3(RxD)
4(RTS)	5(CTS)
6(DSR)	7(DTR)

Loopback Connector for Keyboard/Mouse Ports

A loopback connector is required to test the keyboard/mouse connector at the back of the workstation using the Extended Menu Tests. Although the signals to be interconnected are the same, the loopback connector differs, depending on whether the Sun386i under test is equipped with a monochrome or color video board. The monochrome board employs a standard 15 pin connector while the color board adds the RGB signal lines to the keyboard/mouse signal lines. The video signals are ignored in either case and thus the list of necessary interconnects provided below:

Table 17-2 *Keyboard/Mouse Loopback Signals*

From Pin	*To Pin*
1(RxD) mouse	8(TxD) mouse
2(RxD)keyboard	3(TxD)keyboard

Loopback Connector for SCSI Bus

A loopback connector is required to test the SCSI Bus connector at the back of the workstation using the Extended Menu Tests. The SCSI Bus requires a 50 pin connector and the signals which will be interconnected are listed below:

Table 17-3 *SCSI Bus loopback signals*

Source			*Destination*		
Signal	Chip Pin	Connector Pin	Signal	Chip Pin	Connector Pin
ACK	38	P38	DB7	34	P16
BSY	5	P36	DB6	33	P14
SEL	8	P44	DB5	32	P12
ATN	37	P32	DB4	31	P10
REQ	39	P48	DB3	30	P8
MSG	2	P42	DB2	29	P6
C/D	4	P46	DB1	27	P4
I/O	1	P50	DB0	26	P2
RST	36	P40	DBP	25	P18

Loopback Connector for Parallel Port

A loopback connector is required to test the Parallel Port connector at the back of the workstation using the Extended Menu Tests. The Parallel Port requires a 25 pin connector and the signals which will be interconnected as follows:

Table 17-4 *Parallel Port Loopback Signals*

From Pin	*To Pin*
5	15
6	13
7	12
8	10
9	11

17.3. Sun386i Extended Test Main Menu

The main Sun386i Extended Test Menu consists of these options:

Command	Test Name
ie	Intel Ethernet Test
kb	Keyboard Input Test
me	Memory Test
mk	Mouse/Keyboard Test
rs	Serial Port Test
sd	SCSI Disk Bootpath Test
st	SCSI Tape Bootpath Test
fd	Floppy Disk Bootpath Test
vm	Video (Monochrome) Test
vc	Video (Color) Test
cm	Color Map Test
pp	Parallel Port Test

Test Descriptions

The following paragraphs briefly describe each of the tests in the Extended Test menu. To select a test, simply enter the letters shown on the left in the menu and at the beginning of the descriptive paragraph.

`ie`

The *Intel Ethernet Test* checks the Intel 82586 Ethernet Coprocessor chip, the Intel 82501 Serial Interface Unit chip, and the connection to the Ethernet network. This command brings up the Intel Ethernet Test Menu which is described later in this chapter.

`kb`

The *Keyboard Test* command determines whether or not keyboard characters are correctly transmitted to the CPU. After invoking this test, the ASCII code corresponding to a character and the character itself appear on the screen as you press keys on the keyboard. To exit the test, type an ESC character.

`me`

The *Memory Menu Tests* command invokes the Memory Menu. This menu of tests is described later in this chapter.

`mk`

The *Keyboard and Mouse Menu Tests* command invokes the Keyboard and Mouse Menu. This menu contains the keyboard and mouse tests, which are discussed later in this chapter.

`rs`

The *Serial Ports Menu Tests* command invokes the Serial Ports Menu. This menu of tests is discussed later in this chapter.

`sd` The *SCSI Disk Bootpath Test* command tests the bootpath to the SCSI hard disk. It makes sure the machine can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The Options

Menu is described later in this chapter.

st The *SCSI Tape Bootpath Test* command tests the bootpath to the SCSI tape drive. It makes sure the system can boot programs from this device. The test sends a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior.

si

The *SCSI Interface Tests* command checks the the Western Digital WD33C93 SCSI Controller and its addressing and data paths. This command displays the SCSI Interface Menu, which is discussed later in this chapter.

fd Type `fd` from the Main Menu to test the floppy disk bootpath. The test makes sure the system can boot programs from this device. The test performs a boot sequence to the selected device. If the device is present, it reads the boot blocks into system memory without actually executing the boot code. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The Options Menu is described later in this chapter.

vm

The *Monochrome Video Tests* command checks the 128 Kbyte frame buffer space used for the monochrome video display. This command displays the Monochrome Video Menu, which is discussed later in this chapter.

vc

The *Color Video Tests* command checks the 1 Mbyte frame buffer used for color video display. This command displays the Color Video Menu, which is discussed later in this chapter.

cm

The *Color Map Test* command checks the color map table memory space. This command displays the Color Map Menu, discussed later in this chapter.

pp The *Parallel Port Test* checks the data lines and latches leading out to the AT Bus Parallel Port.

Intel Ethernet Test Menu

This menu contains all the tests that check the Intel Ethernet Board.

```
Cmd - Test

l - Local loopback
e - Encoder loopback
x - External loopback

Cmd=>
```

The following paragraphs describe the Intel Ethernet Test Menu selections.

`l` The *local* command tests the Intel 82586 Ethernet LAN co-processor chip. It runs the internal tests built into the chip. Prior to the test, the Intel 82586 Ethernet LAN co-processor chip disconnects itself from the Intel 82501 Serial Interface Unit Chip.

`e` The *encoder* command runs an internal loop back test of the Intel 82501 Serial Interface Unit Chip. The transmitter line, receiver line, noise filters and Manchester encoding/decoding logic are tested. Before executing this test, the transmitter and receiver lines of the chip are connected together.

`x` The *external* command runs the external loop back test. This test checks the Intel 82586 Ethernet LAN Co-processor Chip, the Intel 82501 Serial Interface Unit Chip and the Ethernet transceiver and receiver lines. The test sends data out onto the Ethernet, then receives it back and compares the transmitted and received data. Before running this test, attach an Ethernet transceiver cable to the CPU board and the terminator assemblies to the black transceiver box.

Memory Menu

The Memory Menu contains the options shown below.

```
Enter Cmd [low address >= 100000] [high address <= 3FBFFF] [pattern]

Cmd - Test

 a  - Address Test
 c  - Data Compare Test
 s  - Scan Memory
 w  - Write Only Pattern

Cmd=>
```

`alow_address high_address`

The *Address Test* command executes the address test on a range of memory. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. When the test runs, write-read-compare cycles are performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts two arguments. The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x100000.` The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

`c low_address high_address pattern`

The *Data Compare Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts a maximum of three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x100000`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names the pattern expected throughout the range of addresses. The observed values are compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

s *`low_address high_address pattern`*
The *Scan Memory* command reads the specified range of addresses, and checks for parity errors. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x100000`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names the pattern expected throughout the range of addresses. The observed value is compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*
The *Write Only Pattern* command writes a pattern to a range of addresses. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x100000`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names the pattern written throughout the range of addresses. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

Keyboard and Mouse Menu

This menu selects tests for one of the Sun386i Z8530 Serial Communications Controller (SCC) chips, which controls both the keyboard and the mouse. The Keyboard and Mouse Menu contains four options. These options are presented below:

```
Mouse/Keyboard Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(M or K)] [Baud rate (decimal #)] [hex byte pattern]

Cmd  - Test


w  -  Wr/Rd SCC Reg 12 Test
x  -  Xmit Char Test
i  -  Internal Loopback test
e  -  External Loopback Test


Cmd=>
```

The following paragraphs describe the Mouse/Keyboard Ports menu choices:

w *`channel pattern`*

The *Wr/Rd SCC Reg* command performs write-read-compare cycles to register 12 of the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments.

The *`channel`* argument determines on which port the test is performed. Legal values for channel are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. *`pattern`* should be a hexadecimal (base 16) number.

x *`channel baud pattern`*

The *Xmit* command writes patterns to the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments.

The *`channel`* argument determines which port the test is performed on. Legal values for *`channel`* are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *baud* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number.

The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. *pattern* should be a hexadecimal (base 16) number.

`i` *channel baud pattern*

The *Internal* command performs internal loop back write-read-compare cycles on the port under test. When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. The transmitter and receiver lines of the requested port are connected internally prior to the test.

This command accepts three arguments. The *channel* argument determines on which port the test is performed. Legal values for channel are shown in the table below:

Letter	*Device*
k	Keyboard (default)
m	Mouse

The *baud* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number. The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *pattern* should be a hexadecimal (base 16) number.

e *`channel baud pattern`*

The *`External`* command executes an external loop-back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (see the beginning of this chapter.)

When the test is selected, but before it runs, the Options Menu is displayed, through which you may control the test's behavior. This command accepts three arguments.

The *`channel`* argument determines which port is to be tested. Legal values for *`channel`* are shown in the table below:

Letter	*Device*
`k`	Keyboard (default)
`m`	Mouse

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

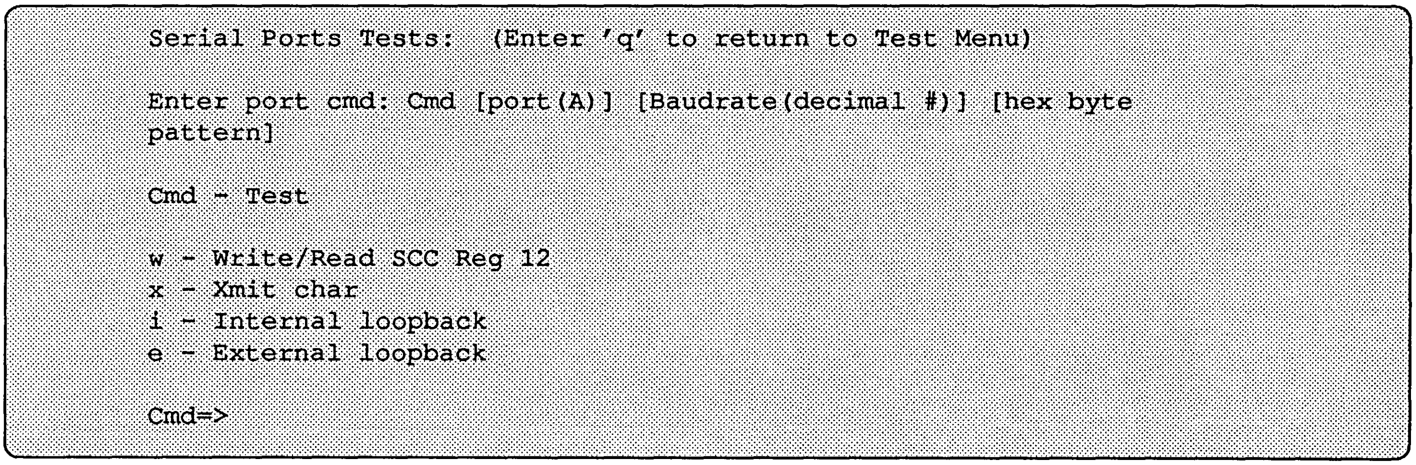
Baud	*Comments*
`300`	
`600`	
`1200`	Default
`2400`	
`4800`	
`9600`	
`19200`	
`38400`	
`76800`	

The value of *`baud`* should be a decimal (base 10) number.

The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

Serial Port Menu

This menu controls testing of the second Z8530 SCC chip in the system, which controls the serial port. The Serial Port Menu contains these options:



```
Serial Ports Tests:  (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(A)] [Baudrate(decimal #)] [hex byte
pattern]

Cmd - Test

w - Write/Read SCC Reg 12
x - Xmit char
i - Internal loopback
e - External loopback

Cmd=>
```

w `channel baud pattern`

The *Wr/Rd SCC Reg* command performs write-read-compare cycles to register 12 of the port under test. When the test is selected, but before it runs, the Options Menu is displayed, allowing you to control the test's behavior. This command accepts three arguments.

The `channel` argument determines which port is to be tested. It is not necessary to enter a parameter for `channel` because the Sun386i has serial port A only, which is tested by default. The appropriate entry for `channel` would be `a`.

The `baud` argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of `baud` should be a decimal (base 10) number.

The `pattern` argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The `pattern` should be a hexadecimal (base 16) number.

x The `Xmit` command writes patterns to the port under test. This command accepts three arguments. The `channel` argument determines which port the test is performed on. For Sun386i, the value for would be `a`, because Port A is the only serial port.

It is not necessary to enter this parameter; Port A is tested by default.

The *baud* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number.

The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *pattern* should be a hexadecimal (base 16) number.

`i` *channel baud pattern*

The `Internal` command performs internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port are connected internally prior to the test.

When the test is selected, but before it runs, the Options Menu is displayed, allowing the user to control the test's behavior.

This command accepts three arguments. The *channel* argument determines on which port the test is performed. For Sun386i, the value would be `a`, because Port A is the only serial port. It is not necessary to enter this parameter; Port A is tested by default.

The *baud* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *baud* should be a decimal (base 10) number.

The *pattern* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *pattern* should be a hexadecimal (base 16) number.

e *`channel baud pattern`*

The `External` command executes an external loop back test on a user-specified port. Write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed (refer to the beginning of this chapter).

When the test is selected, but before it runs, the Options Menu is displayed, allowing the user to control the test's behavior.

This command accepts three arguments. The *`channel`* argument determines on which port the test is performed. Because Port A is the only Sun386i serial port, the appropriate entry would be `a`. It is not necessary to enter this parameter, however; Port A is tested by default.

The *`baud`* argument sets the baud rate at which the test is executed. Legal baud rates are shown in the table below:

Baud	*Comments*
300	
600	
1200	Default
2400	
4800	
9600	
19200	
38400	
76800	

The value of *`baud`* should be a decimal (base 10) number. The *`pattern`* argument specifies which pattern is written to the port. By default, the pattern is `0xaa`. The *`pattern`* should be a hexadecimal (base 16) number.

SCSI Interface Menu

This menu contains the commands that test the Sun386i SCSI interface. It looks something like this:

```
Cmd - Test

b - Reset SCSI Bus
r - SCSI Registers Test
s - Size SCSI Bus

Cmd=>
```

b The *Reset SCSI Bus* command sends a reset signal to the SCSI Bus. All device controllers should be reset. No tests are performed.

c The *SCSI Registers Test* command tests all available host adaptor registers with a default pattern.

s The *Size SCSI Bus* command sizes the SCSI Bus with TEST UNIT READY and INQUIRE commands and prints the results. This test could be used to whether or not a particular device controller is responding.

Monochrome Video Menu

This test is provided to test the 128 Kbyte frame buffer space for the monochrome video display. The following options are available for write/read and address tests.

```
Monochrome Video Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 1FFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test

Cmd=>
```

a *`low_address high_address`*

The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum which is written to each memory "cell" is its own address.

This command accepts two arguments. The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s *`low_address high_address pattern`*

The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors.

The test accepts three optional arguments. The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names the pattern expected throughout the range of addresses. The observed values are compared against this expected value. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

w *`low_address high_address pattern`*

The `Write Only Test` command writes a pattern to a range of addresses. The test accepts three optional arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The *`high_address`* value should be a hexadecimal (base 16) number.

The *`pattern`* argument names the pattern to be written throughout the range of addresses. The default value of *`pattern`* is `0xaaaaaaaa`. The *`pattern`* value should be a hexadecimal (base 16) number.

Color Video Menu

The following options are available.

```
Color Frame Buffer Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0xFFFFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
s - Scan Memory Test
w - Write Only Test

Cmd=>
```

a `low_address high_address`

The *Address Test* command executes the address test on a range of frame buffer memory. Specifically, write-read-compare cycles is performed on long words. The datum that is written to each memory "cell" is its own address. This command accepts two arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only long words are used in the testing.

s `low_address high_address pattern`

The *Scan Memory Test* command reads the specified range of addresses, and checks for parity errors. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names the pattern expected throughout the range of addresses. The observed values are compared against this expected value. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

w `low_address high_address pattern`

The *Write Only Test* command writes a pattern to a range of addresses. The test accepts three optional arguments.

The `low_address` argument specifies the first address to test. By default, the value of `low_address` is `0x0`. The `low_address` value should be a hexadecimal (base 16) number.

The `high_address` argument indicates the final address to test. The default high address is the highest frame buffer memory address available. The `high_address` value should be a hexadecimal (base 16) number.

The `pattern` argument names the pattern to be written throughout the range of addresses. The default value of `pattern` is `0xaaaaaaaa`. The `pattern` value should be a hexadecimal (base 16) number.

Color Map Menu

This menu contains all the tests for the color map. These tests are for the color look-up table (LUT) memory and are similar to the Memory extended tests, with the addition of the Fill Color Map test.

```
Color Map Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low address >= 0] [high address <= 0xFF] [pattern]

Cmd - Test

a - Address Test
c - Write/Read Test
f - Fill color maps with default pattern

Cmd=>
```

a *`low_address high_address`*
The *Address Test* command executes the address test on a range of LUT memory. Specifically, write-read-compare cycles is performed on each byte. The datum that is written to each memory "cell" is its own address. This command accepts two arguments.

The *`low_address`* argument specifies the first address to test. By default, the value of *`low_address`* is `0x0`. The *`low_address`* value should be a hexadecimal (base 16) number.

The *`high_address`* argument indicates the final address to test. The default high address is the length of the LUT channel. available. The *`high_address`* value should be a hexadecimal (base 16) number.

c The *Write/Read Pattern Test* command performs write-read-compare cycles on a range of addresses with a specified pattern. Only bytes are used in the testing.

f The *Fill Color Map* command sets all locations to 0 except

R G B
0 = FF FF FF (White = Background Color)
1 = 0 0 FF (Blue = Logo Color)
2 = 0 FF 0 (Green = Progress Meter Color)

Parallel Port Test Menu

The Parallel Port Test Menu looks something like this:

```
Cmd -  Test

w    Wr/Rd Parallel Port Reg 378
x    Xmit Char
e    External Loopback
```

w The *Write/Read Parallel Port* selection writes and reads a default pattern (W — ASCII 0x57) to Parallel Port Register 0x378.

x *pattern port*
The *Transmit Character* selection writes *pattern* to the *port* entry. By default, the test writes "W" to location 0x378.

e *pattern*
The *External Loopback* selection writes the *pattern* you enter to all three port registers (0x378, 0x379, and 0x37A).

Test Options Submenu

This Submenu is displayed after selecting almost any test. It controls the number of times a test is executed, and what happens when an error is discovered.

```
Test Options: (Enter 'q' to return to Test Menu)

Cmd - Option

f - Loop forever
h - Loop forever with Halt on error
l - Loop once with Loop on error
n - Loop forever with error messages inhibited
<cr> - Loop once

Cmd=>
```

f The *Loop Forever* command runs all tests in an endless loop. Error messages are still reported.

h The *Loop until Error* command makes all tests run in an endless loop until an error is detected. If an error occurs, testing halts.

l The *Loop on Error* command runs the test once. If an error occurs, the test enters a scope loop.

n The *No Error Messages* command runs all tests in an endless loop. No error messages are reported.

<cr>
The *Execute Once* command runs the test once, then returns to the current menu. All errors are reported. Select this option by pressing Return.

18

Sun386i NVRAM Layout

Sun386i NVRAM Layout **355**

18.1. NVRAM Introduction 355
18.2. Changing NVRAM Parameters 356
18.3. The NVRAM Layout 356

18

Sun386i NVRAM Layout

18.1. NVRAM Introduction

The NVRAM (Non-Volatile Random Access Memory) storage area on the Sun386i is implemented with the Thompson MK48T02 2K x 8 Zeropower/Timekeeper RAM. This device performs the same function as the EEPROM on Sun-3 CPU boards. The NVRAM is backed up by an internal battery designed to last up to ten years. As a consequence of this fact and because of the different technology involved between the product families, many of the Sun-3 fields are either not used or reflect the closest approximation to the field description.

The PROM monitor **q** command opens the NVRAM to allow examination or modification of configuration parameters. If you do not enter an address following the command, the content of the first address assigned to the NVRAM is presented. (NVRAM addresses are off-set, rather than complete addresses.)

The `NVRAM edit tool` in the Diagnostic Executive provides a simpler method of changing NVRAM parameters. The System Doctor also reads and writes NVRAM configuration information. This document, however, deals solely with the PROM monitor method of NVRAM editing.

NVRAM parameters set these functions:

- vary the quantity of memory tested during self-test;
- change the action that follows a watchdog reset;
- boot from a specified device during a normal boot, or poll the devices;
- recognize the specified device as the primary terminal or console;
- display the Sun banner or a custom banner during power-up;
- store and display a custom logo upon power-up;
- turn the keyboard "click" on or off;
- select special keyboard characters;
- boot a selected program from a specific device when a diagnostics boot is selected;
- inhibit serial port DTR and RTS signals;
- select a serial port baud rate;
- store a system configuration record on NVRAM;
- erase NVRAM contents;
- store the value in the LED register after an error occurs during self-test;
- in Diagnostic mode, cause a selected power-on self-test (POST) to loop, regardless of test results;
- control self-test modes

18.2. Changing NVRAM Parameters

The *Sun386i PROM Monitor Commands* chapter contains **q** command syntax variations. The paragraphs that follow represent examples of one way to change or view NVRAM parameters. The layout section describes which parameters are stored at which NVRAM addresses.

To change the value of a specific NVRAM address, you must be in the monitor mode, signified by the > prompt. Now, enter the PROM monitor command

```
>q offset_address
```

followed by the offset NVRAM address of the parameter you wish to change, and Return.

When the program displays the contents of that location, enter the new value followed with a non-hexadecimal character, such as a period, or a **q** for quit, and Return:

```
>q 1f Return
>NVRAM 01f: 10? 11 q Return
>
```

To exit from the modify mode when you have *not* entered a new value, simply press the space bar and Return after the question mark.

To increment to the next NVRAM address instead of returning to the PROM monitor program, simply press Return after the question mark, or immediately following your entry.

18.3. The NVRAM Layout

This section has a detailed description of the NVRAM layout. The layout is divided into a diagnostic section, a reserved section, a ROM section, and a software section.

The locations are described first, with tables that illustrate the result of various parameter entries. At the end of each description, the offset addresses are shown with illustrations of the content of each byte in that range. If, for example, the illustration shows `size` as the content of the first byte, the previous text would contain a table of possible hexadecimal values that `size` represents.

Diagnostic NVRAM Test Write Area

Four bytes are provided for the NVRAM portion for the CPU Diagnostic. The contents of these locations after the test are meaningless because these four locations are NOT part of the checksum data area. The Diagnostic area write count locations are updated each time these locations are written.

Address [0x000-0x003]:

Diag Test
Diag Test
Diag Test
Diag Test

Diagnostic Area Write Count

These write counters are for the Diagnostic area of the NVRAM. There are three counters that should contain the same count. The purpose of multiple write counters is reliability of their correctness.

Address [0x004-0x009]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Diagnostic Area Checksum

Each NVRAM area maintains three identical 8 bit (byte) checksums. These separate checksums are intended to be the same.

Address [0x00C-0x00E]:

Checksum #1
Checksum #2
Checksum #3

Date of Last System Hardware Update

This four byte location contains the date of the last system update. This date is recorded in the same fashion as the Manufacturing date: total seconds since 1 January, 1970.

Address [0x010-0x013]:

Date (Bits 31-24)
Date (Bits 23-16)
Date (Bits 15-8)
Date (Bits 7-0)

Mbytes of Installed Memory

This byte contains the total number (in hexadecimal) of Megabytes of memory installed in the system.

Address [0x014]:

Mbytes Installed

Mbytes of Memory to test on Normal Boot

This byte contains the total number (in hexadecimal) of Megabytes of memory that the firmware tests prior to a normal boot. The firmware ignores this value and tests all of memory if the CPU board is set for a diagnostics boot. All of memory is initialized even if not tested.

Address [0x015]:

Mbytes to Test

Monitor Screen Size

This byte selects the appropriate video screen sizes for the Monitor in the system. The following table illustrates the options:

Size	*Definition*
0x00	1152x900 Screen
0x12	1024x1024 Screen
0x13	1600x1280 Screen
0x14	1440x1440 Screen

A hardware change on the CPU Board is necessary to complete a screen size change.

Address [0x016]:

Size

Watchdog Reset Action

This byte selects the appropriate action for the firmware after a Watchdog Reset. The following table illustrates the options:

Action	*Definition*
0x00	Watchdog Reset will fall into Boot PROM Monitor
0x12	Watchdog Reset will cause a Power-On-Reset

Address [0x017]:

Action

Operating System Boot-Up

This byte selects whether the Boot PROM polls for boot devices in the system or uses an NVRAM selectable boot device for loading the Sun Operating System (SunOS) during a normal boot. If the value 0x00 is in this location, the boot PROM will look first attempt to boot from a floppy disk, followed by a tape, hard disk, and Ethernet, in that order. If this byte contains 0x12, the boot device is specified in NVRAM location 0x019-0x01D. The following table illustrates the options:

Boot Device	*Definition*
0x00	Poll devices for the SunOS operating system
0x12	Use NVRAM specified boot device

Address [0x018]:

Boot Device

Boot Device

These five bytes provide for installation of a command string that will boot the operating system from a specified device when NVRAM address 0x018 is set to 0x12, and the diagnostics switch is set to NORM. The locations are assigned as follows:

Address	*Definition*
0x019	Default boot device (1st character converted to hex)
0x01A	Default boot device (2nd character converted to hex)
0x01B	Controller number in Hex
0x01C	Unit number in Hex
0x01D	Partition number in Hex

Use the following table to convert these boot devices from ASCII to Hex:

Operating System Boot Device	*Address[0x019]*	*Address[0x01A]*
xy: Xylogics 450/451 Disk	0x78	0x79
xd: Xylogics Disk (7053)	0x78	0x64
fd: Floppy Disk	0x66	0x64
sd: SCSI Disk	0x73	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address [0x019-0x01D]:

Device
Device
Controller
Unit
Partition

Keyboard Type

This byte is to signify a NON-SUN keyboard type. It is currently ignored by the Boot PROM.

Address [0x01E]:

Keyboard

Primary Terminal

This byte selects the appropriate device to use as the primary terminal or user interface. The following table illustrates the options:

Terminal	*Definition*
0x10	Use Serial Port A
other value	Use video monitor

Address [0x01F]:

Terminal

Display Sun Banner

This byte selects whether to display the Sun banner or a custom banner on the screen when booting. The custom banner is defined in a 80-byte character buffer at NVRAM addresses 0x068-0x0B8. See the paragraphs *Custom Logo Selector* and *Custom Logo* near the end of this chapter for the location and selection of a bit-mapped image that replace the Sun logo. The following table illustrates the banner options:

Value	*Definition*
0x00	Display the Sun Banner
0x12	Display Custom Banner

Address [0x020]:

Value (0x00 or 0x12)

Keyboard Click

This byte selects whether the keyboard should be initialized with its key click option on or off. The following table illustrates the options:

Click	*Definition*
0x00	Turn click OFF
0x12	Turn click ON

Address [0x021]:

Click

Diagnostic Boot Device

These five bytes define the device that the Boot PROM will use when the CPU board is set for a diagnostic boot. The following table illustrates the boot device specification:

Address	*Definition*
0x022	Default boot device (1st character in hex)
0x023	Default boot device (2nd character in hex)
0x024	Controller number in Hex
0x025	Unit number in Hex
0x026	Partition number in Hex

Use the following table to convert these boot devices from ASCII to Hex:

Diagnostic Boot Device	*Address[0x022]*	*Address[0x023]*
xy: Xylogics 450/451 Disk	0x78	0x79
xd: Xylogics Disk (7053)	0x78	0x64
fd: Floppy Disk	0x66	0x64
sd: SCSI Disk	0x73	0x64
ie: Intel Ethernet	0x69	0x65
le: AMD (Lance) Ethernet	0x6C	0x65
st: SCSI 1/4" Tape	0x73	0x74
xt: Xylogics 1/2" Tape	0x78	0x74
mt: Tapemaster 1/2" Tape	0x6D	0x74

Address [0x022-0x026]:

Device
Device
Controller
Unit
Partition

Diagnostic Boot Path

These 40 bytes represent a character buffer for a user specified diagnostic path (i.e. /stand/diag). These ASCII characters are represented by hex values, terminated with 0x00. You would first open address 0x028 and enter the hexadecimal equivalent of the first character in the selected path, and continue on in that manner, ending with 0x00. An ASCII to Hex conversion chart is included at the back of this manual for your convenience.

Address [0x028-0x04F]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	0x00		

High Res Screen Size

These 2 bytes allow the selection of the number of columns and number of rows for the high resolution monitor.

Address [0x050-0x051]:

# of columns
# of rows

SCC Port A Default Baud Rate

This byte selects whether SCC Port A will use the default baud rate of 9600 Baud or the user specified baud rate defined in the Port A Baud Rate NVRAM address[0x059-0x05A]. The following table illustrates the options:

Value	*Definition*
0x00	Use Default Baud rate of 9600 Baud
0x12	Use NVRAM defined Baud rate

Address [0x058]:

Value

SCC Port A Baud Rate

These two bytes define the baud rate at which SCC Port A is initialized if NVRAM address[0x058] has been set to 0x012. These bytes are the hexadecimal equivalent of the baud rate. The following table illustrates hexadecimal equivalents to the various baud rates.

Baud Rate	*Hex Equivalent*	*Address[0x059]*	*Address[0x05A]*
300	0x012C	0x01	0x2C
600	0x0258	0x02	0x58
1200	0x04B0	0x04	0xB0
2400	0x0960	0x09	0x60
4800	0x12C0	0x12	0xC0
9600	0x2580	0x25	0x80
19200	0x4B00	0x4B	0x00
38400	0x9600	0x96	0x00

Address [0x059-0x05A]:

Baud (High byte)
Baud (Low byte)

SCC Port A DTR/RTS

This byte selects whether SCC Port A will have the signals DTR and RTS asserted in the initialization process. The following table illustrates the options:

Value	*Definition*
0x00	Assert the DTR and RTS signals
0x12	Do NOT assert the DTR and RTS signals

Address [0x05B]:

Value

Custom Banner

These 80 bytes represent a character buffer for a user specified custom banner to be displayed instead of the Sun banner, when the value of NVRAM location is 0x020 is 0x012. All locations up to the terminator (0x00) are displayed; each byte not filled with the hexadecimal equivalent of an ASCII character should contain zeroes.

Address [0x068-0x0B7]:

ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
ASCII	ASCII	0x00				

Test Pattern

These two bytes are used to provide a known data test pattern to check the NVRAM data lines.

Address [0x0B8-0x0B9]:

0xAA	0x55

NOTE *Locations 0xBC through 0xFF are not used at this time.*

System Configuration

The content of locations 0x100 through 0x110 represent the hardware configuration of the system. The layout of the system configuration bytes is illustrated in the following table.

This table shows each offset address in the configuration area, followed with the type of information represented by the presence of certain values in that byte. These values may be changed through the PROM monitor `q` command or, more readily, through the Diagnostic Executive `NVRAM edit tool`. The System Doctor also has read and write capability in this area.

Table 18-1 **Sun386i CPU Configuration Area**

NVRAM Location	*Definition*
0x100	CPU 0=not present 1=present
0x101	# of Megabytes memory
0x102	Disk 0=no disk 1=91MB disk 2=372MB disk
0x103	Diskette 0=not present 1=present
0x104	Display 0=not present 1=hi-res color 2=lo-res color 3=lo-res monochrome 4=hi=res monochrome
0x105	Cartridge tape 0=not present 1=present
0x106	Floating point 0=not present 1=present
0x107	checksum for 0x100 through 0x106
0x108-0x19E	unused
0x10F	NVRAM update flag (Diag Exec use only)
0x110	CPU board revision level
0x111	CPU board artwork level 01 = Rev 1.5 02 = Rev 2.0
0x112	ECO rev level
0x113-0x11F	reserved for kernel
0x120-0x18B	unused/reserved

Remaining NVRAM Layout

Key Table Selector

The value of this byte is used to determine the appropriate key tables to be used based on the following table. NVRAM key tables refer to the tables entered in NVRAM addresses 0x190-0x20F (lower case) and 0x210-0x28F (upper case).

Key Table Selector	*Definition*
0x58	Use NVRAM key tables
Other than 0x58	Use PROM Key tables

Address [0x18C]:

Key table selector

NVRAM Locale Specifier

This byte contains the specifier for the locale (country, language, codeset) for which the system is configured. The specifier is placed in this location by the SunOS operating system.

Address [0x18D]:

Locale Specifier

Keyboard ID

The Boot PROM checks the NVRAM Key Table Selector (address 0x18C), and if the value is 0x58, it then compares the value in the address 0x18E with the keyboard type it finds in the system. This byte contains the hard coded keyboard type.

Address [0x18E]:

Keyboard ID

Custom Logo Selector

The value of this byte is used to determine the appropriate Logo bit-map to be displayed upon power-up, based on the following table.

Custom Logo Selector	*Definition*
0x12	Use NVRAM logo bit-map (see Custom Logo, below)
Other than 0x12	Use Sun logo bit-map

Address [0x18F]:

Logo bit-map selector

NVRAM Lower Case Key Table

An array of 128 bytes from address 0x190 can be used for a different lower case key table. This table is used by the firmware if location 0x18C is set to 0x58 and Keyboard ID matches the hard coded ID.

Address [0x190-0x20F]:

128 bytes for lower case key table

NVRAM Upper Case Key Table

An array of 128 bytes from address 0x210 can be used for a different upper case key table. This table is used by the firmware if location 0x18C is set to 0x58 and the Keyboard ID matches the hard coded ID.

Address [0x210-0x28F]:

128 bytes for upper case key table

Custom Logo

This 64x8 matrix may contain a Custom Logo bit-map that can be selected by setting location 0x18F to 0x12.

Address [0x290-0x48F]:

64X8 bytes Custom Logo

LED Register Copy

This byte represents the value of the LED register after an error occurs during the POST phase. The booted program can use this program to track warning errors.

Address [0x490]

LED register error value

This feature can only be used in Diagnostic Mode.

POST Loop on Test

Entering the hexadecimal value of a Power-On Self-Test number (as shown in *Chapter 15*), causes the given test to loop, regardless of test results. The activity meter changes to blue while the test is looping.

Address [0x491]

POST Loop on Test Number

PROM Mode Control

This byte contains the mode control settings for various POST run-time characteristics. Refer to the *Determination of Run-time Mode* in *Chapter 15* for more information on these modes.

The Soft Switch settings are interpreted as follows:

Hex Value	*Description*
0x7	Normal Mode
0x6	Diagnostic Mode

The hexadecimal equivalent of the binary value of Bits 0 - 7 is stored in this byte:

Address [0x492]

POST Mode Control

Auto Configuration Message Flag

This option shields the user from all the boot messages that announce which device drivers, daemons, and so on are being loaded. The value in this byte is interpreted as follows:

Value	*Description*
0x0	no messages
0x1	Sun3 (UNIX-Expert type messages)
0x2	Verbose messages
others	reserved

Address [0x494]

Default Console Colors

This byte is not used by the boot PROM at the present time. This location provides a choice of eight background and foreground colors for color systems. A bit is also included for reverse video.

Bits	*Effect*
Bits 0 - 2	Foreground Color
Bits 3 - 5	Background Color
Bit 6	Intensity Low=0, High=1
Bit 7	Normal video = 0 Reverse video = 1

Values assigned to the color options are:

0 - Black	4 - Amber
1 - Red	5 - Magenta
2 - Green	6 - Cyan
3 - Blue	7 - White

In order to enter a value in this byte, you must select foreground and background colors and convert the decimal values for those colors (shown above) to binary values. Those binary values are placed in the bits assigned to foreground and background colors, and, along with the values selected for intensity and normal/inverse video, create an 8-bit value.

Binary value converted to hexadecimal

For example, if the following choices are made:

Option	*Choice*	*Binary Value*
Foreground Color	Cyan	110
Background Color	White	111
Intensity	High	1
Video	Normal	0

The binary value created by the choices shown above would be:

```
01111110
```

The hexadecimal equivalent of this binary value would be 0x7e and would be entered in:

Address [0x493]

Reserved Area

Write Count

These write counters are for the Reserved area of the NVRAM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address [0x500-0x505]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Reserved Area

Checksum

Each NVRAM area maintains three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address [0x508-0x50A]:

Checksum #1
Checksum #2
Checksum #3

Locations 0x50A - 0x5FF are reserved.

Scratch Area

Address [0x600-0x700]:
This area is reserved for scratchpad use.

Software Area

Write Count

These write counters are for the Software area of the NVRAM. There are three counters that should contain the same count. The purpose of multiple write counters is to insure reliability of their correctness.

Address [0x700-0x705]:

Count #1 (High byte)
Count #1 (Low byte)
Count #2 (High byte)
Count #2 (Low byte)
Count #3 (High byte)
Count #3 (Low byte)

Software Area

Checksum

Each NVRAM area will maintain three identical 8 bit (byte) checksums. These separate checksums are to be the same.

Address [0x708-0x70A]:

Checksum #1
Checksum #2
Checksum #3

Locations 0x70A - 0x7FF are reserved.

A

ASCII/Hex Conversion Chart

ASCII/Hex Conversion Chart 373

A

ASCII/Hex Conversion Chart

This chart is provided for your convenience when programming the EEPROM.

Table A-1 *ASCII/Hex Conversion*

ASCII	*Hex*	*ASCII*	*Hex*	*ASCII*	*Hex*	*ASCII*	*Hex*
nl (line feed)	0A	6	36	N	4E	f	66
cr (return)	0D	7	37	O	4F	g	67
sp (space)	20	8	38	P	50	h	68
!	21	9	39	Q	51	i	69
"	22	:	3A	R	52	j	6A
#	23	;	3B	S	53	k	6B
$	24	<	3C	T	54	l	6C
%	25	=	3D	U	55	m	6D
&	26	>	3E	V	56	n	6E
’	27	?	3F	W	57	o	6F
(	28	@	40	X	58	p	70
)	29	A	41	Y	59	q	71
*	2A	B	42	Z	5A	r	72
+	2B	C	43	[	5B	s	73
,	2C	D	44	\	5C	t	74
-	2D	E	45	]	5D	u	75
.	2E	F	46	^	5E	v	76
/	2F	G	47	_	5F	w	77
0	30	H	48	‘	60	x	78
1	31	I	49	a	61	y	79
2	32	J	4A	b	62	z	7A
3	33	K	4B	c	63	{	7B
4	34	L	4C	d	64	|	7C
5	35	M	4D	e	65	}	7D
						~	7E
						del	7F

Index

A

AMD Ethernet extended test, 107
automatic boot
  Sun-2, 29
  Sun-3, 63
  Sun-4, 173
  Sun386i, 297

B

banner display, 139
boot path Sun-4 extended test, 233
Boot PROM, 3
booting procedures, 4, 11, 16, 17, 203
  Sun-3, 64
  Sun-3 EEPROM selection, 137
  Sun-4 EEPROM selection, 260
  Sun386i, 297
  Sun386i NVRAM, 359

C

changing EEPROM parameters, 131, 255
changing NVRAM parameters, 356
click,keyboard: enable/disable in EEPROM, 139, 262
color map extended test
  Sun-3/110, 126
  Sun386i, 351
color selection — Sun386i, 368
color video extended test
  Sun-3, 122
  Sun386i, 349
commands
  PROM monitor, 33 *thru* 45, 81 *thru* 97, 207 *thru* 223, 319 *thru* 332
  Sun-4 extended tests, 227
country/language specifier
  in EEPROM, 166, 288
  in NVRAM, 366
custom banner
  in EEPROM, 144, 267
  in NVRAM, 364
custom logo
  selection, 167, 289, 366
  stored in EEPROM, 168, 290
  stored in NVRAM, 367

D

diagnostic boot device selection
  in EEPROM, 140, 263
  in NVRAM, 361
diagnostic boot path
  in EEPROM, 141, 264
  in NVRAM, 362
diagnostic power-up, 15
  Sun-3, 65, 67
  Sun-4, 174
  Sun386i, 299
diagnostics switch, 12, 14, 173, 174, 203
DTR/RTS signal selection
  in EEPROM, 142, 144, 265, 267
  in NVRAM, 363

E

EEPROM, 3
  Sun-3, 131 *thru* 170
  Sun-4, 255 *thru* 292
  Sun386i, see NVRAM
EEPROM setting
  console device, 138, 261
  diagnostic boot device, 140, 263
  for action after reset, 15, 136, 259
  for screen size, 136, 259
  high resolution screen, 141, 264
  how to do it, 131, 255
  key table selection, 166, 288
  keyboard click, 139, 262
  keyboard type, 138, 167, 261, 289
  locale specifier, 166, 288
  memory to test, 136, 258
  normal boot sequence, 137, 260
  port A DTR/RTS signals, 142, 265
  port B DTR/RTS signals, 144, 267
  select banner display, 139, 262
  select boot device, 137, 260
  selecting custom logo, 167, 289
  serial port A baud rate, 141, 264
  serial port B baud rate, 143, 266
  store custom logo, 168, 290
  store diagnostic boot path, 141, 264
  store lower case key table, 167, 289
  store upper case key table, 167, 289
  Sun-3 system configuration, 145 *thru* 166
  Sun-4 system configuration, 257, 268 *thru* 288
enable plane extended test — Sun-3/110, 124

`/etc/halt`, 17
Ethernet extended tests
  Sun-2, 52
  Sun-4, 235
extended power-up tests
  Sun-2, 49 *thru* 60
  Sun-3, 101 *thru* 128
  Sun-4, 175, 227 *thru* 252
  Sun386i, 335 *thru* 352

H

halting SunOS, 22
hardware test equipment, 6, 335
high-res screen size
  in EEPROM, 141, 264
  in NVRAM, 362

I

ID PROM, 3
Intel Ethernet extended test, 108, 338
  Sun386i, 337

J

jumper location, Sun386i CPU, 296

K

key table selector
  in EEPROM, 166, 288
  in NVRAM, 366
key tables
  in EEPROM, 167, 289
  in NVRAM, 367
keyboard
  ID — in EEPROM, 167, 289
  ID — in NVRAM, 366
  noise control, in EEPROM, 139, 262
  noise control, in NVRAM, 361
  Sun386i extended test, 337
  type, stored in EEPROM, 138, 261
  type, stored in NVRAM, 360
keyboard/mouse extended test
  Sun-2, 56
  Sun-3, 111
  Sun-4, 237
  Sun386i, 341

L

`L1-a`, 17, 22, 64, 87, 93
LEDs on CPU Board, 14, 174
  Sun-2, 26
  Sun-3, 70
  Sun-4, 180
  Sun386i, 302
logo
  non-Sun, 168, 290
  selection, 167, 289, 366

M

main menu
  Sun-2 extended tests, 50
  Sun-3 extended tests, 102
  Sun-4 extended test system, 230
main menu, *continued*
  Sun386i extended tests, 337
manufacturing burn-in, 15, 175, 300
memory
  EEPROM record of, 136, 258
  NVRAM record, 357
memory extended test
  Sun-2, 54
  Sun-3, 109
  Sun-4, 241
  Sun386i, 337, 339
monitor, 21
  Sun-2 PROM, 33
  Sun-3 PROM, 81
  Sun-4 PROM, 207
  Sun386i PROM, 319
monochrome video extended test
  Sun-3/110, 120
  Sun386i, 338, 347

N

NVRAM, 3, 355 *thru* 370
NVRAM setting
  console device, 360
  diagnostic boot device, 361
  display size, 358
  DTR/RTS signals, 363
  for action after reset, 358
  high resolution screen, 362
  key table selection, 366
  keyboard click, 361
  keyboard type, 360, 366
  locale specifier, 366
  normal Sun386i boot, 359
  run-time mode, 368
  select banner display, 360
  selecting custom logo, 366
  serial port A baud rate, 362
  store diagnostic boot path, 362
  store lower case key table, 367
  store upper case key table, 367
  Sun386i system configuration, 365
  turn messages on/off, 368

P

power-up banner selection
  in EEPROM, 139, 262
  in NVRAM, 360
power-up display, 13
  Sun-2, 25
  Sun-3, 64
  Sun-4, 173
  Sun-4 diagnostic, 177
  Sun386i, 299
primary terminal
  set in EEPROM, 138, 261
  set in NVRAM, 360
programming the EEPROM, 131, 255
PROM monitor, 21
  Sun-2, 33 *thru* 45
  Sun-3, 81 *thru* 97
  Sun-4, 207 *thru* 223
  Sun386i, 319 *thru* 332

PROMs
  overview, 3
  revisions, 5
  Sun-2, 4, 33 *thru* 45
  Sun-3, 5, 81 *thru* 97
  Sun-4, 5, 207 *thru* 223
  Sun386i, 319 *thru* 332

R

remote testing, 66, 176
reset switch, 15, 175
resolution of video display, 136, 358

S

SCSI extended tests — Sun386i, 337
SCSI Interface extended test
  Sun-3, 117
  Sun386i, 338
select boot device
  in EEPROM, 137, 260
  in NVRAM, 359
self-test
  memory, 13, 173
  Sun-2, 25
  Sun-3, 63
  Sun-4, 173
  Sun386i, 295
serial port A baud rate
  in EEPROM, 141, 264
  in NVRAM, 362
serial port B baud rate
  in EEPROM, 143, 266
serial port extended test
  Sun-2, 57
  Sun-3, 114
  Sun-4, 244
  Sun386i, 344
starting the PROM monitor program, 21
store custom banner
  in EEPROM, 144, 267
  in NVRAM, 364
Sun architectures, 4
Sun-1
  description, 4
Sun-2
  description, 4
  extended test main menu, 50
  extended tests, 49 *thru* 60
  monitor commands, 33 *thru* 45
  self-tests, 25 *thru* 29
Sun-3
  description, 5
  EEPROM layout, 131
  extended tests, 101 *thru* 128
  monitor commands, 81 *thru* 97
  self-tests, 63 *thru* 78
Sun-4
  description, 5
  EEPROM layout, 255, 256
  extended tests, 227 *thru* 252
  monitor commands, 207 *thru* 223
  self-tests, 173 *thru* 204
Sun386i
  console color — in NVRAM, 368
  description, 5
  extended tests, 335 *thru* 352
  jumper settings, 296
  mode control, 368
  monitor commands, 323 *thru* 332
  NVRAM, 355 *thru* 370
  power-up messages, 368
  self-tests, 295 *thru* 315
switch
  diagnostics, 12, 14, 65, 173, 174, 203
  reset, 15, 175
`sync`, 17, 22
system configuration
  in NVRAM, 365
  in Sun-3 EEPROM, 145 *thru* 166
  in Sun-4 EEPROM, 257, 268 *thru* 288
system initialization, 28, 314

T

terminal set-up
  for diagnostics, 65, 174
testing
  extended, under boot PROM, 50, 101, 227, 335
  minimum system function, 26, 66, 177, 306

V

video display size
  in EEPROM, 136, 259
  in NVRAM, 358
video extended test
  Sun-2, 58
  Sun-3, 118
  Sun-4, 248

W

watchdog reset, 175
  EEPROM setting, 136, 259
  NVRAM setting, 358

Revision History

Dash Number	Date	Comments
01	April 19, 1987	Alpha Draft (PN 800-1634)
01	July 27, 1987	Alpha Release with SunOS Operating System Release 4.0 (PN 800-1734)
02	October 7, 1987	Alpha5 Review (PN 800-1736)
03	December 15, 1987	ECD Beta Release (PN 800-1736)
04	January 19, 1988	Beta1 Release (PN 800-1736)
10	May 9, 1988	Production Release (PN 800-1736)

PROM USER'S MANUAL READER COMMENT SHEET

Dear Reader,

We who work at Sun Microsystems wish to provide the best possible documentation for our products. To this end, we solicit your comments on the PROM User's Manual. We would appreciate your telling us about errors in the content of the manual, and about any material that you feel should be there but isn't.

Typographical Errors

Please list typographical errors by page number and actual text of the error.

Technical Errors

Please list errors in technical accuracy by page number and actual text of the error.

Content

Did this guide meet your needs? If not, please indicate what you think should be added or deleted in order to do so. Please comment on any material that you feel should be present but is not. Is there material found in other manuals that would be more convenient if it were in this manual?

Layout and Style

Did you find the organization of this guide useful? If not, how would you rearrange things? Do you find the style of this manual pleasing or irritating? What would you like to see different?

Mail this completed form to:

Manager, Hardware Technical Publications
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

Notes

Notes